Smallest Synthetic Witnesses for Conjunctive Queries*

ARYAN ESMAILPOUR, Department of Computer Science, University of Illinois Chicago, USA
BORIS GLAVIC, Department of Computer Science, University of Illinois Chicago, USA
XIAO HU, Cheriton School of Computer Science, University of Waterloo, Canada
STAVROS SINTOS, Department of Computer Science, University of Illinois Chicago, USA

Given a self-join-free conjunctive query Q and a set of tuples S, a *synthetic witness* D is a database instance such that the result of Q on D is S. In this work, we are interested in two problems. First, the existence problem ESW decides whether any synthetic witness D exists. Second, given that a synthetic witness exists, the minimization problem SSW computes a synthetic witness of minimal size. The SSW problem is related to the *smallest witness problem* recently studied by Hu and Sintos [22]; however, the objective and the results are inherently different. More specifically, we show that SSW is poly-time solvable for a wider range of queries. Interestingly, in some cases, SSW is related to optimization problems in other domains, such as the *role mining* problem in data mining and the *edge concentration* problem in graph drawing. Solutions to ESW and SSW are of practical interest, e.g., for *test database generation* for applications accessing a database and for *data compression* by encoding a dataset S as a pair of a query Q and database D.

We prove that ESW is in P, presenting a simple algorithm that, given any S, decides whether a synthetic witness exists in polynomial time in the size of S. Next, we focus on the SSW problem. We show an algorithm that computes a minimal synthetic witness in polynomial time with respect to the size of S for any query Q that has the *head-domination* property. If Q does not have such a property, then SSW is generally hard. More specifically, we show that for the class of *path queries* (of any constant length), SSW cannot be solved in polynomial time unless P = NP. We then extend this hardness result to the class of *Berge-acyclic* queries that do not have the head-domination property, obtaining a full dichotomy of SSW for Berge-acyclic queries. Finally, we investigate the hardness of SSW beyond Berge-acyclic queries by showing that SSW cannot be solved in polynomial time for some cyclic queries unless P = NP.

CCS Concepts: • **Theory of computation → Data provenance**; **Database query processing and optimization (theory)**.

Additional Key Words and Phrases: conjunctive queries, synthetic witness, head-domination, Berge-acyclic

1 Introduction

In the era of big data, data summarization techniques have been developed to reduce the computational cost and space usage for approximate and exact query processing. Sampling [9, 10, 46], sketches [14], and coresets [38] are examples of data summarization techniques for approximation. The notion of *witness* has been proposed in databases as a notion of exact data summarization. Witnesses have been utilized as a form of why-provenance [1, 7, 20] providing proof for the existence of query results, with wide applications in explainable data-intensive analytics. Given a SJFCQ[1] Q, and a database $\bar{D}$, Buneman et al. [7] first introduced the *witness* for a query result $t \in Q(\bar{D})$ as a subset $D' \subseteq \bar{D}$ of tuples such that $t \in Q(D')$, where $Q(X)$ for a database X is the result of query Q on X. The smallest witness problem was first proposed by [35], where the goal was to compute a witness $D' \subseteq \bar{D}$ of minimal size. Recently, Hu and Sintos [22] redefined the notion of *witness* to be a subset $D' \subseteq \bar{D}$ of tuples such that $Q(\bar{D}) = Q(D')$. In this paper, we study the *smallest synthetic*

*This work was partially supported by NSF grants IIS-2420577, IIS-2420691, IIS-2348919, and NSERC Discovery Grant.
[1]We use SJFCQ to denote a self-join-free conjunctive query.

Authors' Contact Information: Aryan Esmailpour, Department of Computer Science, University of Illinois Chicago, Chicago, USA, aesmai2@uic.edu; Boris Glavic, Department of Computer Science, University of Illinois Chicago, Chicago, USA, bglavic@uic.edu; Xiao Hu, Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada, xiaohu@uwaterloo.ca; Stavros Sintos, Department of Computer Science, University of Illinois Chicago, Chicago, USA, stavros@uic.edu.

witness problem, the analog problem where no input database is given. Given a query Q and the results set S, our goal is to construct the database D of a minimal size such that $Q(D) = S$. We also investigate the problem of deciding the existence of such a database: given Q and S, does a database D exist such that $Q(D) = S$? We present a simple algorithm that can decide whether such a witness exists in polynomial time in the size of S.

The problem we study in this work arises naturally when testing applications that access a database. To ensure certain control flow paths are taken in the application, queries executed by the application must return certain results. One way to ensure this is to generate a test database such that evaluating the query on the test database returns the desired result [5, 42]. Minimizing the size of the test database is beneficial as it reduces the storage cost and runtime of evaluating the tests. Another application is to reduce communication costs when sending data between two servers. Instead of sending a dataset S directly, we can "compress" S by encoding it as a pair (Q, D) such that $Q(D) = S$ and $|Q| + |D| \ll |S|$. While the techniques from [22] can also be used for this purpose, we observe that the restriction to $D' \subseteq \bar{D}$ is unnecessary in this context (and might not achieve high compression) as the server we are sending S to does not care whether the database we send is synthetic or not.

