Finding Smallest Witnesses for Conjunctive Queries

Xiao Hu
University of Waterloo, Canada

Stavros Sintos
University of Illinois Chicago, IL, USA

Abstract

A witness is a sub-database that preserves the query results of the original database but of much smaller size. It has wide applications in query rewriting and debugging, query explanation, IoT analytics, multi-layer network routing, etc. In this paper, 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 $\mathtt{P} = \mathtt{NP}$. 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 [31] – 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 $\mathtt{P} = \mathtt{NP}$.

We further explore efficient approximation algorithms for CQs without head-domination property: (1) we show a trivial algorithm which 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 a much higher lower bound, exponentially larger than the logarithmic lower bound obtained above. It remains open to close the gap between the lower and upper bound of the approximated SWP for CQs without head-domination property.

2012 ACM Subject Classification Theory of computation → Data provenance

Keywords and phrases conjunctive query, smallest witness, head-cluster, head-domination

Digital Object Identifier 10.4230/LIPIcs.ICDT.2024.24

Related Version *Full Version*: `https://arxiv.org/abs/2311.18157`

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 [10, 41, 12], sketch [15], coreset [36] and factorization [34]. The notion of witness has been studied as one form of why-provenance [9, 26, 3] that provides a *proof* for output results, with wide applications in explainable data-intensive analytics. The *smallest witness problem* was first proposed by [32] 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 paper, we consider a generalized 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 generalized smallest witness has many useful applications in practice, such as helping students learn SQL queries [32], query rewriting, query explanation, multi-layer network routing, IoT analytics on edge devices[35]. We mention three application scenarios:

▶ **Example 1.** Alice located at Seattle wants to send the query results of Q over a database D (which is also stored at Seattle) to Bob located at 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 2.** Charlie is a novice at learning SQL in a 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 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 smallest witness problem 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 efforts 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 5.3) 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 paper, we aim to design algorithms that can efficiently compute or approximate the smallest witness for conjunctive queries, and understand the hardness of this problem when such algorithms do not exist.

1.1 Problem Definition

Let $\mathbb{R}$ be a database schema that contains m relations $R_1, R_2, \cdots, R_m$. 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 $A, B, C, A_1, A_2, A_3, \cdots$ etc. to denote the attributes in $\mathbb{A}$ and $a, b, c, \cdots$ etc. to denote their values. Let $\texttt{dom}(A)$ be the domain of attribute $A \in \mathbb{A}$. The domain of a set of attributes $X \subseteq \mathbb{A}$ is defined as $\texttt{dom}(X) = \prod_{A \in X} \texttt{dom}(A)$. Given the database schema $\mathbb{R}$, let D be a given database of $\mathbb{R}$, and let the corresponding relations of $R_1, \cdots, R_m$ be $R_1^D, \cdots, R_m^D$, where R_i^D is a collection of tuples defined on $\texttt{dom}(\mathbb{A}_i)$. 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.

[1] The smallest witness for a single query result [32] has been incorporated into an educational tool (`https://dukedb-hnrq.github.io/`), successfully employed in Duke database courses with 1,000+ undergraduate users. Our generalized version can also be incorporated and save more efforts by showing one small witness for all answers.

R_1

A	B
a1	b1
a2	b2
a3	b2

R_2

B	C
b1	c1
b2	c3
b3	c2
b3	c3

R_3

C	F
c1	f1
c2	f3
c3	f3

R_4

C	H
c1	h1
c2	h1
c3	h1
c3	h2

$Q(D)$

A	C	F
a1	c1	f1
a2	c3	f3
a3	c3	f3

Figure 1 An example of database schema $\mathbb{R} = \{R_1, R_2, R_3, R_4\}$ (with $\mathbb{A} = \{A, B, C, F, H\}$, $\texttt{attr}(R_1) = \{A, B\}$, $\texttt{attr}(R_2) = \{B, C\}$, $\texttt{attr}(R_3) = \{C, F\}$ and $\texttt{attr}(R_4) = \{C, H\}$), a 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. $D' = \{(a_1, b_1), (b_1, c_1), (c_1, f_1), (c_1, h_1)\}$ is the solution to $\texttt{SWP}(Q, D, \langle a_1, c_1, f_1\rangle)$. 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 the solution to $\texttt{SWP}(Q, D)$.

We consider the class of *conjunctive queries without self-joins*:

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

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*). 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 $R_1(\mathbb{A}_1) \bowtie R_2(\mathbb{A}_2) \bowtie \cdots \bowtie R_m(\mathbb{A}_m)$ onto $\mathbf{A}$ (after removing duplicates).

▶ **Definition 4** (Smallest Witness Problem (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 $|D''| < |D'|$.*

Given Q and D, we denote the above problem by $\texttt{SWP}(Q, D)$. See an example in Figure 1. We note that the solution to $\texttt{SWP}(Q, D)$ may not be unique, hence our target simply finds one such solution. We study the data complexity [38] of SWP i.e., the sizes of database schema and query are considered as constants, and the complexity is in terms of input size N. For any CQ Q and database D, the size of query results $|Q(D)|$ is polynomially large in terms of N, and $Q(D)$ can also be computed in polynomial time in terms of N. In contrast, the size of $\texttt{SWP}(Q, D)$ is always smaller than N, while as we see later $\texttt{SWP}(Q, D)$ may not be computed in polynomial time in terms of N. 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, $\texttt{SWP}(Q, D)$ can be computed in polynomial time in terms of $|D|$. As shown later, SWP is not poly-time solvable for a large class of CQs, so we introduce an approximated version:

▶ **Definition 5** (θ-Approximated Smallest Witness Problem (ASWP)). *For CQ Q and database D, it asks to find a subset of tuples $D' \subseteq D$ such that $Q(D') = Q(D)$ and $|D'| \leq \theta \cdot |D^*|$, where D^* is a solution to $\texttt{SWP}(Q, D)$.*

Also, SWP is θ-approximable for Q if, for an arbitrary database D, there is a θ-approximated solution to $\texttt{SWP}(Q, D)$ that can be computed in polynomial time in terms of $|D|$.

1.2 Our Results

Our main results obtained can be summarized as (see Figure 2):

In Section 3, we obtain a dichotomy of computing SWP for CQs. More specifically, SWP for any CQ with head-cluster property (Definition 11) can be solved by a trivial poly-time algorithm, while SWP for any CQ without head-cluster property is NP-hard by resorting to the *NP-hardness of set cover* problem (Section 3.2).

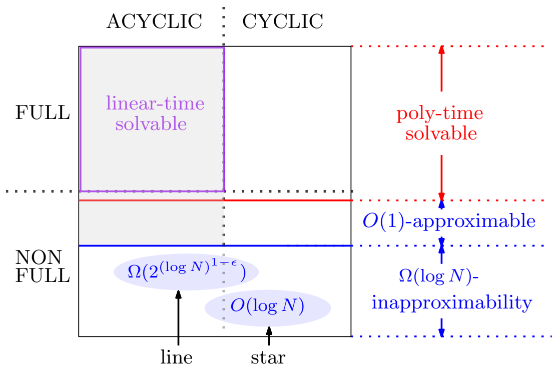

Figure 2 Summary of our results. The shadow area is the class of free-connex CQs.

In Section 4, we show a dichotomy of approximating SWP for CQs without head-cluster property. The *head-domination* property that has been identified for *deletion propagation* problem [31], also captures the hardness of approximating SWP. We show a poly-time algorithm that can return a $O(1)$-approximated solution to SWP for CQs with 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 head-domination property, unless $\mathtt{P} = \mathtt{NP}$, by resorting to the *logarithmic inapproximability of set cover* problem (in Section 4.2). Interestingly, this separation on approximating SWP for *acyclic* CQs (in Section 2.1) coincides with the separation of *free-connex* and *non-free-connex* CQs (in Section 4.1) in the literature. In Section 5, we further explore approximation algorithms for CQs without head-domination property. Firstly, we show a baseline of returning the union of witnesses for every query result leads to a $O(N^{1-1/\rho^*})$-approximated solution, where ρ^* is the fractional edge covering number[2] of the input CQ [4]. 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 a $O(\log N)$ 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 (see full version [28]). Meanwhile, we observe that SWP problem for line queries is a special case of the *directed Steiner forest* (DSF) problem (Section 5.3), 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.