Example 1.1. Consider the matrix query $Q_{\mathsf{matrix}}(A_1, A_3) :- R_1(A_1, A_2) \bowtie R_2(A_2, A_3)$ and a database $\bar{D}$ introduced in the following. For some perfect square number $N \in \mathbb{N}$, let $\mathcal{I} = \{1, 2, \ldots, \sqrt{N}\}$ and $\mathcal{J} = \mathcal{I} \times \mathcal{I} = \{(1,1), \ldots, (1,\sqrt{N}), (2,1), \ldots, (2,\sqrt{N}), \ldots, (\sqrt{N},1), \ldots, (\sqrt{N},\sqrt{N})\}$. Consider an arbitrary order of the tuples in $\mathcal{J}$. For the i-th tuple $(j_1, j_2) \in \mathcal{J}$, we have in $\bar{D}$ the tuple $(j_1, i) \in R_1$ and the tuple $(i, j_2) \in R_2$. Moreover, $Q_{\mathsf{matrix}}(\bar{D}) = \mathcal{J}$. Consider sending the data set $Q_{\mathsf{matrix}}(\bar{D})$ between two servers. The smallest witness $D' \subseteq \bar{D}$ according to [22] is $D' = \bar{D}$ with size $|D'| = 2N$. However, the smallest synthetic witness D includes tuples $R_1 = \{(i, *) \mid i \in \mathcal{I}\}$ and $R_2 = \{(*, i) \mid i \in \mathcal{I}\}$, where $*$ is a special value, with $|D| = 2\sqrt{N}$. In the first case, the server needs to send $2N$ tuples, while in the second case, only $2\sqrt{N}$ tuples.

1.1 Problem Definition

Let $\mathbf{R}$ be a database schema that contains m relations $R_1, R_2, \ldots, R_m$ over a set of attributes $\mathbb{A}$. Each relation R_i is defined on a subset of attributes $\mathbb{A}_i \subseteq \mathbb{A}$. Let $\mathsf{dom}(A)$ be the domain of attribute $A \in \mathbb{A}$. Let $\mathsf{dom}(X) = \prod_{A \in X} \mathsf{dom}(A)$ be the domain of a set of attributes $X \subseteq \mathbb{A}$. Given the database schema $\mathbf{R}$, let D be a given database instance of $\mathbf{R}$, and let the corresponding relations of $R_1, R_2, \ldots, R_m$ be $R_1^D, R_2^D, \ldots, R_m^D$, where R_i^D is a collection of tuples defined on $\mathsf{dom}(\mathbb{A}_i)$. The size of a database instance D is denoted as $|D| = \sum_{i \in [m]} |R_i^D|$. We will drop the superscript where D is clear from the context. For any attribute $A \in \mathbb{A}$, $\pi_A t$ denotes the value over attribute A of tuple t. Similarly, for a set of attributes $X \subseteq \mathbb{A}$, $\pi_X t$ denotes values over attributes in X of tuple t.

We consider the class of conjunctive queries without self-joins:[2]

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

where $\mathbf{A} \subseteq \mathbb{A}$ is a set of output (head) attributes and $\mathbb{A} - \mathbf{A}$ is the set of non-output (body) attributes. A SJFCQ is *full* if $\mathbf{A} = \mathbb{A}$, indicating the natural join of the given relations; otherwise, it is *non-full*. We call $Q_{\mathtt{full}}(\mathbb{A}) = R_1(\mathbb{A}_1) \bowtie R_2(\mathbb{A}_2) \bowtie \ldots \bowtie R_m(\mathbb{A}_m)$ as the underlying full join query. We call Q *self-join-free* if each R_i in Q is distinct. The query result of a SJFCQ Q on database D, denoted as $Q(D)$, is the projection of the natural join result of $Q_{\mathtt{full}}$ onto $\mathbf{A}$, i.e.,

$$Q(D) = \{t' \in \mathsf{dom}(\mathbf{A}) \mid \exists t \in \mathsf{dom}(\mathbb{A}) \text{ such that } \pi_{\mathbf{A}}(t) = t' \text{ and } \forall i \in [m] : \pi_{\mathbb{A}_i}(t) \in R_i\}$$

[2]This is the definition of join-project queries, however, we call them conjunctive queries to be consistent with [22].

A tuple $t \in R_i$ is *dangling* if it does not participate in any join, i.e. if there does not exist any tuple $t' \in Q_{\mathtt{full}}(D)$ such that $\pi_{\mathtt{attr}(R_i)}t' = t$, and *non-dangling* otherwise. We use $\mathtt{rels}(Q)$ to denote all the relations that appear in the body of Q, and use $\mathtt{head}(Q)$ to denote all the attributes that appear in the head of Q, i.e., $\mathtt{head}(Q) = \mathbf{A}$. We also use $\mathtt{attr}(R_i) = \mathbb{A}_i$ to denote all the attributes that appear in R_i. Moreover, we use $\mathtt{head}(R_i)$ to denote $\mathtt{head}(Q) \cap \mathtt{attr}(R_i)$, i.e., the subset of head attributes that appear in R_i.

For a set $S \subseteq \mathtt{dom}(\mathbf{A})$ of tuples, a database D is called a *synthetic witness* of (Q, S) if $Q(D) = S$. For every attribute $A \in \mathbb{A} - \mathtt{head}(Q)$, we assume that $\mathtt{dom}(A)$ is an infinite set (for example, the set of real numbers $\mathbb{R}$). For simplicity, for every attribute $A \in \mathtt{head}(Q)$, we also assume that $\mathtt{dom}(A)$ is an infinite set. Even if the finite set of tuples in S contains values from a different domain, they can be represented as items from the infinite set $\mathtt{dom}(A)$.

Definition 1.2 (Existence of Synthetic Witness (ESW)). For a SJFCQ Q and a set of tuples $S \subseteq \mathtt{dom}(\mathtt{head}(Q))$, decide whether there exists a synthetic witness for (Q, S).

Given Q and S, we denote the above problem as $\mathtt{ESW}(Q, S)$.

Definition 1.3 (Smallest Synthetic Witnesses (SSW)). For a SJFCQ Q and a set of tuples $S \subseteq \mathtt{dom}(\mathtt{head}(Q))$ such that $\mathtt{ESW}(Q, S)$ returns **true**, return a synthetic witness D for (Q, S) such that there exists no other synthetic witness D' for (Q, S) with $|D'| < |D|$.

Given Q and S, we denote the above problem as $\mathtt{SSW}(Q, S)$. We note that the solution to $\mathtt{SSW}(Q, S)$ may not be unique; hence, we aim to find some minimum synthetic witness. See Appendix A for an example with multiple solutions. In this work, we study the data complexity of ESW and SSW, i.e., the size of the database schema and query is considered to be constant. As we will show, for any SJFCQ Q and any set $S \subseteq \mathtt{dom}(\mathtt{head}(Q))$ of tuples, $\mathtt{ESW}(Q, S)$ is poly-time solvable in terms of $|S|$. However, SSW is hard for general SJFCQs. We say that SSW is *poly-time solvable* for Q, if for an arbitrary set of tuples $S \subseteq \mathtt{dom}(\mathtt{head}(Q))$, $\mathtt{SSW}(Q, S)$ can be computed in polynomial time in terms of $|S|$. As SSW is not always poly-time solvable, we introduce an approximated version:

Definition 1.4 (θ-Approximated Smallest Synthetic Witnesses (θ-SSW)). For a SJFCQ Q and a set of tuples $S \subseteq \mathtt{dom}(\mathtt{head}(Q))$ with $\mathtt{ESW}(Q, S) =$ **true**, return a synthetic witness D for (Q, S) such that $|D| \leq \theta \cdot |D^*|$, where D^* is a solution to $\mathtt{SSW}(Q, S)$, and $\theta > 1$.

1.2 Our Results

Our main results achieved in this paper are summarized as follows:

- In Section 3, we show a poly-time algorithm for ESW on arbitrary SJFCQs.
- In Section 4, we show a poly-time algorithm for SSW on SJFCQs with the *head-domination* property (formally defined in Section 2).
- In Section 5, we investigate the hardness of SSW on SJFCQs without the head-domination property. We first show that SSW is not poly-time solvable for *path queries* by a reduction from the *vertex cover* problem. Next, we extend the hardness result to arbitrary Berge-acyclic SJFCQs without the head-domination property. This leads to a full dichotomy of SSW for the class of Berge-acyclic SJFCQs: For a Berge-acyclic SJFCQ Q, if it has the head-domination property, SSW is poly-time solvable for Q; otherwise, SSW is not poly-time solvable for Q, unless P = NP. In Section 5.4, we show the hardness of SSW for some SJFCQs which are not Berge-acyclic.
- In Section 6, we show that a minor modification of our algorithm for the ESW problem returns an $O\left(\min\left\{|S|^{1-\frac{1}{\rho^\star(Q)}}, \frac{|S|}{\beta}\right\}\right)$-approximated solution to SSW for any SJFCQ Q, where $\rho^\star(Q)$ is the fractional edge covering number of Q and $\beta = \max_{R_i \in \mathtt{rels}(Q)} |\pi_{\mathtt{head}(R_i)}S|$.

1.3 Related Work

Smallest witnesses for SJFCQs. Given a SJFCQ Q and a database $\bar{D}$, Buneman et al. [7] first introduced the *witness* for a query result $t \in Q(\bar{D})$ as a subset $D' \subseteq \bar{D}$ of tuples such that $t \in Q(D')$. Witnesses can be determined in poly-time for SJFCQs. Hu and Sintos [22] defined a *witness* for a SJFCQ Q and database $\bar{D}$ to be a subset $D' \subseteq \bar{D}$ of tuples such that $Q(\bar{D}) = Q(D')$. The *smallest witness problem* (SWP) is then to find a witness for $(Q, \bar{D})$ of minimal size. While SSW is quite similar to SWP, there are a couple of fundamental differences. First, for SSW, witnesses are not constrained to subsets of the input database $\bar{D}$. To compare SSW and SWP, set $S = Q(\bar{D})$ to be the input of SSW. While SWP returns a smallest witness as a subset of $\bar{D}$, SSW determines the smallest synthetic witness that is not necessarily a subset of $\bar{D}$. Their differences lead to different techniques and results. In [22], SWP is poly-time solvable if Q has the *head-cluster property*. Note that the head-cluster property implies head-domination, but the other direction does not hold. For head-dominated SJFCQs but without the head-cluster property, SWP is not poly-time solvable, but interestingly, SSW is poly-time solvable. Therefore, SSW is poly-time solvable for a greater class of queries (unless P = NP), and by definition, the size of the witness in SSW is no greater than the size of the witness in SWP. When we are not restricted to using tuples from the input database, it is always better to obtain a synthetic witness.