2 Preliminaries

2.1 Notations and Classifications of CQs

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

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

▶ **Definition 6** (Acyclic CQs [6, 19]). *A CQ Q is acyclic 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 $A \in \texttt{attr}(Q)$, the set of nodes containing A forms a connected subtree of $\mathbb{T}$. Such a tree is called the* join tree *of Q.*

▶ **Definition 7** (Free-connex CQs [5]). *A CQ Q is free-connex if Q is acyclic and the resulted CQ by adding another relation contains exactly $\texttt{head}(Q)$ to Q is also acyclic.*

2.2 SWP for One Query Result

The SWP problem for one query result is formally defined as:

▶ **Definition 8** (SWP for One Query Result). *For CQ Q, database D and query result $t \in Q(D)$, it asks for finding a subset of tuples $D' \subseteq D$ such that $t \in Q(D')$, while there is no subset $D'' \subseteq D$ such that $t \in Q(D'')$ and $|D''| < |D'|$.*

Given Q, D and t, we denote the above problem by $\texttt{SWP}(Q, D, t)$. It has been shown by [32] that $\texttt{SWP}(Q, D, t)$ can be computed in polynomial time for arbitrary Q, D, and $t \in Q(D)$. Their algorithm [32] simply finds an arbitrary full join result $t' \in \Join_{R_i \in \texttt{rels}(Q)} R_i$ such that $\pi_{\texttt{head}(Q)} t' = t$, and returns all participating tuples in $\left\{\pi_{\texttt{attr}(R_i)} t' : R_i \in \texttt{rels}(Q)\right\}$ as the smallest witness for t. This primitive is used in building our SWP algorithm. The SWP problem for a Boolean CQ ($\mathbf{A} = \emptyset$, indicating whether the result of underlying natural join is empty or not) can be solved by finding SWP for an arbitrary join result in its full version.

2.3 Notions of Connectivity

We give three 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 $R_i, R_j \in \texttt{rels}(Q)$ if $\texttt{attr}(R_i) \cap \texttt{attr}(R_j) \neq \emptyset$. 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 form a connected subquery of Q. Given a disconnected CQ Q, let $Q_1, Q_2, \cdots, Q_k$ 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 9 follows. In the remaining of this paper, we assume that Q is connected.

▶ **Lemma 9.** *For a disconnected CQ Q of k connected components $Q_1, Q_2, \cdots, Q_k$, 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. We capture the existential-connectivity of a CQ Q by modeling it as a graph $G^{\exists}_Q$, where each relation R_i with $\texttt{attr}(R_i) - \texttt{head}(Q) \neq \emptyset$ is a vertex, and there is an edge between $R_i, R_j \in \texttt{rels}(Q)$ if $\texttt{attr}(R_i) \cap \texttt{attr}(R_j) - \texttt{head}(Q) \neq \emptyset$. We can find the connected components of $G^{\exists}_Q$ by applying search algorithm on $G^{\exists}_Q$, and finding all connected components for $G^{\exists}_Q$. Let $E_1, E_2, \cdots, E_k \subseteq \texttt{rels}(Q)$ be the connected components of $G^{\exists}_Q$, each corresponding to a subset of relations in Q.

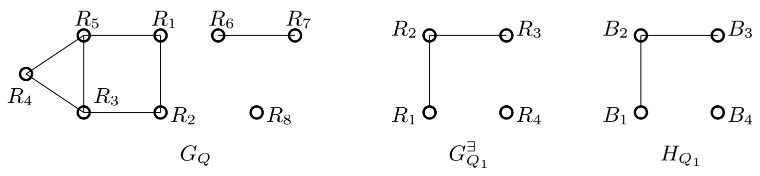

Figure 3 An illustration of G_Q for $Q(A_1, A_2, A_3, A_4, A_5) :- R_1(A_1, B_1), R_2(B_1, B_2), R_3(A_2, B_2, B_3), R_4(A_2, A_3, B_4), R_5(A_1, A_2), R_6(A_4, B_5), R_7(B_5, A_5), R_8(B_6, B_7)$ with three sub-queries $Q_1(A_1, A_2, A_3) :- R_1(A_1, B_1), R_2(B_1, B_2), R_3(A_2, B_2, B_3), R_4(A_2, A_3, B_4), R_5(A_1, A_2)$, 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 middle is $G^{\exists}_{Q_1}$ for Q_1, with two connected components $\{R_1, R_2, R_3\}$, $\{R_4\}$ and dominants R_5, R_4. The right is H_{Q_1} for Q_1, with two connected components $\{B_1, B_2, B_3\}, \{B_4\}$.

Nonout-Connectivity of CQ. We capture the nonout-connectivity of a CQ Q by modeling it as a graph H_Q, where each non-output attribute $A \in \texttt{attr}(Q) - \texttt{head}(Q)$ is a vertex, and there is an edge between $A, B \in \texttt{attr}(Q) - \texttt{head}(Q)$ if there exists a relation $R_i \in \texttt{rels}(Q)$ such that $A, B \in \texttt{attr}(R_i)$. We can find the connected components of H_Q, and finding all connected components for H_Q. Let $H_1, H_2, \cdots, H_k \subseteq \texttt{attr}(Q) - \texttt{head}(Q)$ be the connected components of H_Q, each corresponding to a subset of non-output attributes in Q.

2.4 Head Cluster and Domination

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 $E \subseteq \texttt{rels}(Q)$ of relations, $R_i \in \texttt{rels}(Q)$ is a *dominant* relation for E if every output attribute appearing in any relation of E also appears in R_i, i.e., $\cup_{R_j \in E}\texttt{head}(R_j) \subseteq \texttt{head}(R_i)$.

▶ **Definition 10** (Head Domination [30]). *For CQ Q, let $E_1, E_2, \cdots, E_k$ be the connected components of $G^{\exists}_Q$. Q has head-domination property if for any $i \in [k]$, there exists a dominant relation from $\texttt{rels}(Q)$ for E_i.*

The notion of head-domination property has been first identified for deletion propagation with side effect problem [31], which studied the smallest number of tuples to remove so that a subset of desired query results must disappear while maintain as many as remaining query results. We give a detailed comparison between SWP and deletion propagation in the full version [28], although they solve completely independent problem for CQs without self-joins.

▶ **Definition 11** (Head Cluster). *For CQ Q, let $E_1, \cdots, E_k$ be connected components of $G^{\exists}_Q$. 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 simpler definition for head-cluster property: A CQ Q has head-cluster property if for every pair of relations $R_i, R_j \in \texttt{rels}(Q)$ with $\texttt{head}(R_i) \neq \texttt{head}(R_j)$, there must be $\texttt{attr}(R_i) \cap \texttt{attr}(R_j) \subseteq \texttt{head}(Q)$. Here, we define head-cluster property based on dominant relation, since it is a special case of head-domination property.

3 Dichotomy of Exact SWP

In this section, we focus on computing SWP exactly for CQs, which can be efficiently done if *head-cluster* property is satisfied. All missing proofs are given in the full version [28].