Resilience and view update. The problem of finding witnesses is also intimately related to query resilience [17, 18, 31, 39] and variants of the view update problem including deletion propagation [6, 11, 23, 27, 28], insertion propagation [34], and data-based explanations for missing answers [21, 45]. The decision version of the side-effect free view update problem is given Q, D, $V = Q(D)$, and ΔV, determine whether there exists ΔD such that $Q(D \uplus \Delta D) = V \uplus \Delta V$ where ΔD and ΔV may consist both of deletions and insertions and $\uplus$ denotes applying such a delta to a database or relation. The deletion (insertion) propagation problem is a restricted version of this problem where both ΔD and ΔV are deletions (insertions). The relevance of head-domination for deciding the complexity of deletion propagation was recognized in prior work [27]. ESW, the decision version of the synthetic witness problem is a special case of the insertion propagation problem [34] where $D = \emptyset$ and $V = \emptyset$. While ESW is poly-time solvable for all SJFCQs, the insertion propagation problem [34] is NP-hard in terms of data complexity if tuples from ΔD only use values from the active domains of attributes from D. Another problem related to deletion propagation is query resilience [17, 18, 39]: for a database D and query Q, the resilience of Q is the minimum number of tuples ΔD that have to be deleted from D such that $Q(D - \Delta D) = \emptyset$. Makhija and Gatterbauer [31, 33] showed a novel way to unify all the approaches related to deletion propagation (including the SWP problem in [22]).

While related papers on SWP, resilience, and deletion propagation problems, also provide hardness results for some classes of queries, their input consists of the database instance, so their approach is always to create a database instance that makes their problem hard. In our case, the set S is the input to our problem, i.e., we cannot directly define the database instance. Intuitively, they leverage both the structure of their constructed database instances and the query results at the same time, while in our case, we only have control over the results of a query. This makes it challenging to acquire the known techniques for showing hardness results and it is not clear how the approaches from the related papers can be applied to our setting. However, interestingly, the head-domination property appears to be a key in related problems [22], [27], and similarly in our setting.

Data synopses. Data synopses [13, 29, 38] are used in *approximate query processing* to approximate the query's result based on the information in the synopsis. A common technique is identifying a small subset of the input as a synopsis that captures important input properties in a preprocessing step. At query time, the synopsis can then compute a tight approximation of the query's result over the whole dataset. The main difference to our work is that (i) data synopses in databases and

computational geometry have been typically applied for range queries, aggregation queries, top-k, clustering, and other spatial queries instead of SJFCQs and (ii) our goal is to compute an exact result instead of an approximation.

Factorized databases. Factorized databases [32, 37] are nested representations of query results as relational algebra expressions (unions and cross products) that can be exponentially more succinct than flat query results by exploiting the distributivity of product over union and commutativity of product and union. A common problem studied in factorized databases is finding a variable ordering for a given query that leads to the smallest worst-case factorized representation. Kenig and Weinberger [26] investigate how to measure the loss introduced when decomposing a database schema into an acyclic join dependency while Kenig et al. [25] discover approximate acyclic database schemas and multivalued dependencies from relational data. In contrast, in SSW the goal is to minimize a database instance based on a given set of query results. An interesting direction for future work would be to investigate whether it is possible to further compress a synthetic witness through factorization.

SSW for the matrix query. The SSW problem for the *matrix query* $Q_{\mathsf{matrix}}(A_1, A_3) :- R_1(A_1, A_2) \bowtie R_2(A_2, A_3)$ is essentially the *weighted biclique covering* problem [41, 43]. A biclique edge cover C for a bipartite graph $G_B = (V, U, E)$ where $U \cup V$ are the nodes of the graphs and E is the set of edges, is a set of subgraphs of G_B such that (1) each subgraph $c \in C$ is a complete bipartite graph, and (2) every edge in E belongs to at least one subgraph in C. For a subgraph g of G_B, let $\mathcal{W}(g)$ be the number of vertices in g. In the weighted biclique covering problem, the goal is to find a biclique edge cover C such that $\sum_{c \in C} \mathcal{W}(c)$ is minimized. If every edge in G_B represents a tuple in S (and vice versa), the weighted biclique covering problem is equivalent to the SSW problem on Q_{matrix}: each value in A_2 represents a subgraph in the biclique covering, each tuple $(v, c) \in R_1$ encodes that $v \in V$ belongs to subgraph c, and each tuple $(c, u) \in R_2$ encodes that $u \in U$ belongs to c. The same graph problem has been studied in other domains, such as in data mining [36, 44] or in graph drawing [30]. Interestingly, SSW for Q_{matrix} is also equivalent to the edge-role mining problem and the edge concentration problem. It is known that the weighted biclique covering problem is NP-hard [16, 44]. Hence, there is no poly-time algorithm for the SSW problem on Q_{matrix} unless P = NP. Unfortunately, it is not clear how this result can be extended to other queries.