▶ **Theorem 12.** *If a CQ Q has head-cluster property, SWP is poly-time solvable; otherwise, SWP is not poly-time solvable, unless P = NP.*

Algorithm 1 $\texttt{SWP}(Q, D)$.

```
1  $D' \leftarrow \emptyset$;
2  $(E_1, E_2, \cdots, E_k) \leftarrow$ connected components of $G_Q^{\exists}$;
3  $\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_k \leftarrow$ output attributes of $E_1, E_2, \cdots, E_k$;
4  foreach $R_j \in \texttt{rels}(Q)$ with $\texttt{attr}(R_j) \subseteq \texttt{head}(Q)$ do
5      $D' \leftarrow D' \cup \pi_{\texttt{attr}(R_j)} Q(D)$;
6  foreach $i \in [k]$ do
7      Define $Q_i(\mathbf{A}_i) : -\{R_j(\mathbb{A}_j) : R_j \in E_i\}$;
8      foreach $t' \in \pi_{\mathbf{A}_i} Q(D)$ do
9          $D' \leftarrow D' \biguplus \texttt{SWP}(Q_i, \{R_j : R_j \in E_i\}, t')$;
10 return $D'$;
```

3.1 An Exact Algorithm

We prove the first part of Theorem 12 with a poly-time algorithm. The head-cluster property implies that if two relations have different output attributes, they share no common non-output attributes. This way, we can cluster relations by output attributes. As shown, Algorithm 1 partitions all relations into $\{E_1, E_2, \cdots, E_k\}$ based on the connected components in $G_Q^{\exists}$. For the subset of relations in one connected component E_i, every relation is a dominant relation, i.e., shares the same output attributes. If one relation only contains output attributes $\mathbf{A}_i$ (line 4), it must appear alone as a singleton component, since we assume there is no duplicate relations in the input CQ. All tuples from such a relation that participate in any query results must be included by every witness to $Q(D)$. We next consider the remaining components containing at least two relations. In E_i, for each tuple $t' \in \pi_{\mathbf{A}_i} Q(D)$ in the projection of query results onto the output attributes $\mathbf{A}_i$, Algorithm 1 computes the smallest witness for t' in sub-query Q_i defined on relations in E_i. The disjoint union ($\biguplus$) of witnesses returned for all groups forms the final witness. On a CQ with head-cluster property, Algorithm 1 can be stated in a simpler way (see the full version [28]). Algorithm 1 runs in polynomial time, as (i) $Q(D)$ can be computed in polynomial time; (ii) $|Q(D)|$ is polynomially large; and (iii) the primitive in line 9 only takes $O(1)$ time.

▶ **Lemma 13.** *For a CQ Q with head-cluster property, Algorithm 1 finds a solution to $\texttt{SWP}(Q, D)$ for any database D in polynomial time.*

Proof. We prove that for any CQ Q with head-cluster property and an arbitrary database D, Algorithm 1 returns the solution to $\texttt{SWP}(Q, D)$. Together with the fact that Algorithm 1 runs in polynomial time, we finish the proof for Lemma 13. Let D' be the solution returned by Algorithm 1. Let $D'_i \subseteq D'$ be the set of tuples from relations in E_i. We show that D' is a witness for $Q(D)$, i.e., $Q(D') = Q(D)$.

Direction $\subseteq$. As Q is monotone, $Q(D') \subseteq Q(D)$ holds for any sub-database $D' \subseteq D$.

Direction $\supseteq$. Consider an arbitrary query result $t \in Q(D)$. Let $D'_i(t)$ denote the group of tuples returned by $\texttt{SWP}(Q_i, \{R_j : R_j \in E_i\}, \pi_{\mathbf{A}_i} t)$. We note that $t \in \pi_{\mathbf{A}} \Join_{i \in [k]} D'_i(t)$, since

- every tuple has the same value $\pi_A t$ over any output attribute A, if it contains attribute A;
- there is no non-output attribute to join for tuples across groups;
- tuples inside each group can be joined by non-output attribute; (implied by the correctness of $\texttt{SWP}$ for a single query result)

Hence, $t \in Q(D')$. Together, $Q(D') \supseteq Q(D)$.

We next show that there exists no $D'' \subseteq D$ such that $Q(D'') = Q(D)$ and $|D''| < |D'|$. Suppose not, let $D''_i \subseteq D''$ denote the set of tuples from relations in E_i. As $|D''| < |D'|$, there must exist some $i \in [k]$ such that $|D''_i| < |D'_i|$, i.e., $D'_i - D''_i \neq \emptyset$. In Algorithm 1, we can rewrite D'_i as follows:

$$D'_i = \biguplus_{t' \in \pi_{\mathbf{A}_i} Q(D)} \texttt{SWP}(Q_i, \{R_i : R_i \in E_i\}, t'),$$

where $\biguplus$ denotes the disjoint union. As $D'_i - D''_i \neq \emptyset$, there must exist some $t' \in \pi_{\mathbf{A}_i} Q(D)$ such that $t' \notin Q_i(D''_i)$, i.e., t' cannot be witnessed by D''_i. Correspondingly, there must exist some query result t with $\pi_{\mathbf{A}_i} t = t'$ such that $t \notin Q(D'')$, i.e. t cannot be witnessed by D'', contradicting the fact that $Q(D'') = Q(D)$. Hence, no such D'' exists. ◀

▶ **Remark 1.** `SWP` is poly-time solvable for any full CQ, since $\texttt{attr}(R_i) \cap \texttt{attr}(R_j) \subseteq \texttt{head}(Q)$ holds for every pair of relations $R_i, R_j \in \texttt{rels}(Q)$. Hence, the hardness of `SWP` comes from projection. On the other hand, `SWP` is also poly-time solvable for some non-full CQs, say $Q(A_1, A_2, A_3) : -R_1(A_1, A_2), R_2(A_2, A_3), R_3(A_1, A_3), R_4(A_1, B_1)$.

▶ **Remark 2.** It is not always necessary to compute $Q(D)$ as Algorithm 1 does. We actually have much faster algorithms for some classes of CQs. If CQ Q is full, $\texttt{SWP}(Q, D)$ is the set of *non-dangling* tuples in D, i.e., those participate in at least one query result of $Q(D)$. Furthermore, if Q is an acyclic full CQ, non-dangling tuples can be identified in $O(|D|)$ time [40]. It is more expensive to identify non-dangling tuples for cyclic full CQs, for example, the PANDA algorithm [2] can identify non-dangling tuples for any full CQ in $O(N^w)$ time, where $w \geq 1$ is the sub-modular width of input query [2]. It is left as an interesting open question to compute `SWP` for CQs with head-cluster property more efficiently.

3.2 Hardness

We next prove the second part of Theorem 12 by showing the hardness for CQs without head-cluster property. Our hardness result is built on the NP-hardness of set cover [7]: 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. We start with $Q_{\textsf{cover}}(A) : -R_1(A, B), R_2(B)$ and then extend to any CQ without head-cluster property.

▶ **Lemma 14.** *`SWP` is not poly-time solvable for $Q_{\textsf{cover}}(A) : -R_1(A, B), R_2(B)$, unless $\texttt{P} = \texttt{NP}$.*

Proof. We show a reduction from set cover to `SWP` for $Q_{\textsf{cover}}$. Given an arbitrary instance $(\mathcal{U}, \mathcal{S})$ of set cover, we construct a database D for $Q_{\textsf{cover}}$ as follows. For each element $u \in \mathcal{U}$, we add a value a_u to $\texttt{dom}(A)$; for each subset $S \in \mathcal{S}$, we add a value b_S to $\texttt{dom}(B)$, 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 $Q_{\textsf{cover}}(D) = \mathcal{U}$. It is now to show that $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if $\texttt{SWP}(Q_{\textsf{cover}}, D)$ has a solution D' of size $\leq |\mathcal{U}| + k$. Then, if `SWP` is poly-time solvable for $Q_{\textsf{cover}}$, set cover is also poly-time solvable, which is impossible unless $\texttt{P} = \texttt{NP}$. ◀

▶ **Lemma 15.** *For a CQ Q without head-cluster property, `SWP` is not poly-time solvable for Q, unless $\texttt{P} = \texttt{NP}$.*

Proof. Consider such a CQ Q with a desired pair of relations $R_i, R_j \in \texttt{rels}(Q)$. We next show a reduction from $Q_{\textsf{cover}}$ to Q. Given an arbitrary database $D_{\textsf{cover}}$ over $Q_{\textsf{cover}}$, we construct a database D over Q as follows. First, it is always feasible to identify attribute $A' \in \texttt{head}(R_i) - \texttt{attr}(R_j)$ and attribute $B' \in \texttt{attr}(R_i) \cap \texttt{attr}(R_j) - \texttt{head}(Q)$. We set $\texttt{dom}(A') = \texttt{dom}(A)$, $\texttt{dom}(B') = \texttt{dom}(B)$, and remaining attributes with a dummy value $\{*\}$.

Each relation in Q degenerates to $R_1(A, B)$, or $R_2(B)$, or a dummy tuple $\{*, *, \cdots, *\}$. It can be easily checked that there is a one-to-one correspondence between solutions to $\texttt{SWP}(Q, D)$ and solutions to $\texttt{SWP}(Q_{\mathsf{cover}}, D_{\mathsf{cover}})$. 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 14. ◀

4 Dichotomy of Approximated SWP

As it is inherently difficult to compute SWP exactly for general CQs, the next interesting question is to explore approximated solutions for SWP. In this section, we establish the following dichotomy for approximating SWP. All missing proofs are given in the full version [28].

▶ **Theorem 16.** *If a CQ Q has head-domination property, SWP is $O(1)$-approximable; otherwise, SWP of input size N is not $(1 - o(1)) \cdot \log N$-approximable, unless $\mathtt{P} = \mathtt{NP}$.*

4.1 A $O(1)$-Approximation Algorithm

Let's start by revisiting $Q_{\mathsf{cover}}(A) : -R_1(A, B), R_2(B)$. Although SWP is hard to compute exactly for Q_{cover}, it is easy to approximate $\texttt{SWP}(Q_{\mathsf{cover}}, D)$ for arbitrary database D within a factor of 2. Let D^* be a solution to $\texttt{SWP}(Q_{\mathsf{cover}}, D)$. 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 $a \in Q_{\mathsf{cover}}(D)$, and show that $|D'| \leq 2 \cdot |Q_{\mathsf{cover}}(D)| \leq 2 \cdot |D^*|$. This is actually not a violation to the inapproximability of set cover problem. If revisiting the proof of Lemma 14, $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if $\texttt{SWP}(Q_{\mathsf{cover}}, D)$ has a solution of size $\leq |\mathcal{U}| + k$. Due to the fact that $k \leq |\mathcal{U}|$, the inapproximability of set cover does not carry over to SWP for Q. This observation can be generalized to all CQs with head-domination property.

As described in Algorithm 1, an approximated solution to $\texttt{SWP}(Q, D)$ for a CQ Q with head-domination property consists of two parts. For every relation that only contains output attributes, Algorithm 1 includes all tuples that participate in at least one query result (line 4–5), which must be included by any witness for $Q(D)$. For the remaining relations, Algorithm 1 partitions them into groups based on the existential connectivity. Intuitively, every pair of relations across groups can only join via output attributes in their dominants. Recall that $\mathbf{A}_i$ denotes the set of output attributes appearing in relations from E_i. Then, for each group E_i, we consider each tuple $t' \in \pi_{\mathbf{A}_i} Q(D)$ and find the smallest witness for t' in Q_i defined by relations in E_i. The union of witnesses returned for all groups forms the final answer. As shown before, Algorithm 1 runs in polynomial time.

▶ **Lemma 17.** *For a CQ Q with head-domination property, Algorithm 1 finds a $O(1)$-approximated solution to $\texttt{SWP}(Q, D)$ for any database D in polynomial time.*

Proof. Consider the connected components $E_1, E_2, \cdots, E_k$ of $G_Q^{\exists}$ with dominants $\dot{R}_1, \dot{R}_2, \cdots, \dot{R}_k$. Let Q_i be the subquery defined over relations in E_i, with output attributes $\texttt{head}(Q_i) = \texttt{attr}(\dot{R}_i)$, and $D_i = \{R_j : R_j \in E_i\}$ be the corresponding database for Q_i. Let D^* be the solution to $\texttt{SWP}(Q, D)$. We point out some observations on D^*:

- For each R_j with $\texttt{attr}(R_j) \subseteq \texttt{head}(Q)$, D^* must include all tuples in $\pi_{\texttt{attr}(R_j)} Q(D)$.
- For every dominant $\dot{R}_i$, D^* must include at least $|\pi_{\texttt{head}(\dot{R}_i)} Q(D)|$ tuples from $\dot{R}_i$.

On the other hand, Algorithm 1 includes $|\texttt{rels}(Q_i)|$ tuples for each primitive at line 11, and invokes this primitive for each tuple in $\pi_{\texttt{head}(\dot{R}_i)} Q(D)$. Together, we come to:

$$\begin{aligned} |D'| &= \sum_{R_j : \texttt{attr}(R_j) \subseteq \texttt{head}(Q)} \left|\pi_{\texttt{attr}(R_j)} Q(D)\right| + \sum_{\dot{R}_i : i \in [k]} \left|\pi_{\texttt{head}(\dot{R}_i)} Q(D)\right| \cdot |\texttt{rels}(Q_i)| \\ &\leq |D^*| + |D^*| \cdot |\texttt{rels}(Q_i)| \leq 2 \cdot |D^*| \cdot |\texttt{rels}(Q)| \end{aligned}$$

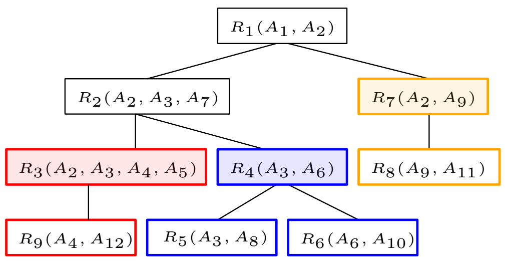

Figure 4 A free-connex CQ Q with $\texttt{head}(Q) = \{A_1, A_2, A_3, A_7\}$. A partition of relations containing non-output attributes is $\{\{R_3, R_9\}, \{R_4, R_5, R_6\}, \{R_7, R_8\}\}$, with dominant relations R_3, R_4, R_7 respectively.

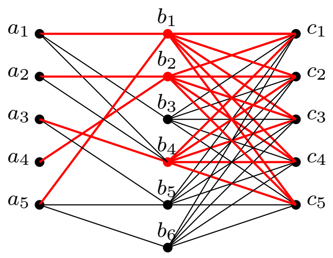

Figure 5 A database D for Q_{matrix} with an integral sub-database in red. Each vertex is a value in the attribute, and each edge is a tuple in D.

It can also be easily checked that $Q(D) = Q(D')$. Hence, Algorithm 1 always returns a $O(1)$-approximation solution for $\texttt{SWP}(Q, D)$. Moreover, the query results $Q(D)$ can be computed in poly-time in the input size of D. Each primitive of finding the smallest witness for one query result takes $O(1)$ time. Hence, Algorithm 1 runs in polynomial time. ◀

Connection with Free-connex CQs. We point out that every free-connex CQ has head-domination property, which is built on an important property as stated in Lemma 18.

▶ **Lemma 18** ([5]). *A free-connex CQ has a tree structure $\mathbb{T}$ such that (1) each node of $\mathbb{T}$ corresponds to $\texttt{attr}(R_i)$ or $\texttt{head}(R_i)$ for some relation $R_i \in \texttt{rels}(Q)$; and for each relation $R_i \in \texttt{rels}(Q)$, there exists a node of $\mathbb{T}$ corresponding to $\texttt{attr}(R_i)$; (2) for every attribute $A \in \texttt{attr}(Q)$, the set of nodes containing A form a connected subtree of $\mathbb{T}$; (3) there is a connected subtree $\mathbb{T}_{\mathsf{con}}$ of $\mathbb{T}$ including the root of $\mathbb{T}$, such that the set of attributes appearing in $\mathbb{T}_{\mathsf{con}}$ is exactly $\texttt{head}(Q)$.*