2 Preliminaries

Connectivity of SJFCQs. We model a SJFCQ Q as a graph G_Q, where each relation R_i is a vertex and there is an edge between $R_i, R_j \in \mathtt{rels}(Q)$ if $\mathtt{attr}(R_i) \cap \mathtt{attr}(R_j) \neq \emptyset$. A SJFCQ Q is *connected* if G_Q is connected, and *disconnected* otherwise. For a disconnected SJFCQ Q, we can decompose it into multiple connected subqueries by identifying the connected components of 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 SJFCQ Q, let $Q_1, Q_2, \cdots, Q_k$ be its connected subqueries. Given a set $S \subseteq \mathtt{dom}(\mathtt{head}(Q))$ of tuples for Q, let $S_i = \pi_{\mathsf{head}(Q_i)} S$ be the corresponding tuples defined for Q_i. It is easy to verify that every synthetic witness for S, if it exists, satisfies $S = \times_{i \in [k]} S_i$. Hence, from now on, we assume that Q is connected.

Lemma 2.1. *For a disconnected SJFCQ Q of k connected components $Q_1, Q_2, \cdots, Q_k$,* SSW *is poly-time solvable if and only if* SSW *is poly-time solvable for (Q_i, S_i) for each $i \in [k]$.*

Head-domination property. The notion of *head-domination* was first introduced by Kimelfeld et al. in [27] to capture the hardness of the deletion propagation problem with minimal view side effects. Later, Hu and Sintos [22] discovered that this notion could be used for capturing the approximability of the SWP problem but not its hardness. In this paper, we use this notion to capture

Algorithm 1: $\texttt{ESW}(Q, S)$

```
D ← ∅, j ← 1 ;
foreach tuple t ∈ S do
    foreach i ∈ [m] do
        t_i ← a tuple defined on attr(R_i) such that π_A(t_i) = π_A(t) for each attribute
          A ∈ head(R_i) and π_A(t_i) = j for each attribute A ∈ attr(R_i) − head(R_i);
        R_i^D ← R_i^D ∪ {t_i};
    j ← j + 1;
if Q(D) = S then return true;
else return false;
```

the poly-time solvability of the SSW problem. For a SJFCQ Q and a subset $E \subseteq \texttt{rels}(Q)$ of relations, $R_i \in \texttt{rels}(Q)$ *dominates* E if every output attribute appearing in any relation of E also appears in R_i, i.e., $\bigcup_{R_j \in E} \texttt{head}(R_j) \subseteq \texttt{head}(R_i)$. We also need to introduce the notion of *existential-connectivity*. We model a SJFCQ Q as a graph $G_Q^{\exists}$, where each relation R_i is a vertex, and there is an edge between $R_i, R_j \in \texttt{rels}(Q)$ if $\texttt{attr}(R_i) \cap \texttt{attr}(R_j) - \texttt{head}(Q) \neq \emptyset$. Let $E_1, E_2, \cdots, E_\kappa \subseteq \texttt{rels}(Q)$ be the connected components of $G_Q^{\exists}$, each corresponding to a subset of relations in Q.

Definition 2.2 ([27]). For SJFCQ Q, let $E_1, E_2, \cdots, E_\kappa$ be the connected components of $G_Q^{\exists}$. Q has the *head-domination property* if for any $i \in [\kappa]$, there is a relation from $\texttt{rels}(Q)$ that dominates E_i.

Berge-acyclic SJFCQs. There are several notions of acyclicity for CQs in the literature. We show our main hardness results using the definition of acyclicity given by Berge [4]. Consider the bipartite graph $G_Q^{\bowtie}$ for a SJFCQ Q, where each attribute $A \in \mathbb{A}$ is a vertex on one side, each relation $R_i \in \texttt{rels}(Q)$ is a vertex on the other side, and there is an edge between A and R_i if $A \in \texttt{attr}(R_i)$. A SJFCQ Q is *Berge-acyclic* if $G_Q^{\bowtie}$ is acyclic. Note that this definition of acyclicity does not allow two relations to have two or more common attributes. But if these attributes always appear together in any relation, they can simply be considered as one "combined" attribute.

Some of the well-studied classes of queries in databases are Berge-acyclic, for example, path, star, and tree queries. Berge-acyclic queries have been used to derive some interesting results in database theory [8, 15, 24, 40]. A more relaxed notion of acyclicity is α-acyclicity (see Appendix C for a formal definition), which does not always capture some intuitive properties. For example, removing a relation from an α-acyclic SJFCQ can make it cyclic, which is quite counter-intuitive. In contrast, Berge-acyclic queries have some nice properties that lead us to the hardness results.