▶ **Lemma 19.** *Every free-connex CQ has head-domination property.*

Proof. Recall that Q has head-domination property if for any connected component E_j of $G_Q^{\exists}$, there exists a dominant relation from $\texttt{rels}(Q)$ for E_j. Suppose we are given a tree structure $\mathbb{T}$ for a free-connex CQ Q with $\mathbb{T}_{\mathsf{con}}$ as described by Lemma 18. Consider an arbitrary relation R_i with $\texttt{attr}(R_i) - \texttt{head}(R_i) \neq \emptyset$, such that all its ancestors in $\mathbb{T}$ only contain output attributes. Let $\mathbb{T}_i$ be the subtree rooted at R_i. We note that any relation in $\mathbb{T}_i$ cannot fall into the same connected component with a relation in $\mathbb{T} \setminus \mathbb{T}_i$, since they do not share any common non-output attribute. Consider any relation $R_j \in \mathbb{T}_i$. Implied by the fact that $\mathbb{T}_{\mathsf{con}}$ is a connected subtree and $R_i \notin \mathbb{T}_{\mathsf{con}}$, we have $R_j \notin \mathbb{T}_{\mathsf{con}}$. Then, $\texttt{head}(R_j) \subseteq \texttt{head}(R_i)$; otherwise, any output attribute in $\texttt{head}(R_j) - \texttt{head}(R_i)$ does not appear in $\mathbb{T}_{\mathsf{con}}$. Hence, for any connected component E_j formed by relations from $\mathbb{T}_i$, R_i is a dominant relation for E_j. This argument applies for every connected component of $G_Q^{\exists}$. ◀

4.2 Logarithmic Inapproximability

Now, we turn to the class of CQs without head-domination property, and show their hardness by resorting to inapproximability of set cover [20, 17]: there is no poly-time algorithm for approximating set cover of input size n within factor $(1 - o(1)) \cdot \log n$, unless $\texttt{P} = \texttt{NP}$. We identify two hardcore structures (Definition 23 and Definition 26), and prove that no poly-time algorithm can approximate $\texttt{SWP}$ for any CQ containing a hardcore within a logarithmic factor, unless $\texttt{P} = \texttt{NP}$. Lastly, we complete the proof of Theorem 16 by establishing the connection between the non-existence of a hardcore and head-domination property for CQs.

$$(O1) \quad \min \sum_{b\in B} |C_b| \quad \text{s.t.} \bigcup_{b:(a,b)\in R_1} C_b = C, \forall a \in A; \quad C_b \subseteq C, \forall b \in B$$

$$(O2) \quad \min \sum_{b\in B} x_b \quad \text{s.t.} \sum_{b:(a,b)\in R_1} x_b \geq 1, \forall a \in A; \quad x_b \in \{0,1\}, \forall b \in B$$

Figure 6 Optimization problems in the proof of Lemma 21.

4.2.1 Free sequence

Let's start with the simplest acyclic but non-free-connex CQ $Q_{\mathsf{matrix}}(A, C) :- R_1(A, B), R_2(B, C)$. 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, such that for any database D where R_2 is a Cartesian product, $\texttt{SWP}(Q_{\mathsf{matrix}}, D)$ always admits an integral witness.

▶ **Definition 20** (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 21.** *For Q_{matrix} and any database D where $R_2 = (\pi_B R_2) \times (\pi_C R_2)$, there is an integral solution to $\texttt{SWP}(Q_{\mathsf{matrix}}, D)$, i.e., a smallest witness to $Q(D)$ that is also integral.*

Proof. Consider a database D where $R_2 = (\pi_B R_2) \times (\pi_C R_2)$. We assume that there exists no dangling tuples 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 6. 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 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 5.

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 6. Let x^* be the optimal solution of the program above, which only depends on the input relation R_1, and 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 $\texttt{SWP}(Q_{\mathsf{matrix}}, D)$ 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 witness to $\texttt{SWP}(Q_{\mathsf{matrix}}, D)$. ◀

▶ **Lemma 22.** *There is no poly-time algorithm to approximate SWP for Q_{matrix} within a factor of $(1 - o(1)) \cdot \log N$, unless P = NP.*

Proof. 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 $\texttt{dom}(A)$ and c_u to $\texttt{dom}(C)$; for each subset $S \in \mathcal{S}$, we add a value b_S to $\texttt{dom}(B)$. Moreover, for each pair $(u, S) \in \mathcal{U} \times \mathcal{S}$ with $u \in S$, we add tuple (a_u, b_S)

[3] The 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 [33].

to R_1. R_2 is a Cartesian product between $\texttt{dom}(B)$ and $\texttt{dom}(C)$. Every relation contains at most n^{c+1} tuples. Hence $N \leq 2n^{c+1}$. Note that $Q_{\mathsf{matrix}}(D)$ is the Cartesian product between A and C. Implied by Lemma 21, it suffices to consider integral witness to $\texttt{SWP}(Q_{\mathsf{matrix}}, D)$. Here, we show that $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if the $\texttt{SWP}(Q_{\mathsf{matrix}}, D)$ 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 $\texttt{SWP}(Q_{\mathsf{matrix}}, D)$ as follows. Let $B' \subseteq \texttt{dom}(B)$ be the set of values that corresponding 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 $a_u \in \texttt{dom}(A)$, 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'.

Direction if. Suppose we are given a integral solution D' of size k' to $\texttt{SWP}(Q_{\mathsf{matrix}}, D)$. 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' forms a valid cover of size $\frac{k'}{n-1}$. By definition of integral solution, $|B'| = \frac{k'}{n} - 1$. Moreover, for every value $a \in \texttt{dom}(A)$, 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 $\texttt{SWP}$ is $(1 - o(1)) \cdot \log N$-approximable for Q_{matrix}, there is a poly-time algorithm that approximates set cover of input size n within a $(1 - o(1)) \cdot \log N$ factor, which is impossible unless $\texttt{P} = \texttt{NP}$. ◀

▶ **Definition 23** (Free Sequence). *In a CQ Q, a free sequence is a sequence of attributes $P = \langle A_1, A_2, \cdots, A_k \rangle$ such that*[4]

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

▶ **Lemma 24.** *For a CQ Q containing a free sequence, there is no poly-time algorithm that can approximate $\texttt{SWP}$ for Q within a factor of $(1 - o(1)) \cdot \log N$, unless $\texttt{P} = \texttt{NP}$.*

Proof. Let $P = \langle A_1, A_2, \cdots, A_k \rangle$ be such a sequence. For simplicity, let $P' = \{A_2, A_3, \cdots, A_{k-1}\}$. We next show a reduction from set cover to $\texttt{SWP}(Q, D)$. 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 $\texttt{dom}(A_1)$ and $\texttt{dom}(A_k)$. For each subset $S \in \mathcal{S}$, we add a value b_S to $\texttt{dom}(B)$ for every $B \in P'$. We set the domain of remaining attributes in $\texttt{attr}(Q) - P$ as $\{*\}$. For each relation $R_j \in \texttt{rels}(Q)$, we distinguish the following cases:

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

[4] Free sequence is a slight generalized notion of free path [5] studied in the literature, which further requires that for any relation $R_j \in \texttt{rels}(Q)$, either $\texttt{attr}(R_j) \cap P = \emptyset$, or $|\texttt{attr}(R_j) \cap P| = 1$, or $\texttt{attr}(R_j) \cap P = \{A_i, A_{i+1}\}$ for some $i \in [k-1]$.

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 $\texttt{SWP}(Q, D)$ is integral if $R'_j = \left(\pi_{\texttt{attr}(R_j)\cap P'} R'_j\right) \times (\pi_{A_k} R_j)$ holds for every relation R_j with $A_k \in \texttt{attr}(R_j)$ and $\texttt{attr}(R_j) \cap P' \neq \emptyset$. Applying a similar argument as Lemma 21, we can show that there always exists an integral solution to $\texttt{SWP}(Q, D)$. Below, it suffices to focus on integral solutions. It can be easily proved that $(\mathcal{U}, \mathcal{S})$ has a cover of size $\le k$ if and only if $\texttt{SWP}(Q, D)$ has an integral solution of size $\le nq_1 + knq_2 + q_3$ where $q_1, q_2, q_3 \le |\texttt{rels}(Q)|$ are query-dependent parameters. If $\texttt{SWP}$ 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 $\texttt{P} = \texttt{NP}$. ◀

Connection with Non-Free-connex CQs. We point out that every acyclic but non-free-connex CQ has a free sequence [5], hence does not have head-domination property. Together with Lemma 19, our characterization of $\texttt{SWP}$ for acyclic CQs coincides with the separation between free-connex and non-free-connex CQs. In short, $\texttt{SWP}$ is poly-time solvable or $O(1)$-approximable for free-connex CQs, while no poly-time algorithm can approximate $\texttt{SWP}$ for any acyclic but non-free-connex CQs within a factor of $(1 - o(1)) \cdot \log N$, unless $\texttt{P} = \texttt{NP}$.

4.2.2 Nested Clique

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

▶ **Lemma 25.** *There is no poly-time algorithm to approximate $\texttt{SWP}$ for $Q_{\textsf{pyramid}}$ within a factor of $(1 - o(1)) \cdot \log N$, unless $\texttt{P} = \texttt{NP}$.*

Proof. Consider an instance $(\mathcal{U}, \mathcal{S})$ of set cover, with $\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$ and $\mathcal{S} = \{S_1, S_2, \cdots, S_m\}$, where $m = n^c$ for some constant $c > 1$. We construct a database D for Q as follows. Let $\texttt{dom}(A) = \{a_1, a_2, \cdots, a_n\}$, $\texttt{dom}(B) = \{b_1, b_2, \cdots, b_n\}$, $\texttt{dom}(C) = \{c_1, c_2, \cdots, c_n\}$ and $\texttt{dom}(F) = \texttt{dom}(F^-) \times \texttt{dom}(F^+)$, where $\texttt{dom}(F^-) = \{f_1^-, f_2^-, \cdots, f_m^-\}$ and $\texttt{dom}(F^+) = \{f_1^+, f_2^+, \cdots, f_n^+\}$. 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 $\{(a_\ell, f_j^-)\} \times \texttt{dom}(F^+)$ to R_4. For each $i \in [n]$, we add tuples $\{b_i\} \times \texttt{dom}(F^-) \times \{f_i^+\}$ 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 $\texttt{SWP}(Q, D)$ includes all tuples in R_1, R_2, R_3. Below, we focus on R_4, R_5, R_6.

We observe that D enjoys highly symmetric structure over $\texttt{dom}(B)$. More specifically, each value $b_i \in \texttt{dom}(B)$ induces a subquery $Q_i(A, C) = R_4(A, F_i) \bowtie R_5(F_i) \bowtie R_6(C, F_i)$, where $\texttt{dom}(F_i) = \texttt{dom}(F^-) \times \{f_i^+\}$, and a sub-database $D^i = \left\{R_4^i, R_5^i, R_6^i\right\}$, where $R_4^i = \{(a_\ell, f_j^-, f_i^+) : \forall \ell \in [n], j \in [m], u_\ell \in S_j\}$, $R_5^i = \texttt{dom}(F_i)$ and $R_6^i = \texttt{dom}(C) \times \texttt{dom}(F_i)$. It can be easily checked that $\texttt{SWP}(Q, D) = R_1 \uplus R_2 \uplus R_3 \uplus \left(\uplus_{i\in[n]} \texttt{SWP}(Q_i, D_i)\right)$. For any $i \in [n]$, computing $\texttt{SWP}(Q_i, D_i)$ is almost the same as $Q_{\textsf{matrix}}$. Moreover, the solution to each $\texttt{SWP}(Q_i, D_i)$ shares the same structure, which is independent of the specific value $b_i \in \texttt{dom}(B)$. In a sub-database $D' \subseteq D$, let R'_i be the corresponding sub-relation of R_i. A solution D' to $\texttt{SWP}(Q, D)$ is *integral* if $R'_4 = (\pi_{A,F^-} R'_4) \times \texttt{dom}(F^+)$, $R'_5 = \texttt{dom}(B) \times (\pi_{F^-} R'_5) \times \texttt{dom}(F^+)$, and $R'_6 = \texttt{dom}(C) \times (\pi_{F^-} R'_6) \times \texttt{dom}(F^+)$. Implied by Lemma 21 and analysis above, there always exists an integral solution to $\texttt{SWP}(Q, D)$.

Moreover, $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if $\texttt{SWP}(Q, D)$ has an integral witness of size $\leq 3n^2 + n(n + k + kn) = (k + 4)n^2 + kn$. If $\texttt{SWP}$ 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 $\texttt{P} = \texttt{NP}$. ◀

Now, we are ready to introduce the structure of *nested clique* and the *rename* procedure for capturing the hardness of cyclic CQs:

▶ **Definition 26** (Nested Clique). *In a CQ Q, a nested clique is a subset of attributes $P \subseteq \texttt{attr}(Q)$ such that*
- *for any pair of attributes $A, B \in P$, there is some $R_j \in \texttt{rels}(Q)$ with $A, B \in \texttt{attr}(R_j)$;*
- *$P \cap \texttt{head}(Q) \neq \emptyset$ and $P - \texttt{head}(Q) \neq \emptyset$;*
- *there is no relation $R_j \in \texttt{rels}(Q)$ with $P \cap \texttt{head}(Q) \subseteq \texttt{head}(R_j)$.*

▶ **Definition 27** (Rename). *Given the nonout-connectivity graph H_Q of a CQ Q with connected components $H_1, H_2, \cdots, H_k$, the rename procedure assigns one distinct attribute to all attributes in the same component. The resulted CQ Q' contains the same output attributes as Q, and each $R_i \in \texttt{rels}(Q)$ defines a new relation $R'_i \in \texttt{rels}(Q')$ with $\texttt{attr}(R'_i) = \texttt{head}(R_i) \cup \{F_j : \forall j \in [k], H_j \cap \texttt{attr}(R_i) \neq \emptyset\}$.*

In Figure 3, Q_1 is renamed as $Q'_1(A_1, A_2, A_3) : -R_1(A_1, F_1), R_2(F_1), R_3(A_2, F_1), R_4(A_2, A_3, F_2), R_5(A_1, A_2)$.

▶ **Theorem 28.** *For a CQ Q, if its renamed query Q' contains a nested clique, there is no poly-time algorithm that can approximate $\texttt{SWP}$ for Q within a factor of $(1 - o(1)) \cdot \log N$, unless $\texttt{P} = \texttt{NP}$.*

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

Consider an arbitrary instance of set cover $(\mathcal{U}, \mathcal{S})$ with $\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$ and $\mathcal{S} = \{S_1, S_2, \cdots, S_m\}$, where $m = n^c$ for some constant $c > 1$. We construct a database D for Q. Let $\texttt{dom}(A) = \{a_1, a_2, \cdots, a_n\}$, $\texttt{dom}(B) = \{b_1, b_2, \cdots, b_n\}$, $\texttt{dom}(C) = \{c_1, c_2, \cdots, c_n\}$ for every $C \in P'$, and $\texttt{dom}(F) = \texttt{dom}(F^-) \times \texttt{dom}(F^+)$ where $\texttt{dom}(F^-) = \{f^-_1, f^-_2, \cdots, f^-_m\}$ and $\texttt{dom}(F^+) = \{f^+_{h_1,h_2,\cdots,h_\ell} : \forall h_1, h_2, \cdots, h_\ell \in [n]\}$. We distinguish the following cases:
- If $\texttt{attr}(R_i) \cap P \subseteq \texttt{head}(Q)$, R_i is a Cartesian product over all attributes in $\texttt{head}(R_i) \cap P$;
- Otherwise, we further distinguish the following three cases:
  - R_2 is a Cartesian product over all attributes in $\texttt{attr}(R_2) \cap P$;
  - If $A, F \in \texttt{attr}(R_i)$, 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 $\left\{a_\ell, f^-_j\right\} \times \texttt{dom}(F^+)$ to $(\pi_{A,F} R_i)$. If $\texttt{attr}(R_i) \cap P' \neq \emptyset$, for each tuple $\left(a_\ell, f^+_{h_1,h_2,\cdots,h_\ell}, f^-_j\right) \in (\pi_{A,F} R_i)$, we extend it by attaching value c_{h_j} for attribute $A_j \in \texttt{attr}(R_i) \cap P'$. This way, we already obtain $\left(\pi_{A,F,\texttt{attr}(R_i) \cap P'} R_i\right)$. At last, we construct R_i as the Cartesian product of remaining attributes in $\texttt{head}(R_i) \cap \texttt{head}(R_2) \cap P$ and $\left(\pi_{A,F,\texttt{attr}(R_i) \cap P'} R_i\right)$.