3 ESW for SJFCQs

The ESW problem generalizes the standard satisfiability problem for SJFCQs [12]. Given a query Q (not necessarily a SJFCQ), the goal is to decide whether a database instance D exists such that $Q(D) \neq \emptyset$. It is known that every SJFCQ is satisfiable. However, this is not always true for the ESW problem. Indeed, consider the SJFCQ $Q(A_2, A_3, A_4) :- R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie R_3(A_3, A_4) \bowtie R_4(A_4, A_5)$. If $S = \{(1,2,3), (1,2,4), (6,2,3)\}$, $\texttt{ESW}(Q, S)$ is **false** due to the fact that tuples $\{(1,2), (6,2)\}$ must appear in R_2 and tuples $\{(2,3), (2,4)\}$ must appear in R_3, however, $(6,2,4) \notin S$. On the contrary, if $S = \{(1,2,3), (1,2,4), (6,2,3), (6,2,4)\}$, then $\texttt{ESW}(Q, S)$ is **true**.

We present an algorithm (Algorithm 1) for the $\texttt{ESW}(Q, S)$ problem that runs in polynomial time in terms of $|S|$, and decides whether a synthetic witness D exists such that $Q(D) = S$. We start from an empty database and perform the following step for each tuple $t \in S$. Let t be the tuple

in S processed in the j-th iteration of the outer loop (line 2). For each $R_i \in \texttt{rels}(Q)$, we add a tuple t_i to R_i^D such that $\pi_{\texttt{head}(R_i)}(t_i) = \pi_{\texttt{head}(R_i)}(t)$ and $\pi_A(t_i) = j$, for every non-output attribute $A \in \texttt{attr}(R_i) - \texttt{head}(R_i)$. Note that D contains all tuples added in these iterations. Algorithm 1 returns **true** if $Q(D) = S$, and **false**, otherwise.

Lemma 3.1. *Let D be the database produced by lines 1-6 of Algorithm 1. Then, $S \subseteq Q(D)$.*

Proof. For each tuple $t \in S$, Algorithm 1 adds a tuple t_i to R_i for each $i \in [m]$. It is easy to verify that $t = \pi_{\texttt{head}(Q)}(\bowtie_{i\in[m]} t_i)$, since for all $i \in [m]$, we have $\pi_A(t_i) = \pi_A(t)$ if $A \in \texttt{head}(Q)$ and $\pi_A(t_i) = j$ if $A \notin \texttt{head}(Q)$, where j is the iteration in which Algorithm 1 processes t. Hence, $t \in Q(D)$ and $S \subseteq Q(D)$. □

Lemma 3.2. *Given a SJFCQ Q and a set of tuples $S \subseteq \texttt{dom}(\texttt{head}(Q))$, Algorithm 1 returns* **true** *if and only if* $\texttt{ESW}(Q, S)$ *is* **true**.

proof sketch. From Lemma 3.1, we know that $S \subseteq Q(D)$. It suffices to show that if there exists a tuple $h \in Q(D)$ such that $h \notin S$ then $\texttt{ESW}(Q, S) =$ **false**. We first discuss the high-level idea of the proof in the case where Q has only two relations. Assume the tuple $h \in Q(D) - S$ exists and Q has only two relations R_1 and R_2. Since $h \notin S$, we have that $h = \pi_{\texttt{head}(Q)}(t_1 \bowtie s_2)$, for two different tuples $t, s \in S$, by the definition of Algorithm 1. Again by the definition of Algorithm 1 (line 4), we have that for any body attribute, t_1 and s_2 get different values, i.e, for any attribute $A \in (\texttt{attr}(R_1) \cap \texttt{attr}(R_2)) - \texttt{head}(Q)$, we have $\pi_A(t_1) \neq \pi_A(s_2)$. So, because t_1 and s_2 can be joined together to create the tuple h, the two relations cannot share a body attribute and we have $(\texttt{attr}(R_1) \cap \texttt{attr}(R_2)) - \texttt{head}(Q) = \emptyset$. Moreover, we have $\pi_{\texttt{head}(R_1)\cap\texttt{head}(R_2)}(t_1) = \pi_{\texttt{head}(R_1)\cap\texttt{head}(R_2)}(s_2)$, since t_1 and s_2 can be joined. Assume $\texttt{ESW}(Q, S) =$ **true** and there exists a witness D' for (Q, S). By definition, we have $Q(D') = S$ and hence we have $s, t \in Q(D')$. Let $t_1' \in R_1^{D'}$ and $t_2' \in R_2^{D'}$ (resp, $s_1' \in R_1^{D'}$ and $s_2' \in R_2^{D'}$) be the tuples that create t (resp. s), i.e., $t = \pi_{\texttt{head}(Q)}(t_1' \bowtie t_2')$ (resp. $s = \pi_{\texttt{head}(Q)}(s_1' \bowtie s_2')$). We have $\pi_{\texttt{head}(R_1)}(t_1') = \pi_{\texttt{head}(R_1)}(t) = \pi_{\texttt{head}(R_1)}(t_1)$ and $\pi_{\texttt{head}(R_2)}(s_2') = \pi_{\texttt{head}(R_2)}(s) = \pi_{\texttt{head}(R_2)}(s_2)$. Hence, we have $h = \pi_{\texttt{head}(Q)}(t_1' \bowtie s_2')$, since we know that R_1 and R_2 do not share a body attribute and also $\pi_{\texttt{head}(R_1)\cap\texttt{head}(R_2)}(t_1) = \pi_{\texttt{head}(R_1)\cap\texttt{head}(R_2)}(s_2)$. Therefore we get $h \in Q(D') = S$, which is a contradiction with the assumption $h \notin S$. We show the general proof in Appendix B. □