- Otherwise, $F \in \texttt{attr}(R_i)$ but $A \notin \texttt{attr}(R_i)$. We construct sub-relation $(\pi_F R_i) = \texttt{dom}(F)$. If $\texttt{attr}(R_i) \cap P' \neq \emptyset$, for each tuple $\left(f^+_{h_1,h_2,\cdots,h_\ell}, f^-_j\right) \in \pi_F R_i$, we extend it by attaching value c_{h_j} for attribute $A_j \in \texttt{attr}(R_i) \cap P'$. This way, we already obtain $\left(\pi_{F,\texttt{attr}(R_i)\cap P'} R_i\right)$. At last, we construct R_i as the Cartesian product of remaining attributes in $\texttt{head}(R_i) \cap \texttt{head}(R_2) \cap P$ and $\left(\pi_{A,F,\texttt{attr}(R_i)\cap P'} R_i\right)$.

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

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

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

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

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

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

It can be easily shown that there always exists an integral solution to $\texttt{SWP}(Q, D)$. Moreover, $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if $\texttt{SWP}(Q, D)$ has an integral solution of size $(q_1 k + q_2) \cdot n^{|\texttt{head}(Q)\cap P|-1}$, where $q_1, q_2 \leq |\texttt{rels}(Q)|$ are some query-dependent parameters. Applying a similar argument as Lemma 21, we can show that if $\texttt{SWP}$ is $(1 - o(1)) \cdot \log N$-approximable 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 $\texttt{P} = \texttt{NP}$. ◀

4.3 Completeness

At last, we complete the proof of Theorem 16 by establishing the connection between the non-existence of hardcore structures and head-domination property:

▶ **Lemma 29.** *In a CQ Q, if there is neither a free sequence nor a nested clique in its renamed query, Q has head-domination property.*

We have proved Lemma 29 for acyclic CQs, such that if Q does not contain a free sequence, Q has head-cluster property. It suffices to focus on cyclic CQs. We note that any cyclic CQ contains a *cycle* or *non-conformal clique* [8]. Our proof is based on a technical lemma that if every cycle or non-conformal clique contains only output attributes or only non-output attributes, Q has head-domination property. We complete it by showing that if Q does not contain a free sequence or nested clique in its renamed query, no cycle or non-conformal clique contains both output and non-output attributes.

5 Approximation Algorithms

We next explore possible approximation algorithms for CQs without head-domination property. All missing proofs are in the full version [28].

5.1 Baseline

A baseline for general CQs returns the union of smallest witness for every query result, which includes at most $\min\{N, |\texttt{rels}(Q)| \cdot |Q(D)|\}$ tuples. Meanwhile, AGM bound [4] implies that at least $O(|Q(D)|^{1/\rho^*})$ tuples are needed to reproduce $|Q(D)|$ results, where ρ^* is the fractional edge covering number of Q. Together, we obtain:

▶ **Theorem 30.** *SWP is N^{1-1/ρ^*}-approximable for any CQ Q, where ρ^* is the fractional edge covering number of Q.*

This upper bound is polynomially larger than the logarithmic lower bound proved in Section 4. We next explore better approximation algorithms for some commonly-used CQs.

5.2 Star CQs

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

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

Our approximation algorithm follows the greedy strategy developed for 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 as well as 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, it is critical to keep the size of $\mathcal{S}$ small. Let's start with Q_{matrix} ($m = 2$).

Greedy Algorithm for Q_{matrix}. Given Q_{matrix}, a database D, and a subset of query results $\mathcal{C} \subseteq Q_{\mathsf{matrix}}(D)$, the price of a collections of tuples (X, Y) for $X \subseteq R_1$ and $Y \subseteq R_2$ is defined as $f(\mathcal{C}, X, Y) = \frac{|X|+|Y|}{|\pi_{A,C}(X \bowtie Y) - \mathcal{C}|}$. 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 31.

▶ **Lemma 31.** *Given Q_{matrix} and a database D, for two distinct values $b, b' \in \texttt{dom}(B)$, 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 arbitrary $\mathcal{C} \subseteq Q_{\mathsf{matrix}}(D)$, $\min\{f(\mathcal{C}, X_1, Y_1), f(\mathcal{C}, X_2, Y_2)\} \leq f(\mathcal{C}, X_1 \cup X_2, Y_1 \cup Y_2)$.*

Now, we are ready to present a greedy algorithm that runs in polynomial time. As shown in Algorithm 2, we always maintain a pair (X_i, Y_i) for every value $b_i \in \texttt{dom}(B)$, such that (X_i, Y_i) minimize the function $f(\mathcal{C}, X, Y)$ 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 minimum price, pick all related tuples in this pair, and update the coverage $\mathcal{C}$. After this step, we also need to update the candidate pair (X_i, Y_i) for each value $b_i \in \texttt{dom}(B)$ 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 *densest subgraph in the bipartite graph* in the literature [25, 29], for which a poly-time algorithm based on max-flow has been proposed.

▶ **Definition 32** (Densest Subgraph in Bipartite Graph). *Given a bipartite graph (X, Y, E) with $E : X \times Y \to \{0, 1\}$, it asks to find $X' \subseteq X, Y' \subseteq Y$ to maximize $\frac{\sum_{x \in X', y \in Y'} E(x,y)}{|X'|+|Y'|}$.*

Algorithm 2 GreedySWP(Q_{matrix}, D).

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

Back to our problem, each value $b_i \in \mathtt{dom}(B)$ induces a bipartite graph with $X = \sigma_{B=b_i} R_1$, $Y = \sigma_{B=b_i} R_2$, and $E = \{(x, y) \mid x \in X, y \in Y, \pi_{A,C}(x \bowtie y) \notin \mathcal{C}\}$. 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 [39]. We next focus on the time complexity. The while-loop proceeds in at most $O(|Q(D)|)$ iterations, since $|\mathcal{C}|$ increases by at least 1 in every iteration. It takes $O(N)$ time to find the pair with minimum price, since there are $O(N)$ distinct values in $\mathtt{dom}(B)$. Moreover, it takes polynomial time to update (X_i, Y_i) for each $b_i \in \mathtt{dom}(B)$. Overall, Algorithm 2 runs in polynomial time in terms of N.

Extensions. Our algorithm for Q_{matrix} can be extended to star CQs, and further to all CQs with only one non-output attribute. Similar property as Lemma 31 also holds. Our greedy algorithm needs a generalized primitive, noted as *densest subgraph in hypergraph*[5] [27] for finding the "set" with the smallest price. Following the similar analysis, we obtain:

▶ **Theorem 33.** *For any CQ Q with $|\mathtt{attr}(Q) - \mathtt{head}(Q)| = 1$, SWP is $O(\log N)$-approximable.*

5.3 Line CQs

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

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

with $m \geq 3$. We surprisingly find that SWP for line CQs is closely related to *directed Steiner forest* (DSF) problem [18, 11, 21, 16] in the network design: Given an edge-weighed 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 SWP(Q_{line}, D) is a special case of DSF. This way, all existing algorithm proposed for DSF can be applied to SWP for Q_{line}. The best approximation ratio achieved is $O(\min\{k^{\frac{1}{2}+o(1)}, n^{0.5778}\})$ [21, 11, 1]. Combining the baseline in Section 5.1 (with $\rho^* = 2$ for line CQs) and existing algorithms for DSF, we obtain:

▶ **Theorem 34.** *For Q_{line} and any database D of input size N, there is a poly-time algorithm that can approximate SWP(Q_{line}, D) within a factor of $O(\min\{|Q_{line}(D)|^{\frac{1}{2}+o(1)}, \mathtt{dom}^{0.5778}, N^{\frac{1}{2}}\})$, where $\mathtt{dom}$ is the number of values that participates in at least one full join result.*

[5] Given 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.

There is a large body of works investigating the lower bounds of DSF; and we refer interested readers to [16] 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 [18, 13]) and prove the following:

▶ **Theorem 35.** *No poly-time algorithm approximates* `SWP` *for* Q_{line} *within* $\Omega(2^{(\log N)^{1-\epsilon}})$ *factor for any constant* $\epsilon > 0$, *unless* P = NP.