Algorithm 1 needs $O(|S|)$ time to construct D. For any SJFCQ Q, $Q(D)$ can be computed in polynomial time in terms of $|S|$. Putting everything together, we obtain:

Theorem 3.3. *For a SJFCQ Q and any set $S \subseteq \texttt{dom}(\texttt{head}(Q))$ of tuples, Algorithm 1 returns* **true** *if and only if* $\texttt{ESW}(Q, S)$ *is* **true**, *in polynomial time in terms of $|S|$.*

4 SSW for Head-Dominated SJFCQs

We now present an exact algorithm for the SSW problem for head-dominated SJFCQs, which runs in polynomial time in terms of $|S|$. Algorithm 2 takes as input a SJFCQ Q and a set $S \subseteq \texttt{dom}(\texttt{head}(Q))$ of tuples. First, it checks whether $\texttt{ESW}(Q, S) =$ **false**. If a synthetic witness does not exist, the algorithm returns $\emptyset$. Otherwise, it constructs a smallest synthetic witness D as follows. For each tuple $t \in S$, we iterate through all the relations $R_i \in \texttt{rels}(Q)$. For each relation R_i, we add a tuple t_i defined on $\texttt{attr}(R_i)$ to R_i^D, such that $\pi_{\texttt{head}(R_i)}(t_i) = \pi_{\texttt{head}(R_i)}(t)$ and for each attribute $A \in \texttt{attr}(R_i) - \texttt{head}(Q)$, $\pi_A(t_i) = *$ for a special value $*$.

The new algorithm (Algorithm 2) is essentially the same as Algorithm 1, with one minor but important difference. In Algorithm 1, we followed a defensive mechanism, adding a different value j for each different tuple in S to the body attributes $\texttt{attr}(R_i) - \texttt{head}(R_i)$ for $i \in [m]$. Notice that in the ESW problem, it only matters whether $\texttt{ESW}(Q, S)$ is true or false, and we do not care about constructing a minimum witness. Hence, to be safe, each time we encountered a tuple $t \in S$, we

Algorithm 2: EasySSW(Q, S)

```
1 if ESW(Q, S) = false then return ∅;
2 D ← ∅;
3 foreach tuple t ∈ S do
4     foreach i ∈ [m] do
5         t_i ← a tuple defined on attr(R_i) such that π_A(t_i) = π_A(t) for each attribute
            A ∈ head(R_i) and π_A(t_i) = ∗ for each attribute A ∈ attr(R_i) − head(R_i);
6         R_i^D ← R_i^D ∪ {t_i};
7 return D;
```

added a tuple to each relation so that they could be joined to generate the result t. On the other hand, for the SSW problem, we must compute the smallest witness. We follow an aggressive approach, adding the same value $*$ in all body attributes $\mathtt{attr}(R_i) - \mathtt{head}(R_i)$ (for each $i \in [m]$) for all tuples in S. Of course, if Q was any general SJFCQ, we would expect that such a construction might create multiple new results that do not belong to S, i.e., if D is the constructed database instance, then $Q(D) \supset S$, leading to an incorrect result. However, in this section, we focus only on head-dominated SJFCQs. We take advantage of the properties of head-dominated SJFCQs to show that if we use the same character $*$ for the values of the body attributes over all tuples in S, we always only generate tuples that exist in S. Intuitively, in head-dominated queries, based on our construction, the unwanted additional tuples will get filtered out from the results by the dominating relations.

We first show that D returned by Algorithm 2 is a valid synthetic witness of (Q, S) in Lemma 4.1 and then prove the optimality of D in Lemma 4.2.

Lemma 4.1. *For any SJFCQ Q with the head-domination property and any set of tuples $S \subseteq \mathtt{dom}(\mathtt{head}(Q))$, such that $\mathtt{ESW}(Q, S) =$ **TRUE**, the D returned by Algorithm 2 is a synthetic witness, i.e., $Q(D) = S$.*

Proof. Direction $Q(D) \supseteq S$. Let $t \in S$ be an arbitrary tuple. By definition, for every $i \in [m]$, $t_i \in R_i^D$, and $\pi_{\mathtt{head}(Q)}(\bowtie_{i\in[m]} t_i) = t$ so $t \in Q(D)$, and the result follows. Note that this direction holds even when Q does not have head-domination property.