6 Related Work

Factorized database [34]. Factorized database studies a nested representations of query results that can be exponentially more succinct than flat query result, which has the same goal as `SWP`. They built a tree-based representations by exploiting the distributivity of product over union and commutativity of 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 witness.

Data Synopses [14, 36]. 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 etc. 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 etc. `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 queries results are correct. This notion of why-provenance has been extensively investigated in the literature [9, 26, 3], but not from the perspective of minimizing the size of witness. The `SWP` problem is also related to the resilience problem [22, 23, 37], 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 clear connection here, but their solutions do not imply anything to each other.

7 Conclusion

In this paper, 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 head-domination property. For remaining CQs, both upper and lower bounds remain to be improved, which may lead to fundamental breakthrough for other related problems, such as DSF.

2. *SWP for CQs with self-joins.* It becomes much more challenging when self-joins exists, as one tuple appears in multiple logical copies of input relation. Similar observation has been made for the related resilience problem [23, 31]
3. *Relaxing the number of query results witnessed.* It is possible to explore approximation on the number of query results that can be witnessed. Here, SWP is related to the *partial set cover* problem, for which many approximation algorithms [24] have been studied.

References

1 Amir Abboud and Greg Bodwin. Reachability preservers: New extremal bounds and approximation algorithms. In *SODA*, pages 1865–1883. SIAM, 2018. doi:10.1137/1.9781611975031.122.

2 Mahmoud Abo Khamis, Hung Q Ngo, and Atri Rudra. Faq: questions asked frequently. In *PODS*, pages 13–28, 2016. doi:10.1145/2902251.2902280.

3 Yael Amsterdamer, Daniel Deutch, and Val Tannen. Provenance for aggregate queries. In *PODS*, pages 153–164, 2011. doi:10.1145/1989284.1989302.

4 Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. In *FOCS*, FOCS '08, pages 739–748, 2008. doi:10.1109/FOCS.2008.43.

5 Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. On acyclic conjunctive queries and constant delay enumeration. In *CSL*, pages 208–222. Springer, 2007. doi:10.1007/978-3-540-74915-8_18.

6 C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *JACM*, 30(3):479–513, 1983. doi:10.1145/2402.322389.

7 KORTE Bernhard and JENS Vygen. Combinatorial optimization: Theory and algorithms. *Springer, Third Edition, 2005.*, 2008.

8 Johann Brault-Baron. Hypergraph acyclicity revisited. *CSUR*, 49(3):1–26, 2016. doi:10.1145/2983573.

9 Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330. Springer, 2001. doi:10.1007/3-540-44503-X_20.

10 Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. On random sampling over joins. *ACM SIGMOD Record*, 28(2):263–274, 1999. doi:10.1145/304182.304206.

11 Chandra Chekuri, Guy Even, Anupam Gupta, and Danny Segev. Set connectivity problems in undirected graphs and the directed steiner network problem. *TALG*, 7(2):1–17, 2011. doi:10.1145/1921659.1921664.

12 Yu Chen and Ke Yi. Random sampling and size estimation over cyclic joins. In *ICDT*, 2020. doi:10.4230/LIPIcs.ICDT.2020.7.

13 Rajesh Chitnis, Andreas Emil Feldmann, and Pasin Manurangsi. Parameterized approximation algorithms for bidirected steiner network problems. *ACM Trans. Algorithms*, 17(2), apr 2021. doi:10.1145/3447584.

14 Graham Cormode, Minos Garofalakis, Peter J Haas, Chris Jermaine, et al. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends® in Databases*, 4(1–3):1–294, 2011. doi:10.1561/1900000004.

15 Graham Cormode and Ke Yi. *Small summaries for big data*. Cambridge University Press, 2020.

16 Irit Dinur and Pasin Manurangsi. Eth-hardness of approximating 2-csps and directed steiner network. In Anna R. Karlin, editor, *ITCS*, volume 94 of *LIPIcs*, pages 36:1–36:20, 2018. doi:10.4230/LIPICS.ITCS.2018.36.

17 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *STOC*, pages 624–633, 2014. doi:10.1145/2591796.2591884.

18 Yevgeniy Dodis and Sanjeev Khanna. Design networks with bounded pairwise distance. In *STOC*, pages 750–759, 1999. doi:10.1145/301250.301447.

19 R. Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *JACM*, 30(3):514–550, 1983. doi:10.1145/2402.322390.

20 Uriel Feige. A threshold of ln n for approximating set cover. *JACM*, 45(4):634–652, 1998.

21 Moran Feldman, Guy Kortsarz, and Zeev Nutov. Improved approximation algorithms for directed steiner forest. *JCSS*, 78(1):279–292, 2012. doi:10.1016/j.jcss.2011.05.009.

22 Cibele Freire, Wolfgang Gatterbauer, Neil Immerman, and Alexandra Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries. *PVLDB*, 9(3):180–191, 2015. doi:10.14778/2850583.2850592.

23 Cibele Freire, Wolfgang Gatterbauer, Neil Immerman, and Alexandra Meliou. New results for the complexity of resilience for binary conjunctive queries with self-joins. In *PODS*, pages 271–284, 2020. doi:10.1145/3375395.3387647.

24 Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53(1):55–84, 2004. doi:10.1016/j.jalgor.2004.04.002.

25 Andrew V Goldberg. Finding a maximum density subgraph. Technical report, University of California Berkeley, 1984.

26 Todd J Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007. doi:10.1145/1265530.1265535.

27 Shuguang Hu, Xiaowei Wu, and TH Hubert Chan. Maintaining densest subsets efficiently in evolving hypergraphs. In *CIKM*, pages 929–938, 2017. doi:10.1145/3132847.3132907.

28 Xiao Hu and Stavros Sintos. Finding smallest witnesses for conjunctive queries. *arXiv preprint*, 2023. doi:10.48550/arXiv.2311.18157.

29 Samir Khuller and Barna Saha. On finding dense subgraphs. In *ICALP*, pages 597–608. Springer, 2009. doi:10.1007/978-3-642-02927-1_50.

30 Benny Kimelfeld, Jan Vondrák, and Ryan Williams. Maximizing conjunctive views in deletion propagation. In *PODS*, pages 187–198, 2011. doi:10.1145/1989284.1989308.

31 Benny Kimelfeld, Jan Vondrák, and Ryan Williams. Maximizing conjunctive views in deletion propagation. *TODS*, 37(4):1–37, 2012. doi:10.1145/2389241.2389243.

32 Zhengjie Miao, Sudeepa Roy, and Jun Yang. Explaining wrong queries using small examples. In *SIGMOD*, pages 503–520, 2019. doi:10.1145/3299869.3319866.

33 Dana Moshkovitz. The projection games conjecture and the np-hardness of ln n-approximating set-cover. In *APPROX-RANDOM*, pages 276–287. Springer, 2012. doi:10.1007/978-3-642-32512-0_24.

34 Dan Olteanu and Maximilian Schleich. Factorized databases. *ACM SIGMOD Record*, 45(2):5–16, 2016. doi:10.1145/3003665.3003667.

35 John Paparrizos, Chunwei Liu, Bruno Barbarioli, Johnny Hwang, Ikraduya Edian, Aaron J Elmore, Michael J Franklin, and Sanjay Krishnan. Vergedb: A database for iot analytics on edge devices. In *CIDR*, 2021. URL: http://cidrdb.org/cidr2021/papers/cidr2021_paper11.pdf.

36 Jeff M Phillips. Coresets and sketches. In *Handbook of discrete and computational geometry*, pages 1269–1288. Chapman and Hall/CRC, 2017.

37 Biao Qin, Deying Li, and Chunlai Zhou. The resilience of conjunctive queries with inequalities. *Information Sciences*, 613:982–1002, 2022. doi:10.1016/j.ins.2022.08.049.

38 Moshe Y Vardi. The complexity of relational query languages. In *STOC*, pages 137–146, 1982. doi:10.1145/800070.802186.

39 Vijay V Vazirani. *Approximation algorithms*, volume 1. Springer, 2001.

40 Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, volume 81, pages 82–94, 1981.

41 Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. Random sampling over joins revisited. In *SIGMOD*, pages 1525–1539, 2018. doi:10.1145/3183713.3183739.