Direction $Q(D) \subseteq S$. Let κ be the number of connected components in $G_Q^{\exists}$ and for each $j \in [\kappa]$, let $\hat{R}_j$ be the dominating relation of the j-th connected component. Wlog, we consider the minimum number of dominating relations, so no dominating relations $\hat{R}_{j_1}, \hat{R}_{j_2}$, for $j_1 \neq j_2 \in [k]$ belong in the same existential connected component.[3] By the construction of D, $Q(D) = \pi_{\mathtt{head}(Q)}(\bowtie_{j\in[\kappa]} \hat{R}_j^D)$. Indeed, all the body attributes have the same value $*$, and for each tuple t in a relation R in the j-th existential connected component $\pi_{\mathtt{head}(R)} t \in \pi_{\mathtt{head}(R)} \hat{R}_j$, so all relations except the dominating ones do not affect the query result in our constructed database D. As $\mathtt{ESW}(Q, S) =$ **true**, there exists a witness D' (without dangling tuples) such that $Q(D') = S$. Let $h \in Q(D)$, and let h_j be the tuple in $\hat{R}_j^D$ such that $h = \pi_{\mathtt{head}(Q)}(\bowtie_{j\in[\kappa]} h_j)$. We need to show that $h \in S$. By the construction of D, $\pi_{\mathtt{head}(\hat{R}_j)}(h_j) \in \pi_{\mathtt{head}(\hat{R}_j)}(S)$, for every $j \in [\kappa]$, so there exists a tuple $g_j \in \hat{R}_j^{D'}$ such that $\pi_{\mathtt{head}(\hat{R}_j)}(h_j) = \pi_{\mathtt{head}(\hat{R}_j)}(g_j)$. If no such tuple g_j existed, then $Q(D') = S$ would not

[3] Assume that there are two dominating relations $\hat{R}_{j_1}, \hat{R}_{j_2}$ in E_j. If $\hat{R}_{j_1}$ is dominating for E_j, then we can skip $\hat{R}_{j_2}$ becaus eit is dominated by $\hat{R}_{j_1}$. If the dominating relation of E_j, say $\hat{R}_j$, is in a different connected component E_h, we can safely remove $\hat{R}_{j_1}, \hat{R}_{j_2}$ from the set of dominating relations and add $\hat{R}_j$, because it dominates both $\hat{R}_{j_1}, \hat{R}_{j_2}$. We can continue the same process until every component has at most one dominating relation.

include any tuple $s \in S$ with $\pi_{\mathsf{head}(\hat{R}_j)}(s) = \pi_{\mathsf{head}(\hat{R}_j)}(h_j)$, which is a contradiction since we know $\pi_{\mathsf{head}(\hat{R}_j)}(h_j) \in \pi_{\mathsf{head}(\hat{R}_j)}(S)$. Let $\{R_{j,1}, \ldots, R_{j,b}\}$ be the relations of the j-th connected component and let $\bar{\mathbb{A}}_j = \bigcup_{i\in[b]} \mathsf{head}(R_{j,i})$. For each $i \in [b]$, there exists a tuple $\ell_i \in R_{j,i}^{D'}$ such that $\bowtie_{i\in[b]} \ell_i \neq \emptyset$ and $\pi_{\bar{\mathbb{A}}_j}(\bowtie_{i\in[b]} \ell_i) = \pi_{\bar{\mathbb{A}}_j} g_j$. If these conditions do not hold, then $Q(D') = S$ would not include any tuple $s \in S$ with $\pi_{\mathsf{head}(\hat{R}_j)}(s) = \pi_{\mathsf{head}(\hat{R}_j)}(h_j) = \pi_{\mathsf{head}(\hat{R}_j)}(g_j)$, which is a contradiction. We can argue in the same way to show that g_j is not filtered by relations belonging to the same connected component as $\hat{R}_j$. By the definition of $G_Q^{\exists}$, any two distinct connected components do not share any non-output attributes, and hence $\bowtie_{j\in[\kappa]} g_j \neq \emptyset$. Putting everything together, $h = \bowtie_{j\in[\kappa]} \pi_{\mathsf{head}(\hat{R}_j)}(h_j) = \bowtie_{j\in[\kappa]} \pi_{\mathsf{head}(\hat{R}_j)}(g_j) \in Q(D')$. As D' was assumed to be a witness for S, we conclude that $h \in S$. □

Lemma 4.2. *If* $\mathsf{ESW}(Q, S) =$ **true**, $|D| \leq |D'|$ *for any synthetic witness* D' *of* (Q, S), *where* D *is the output of Algorithm 2.*

Proof. By the construction of D, $|R_i^D| = |\pi_{\mathsf{head}(R_i)}(S)|$ for all $i \in [m]$. For each tuple $t \in S$, $t \in Q(D')$, and hence for each $i \in [m]$, there is a tuple $t_i' \in R_i^{D'}$ such that $\pi_{\mathsf{head}(R_i)}(t_i') = \pi_{\mathsf{head}(R_i)}(t)$. So, for each $i \in [m]$, $|R_i^{D'}| \geq |\pi_{\mathsf{head}(R_i)}(S)| = |R_i^D|$, which implies $|D'| \geq |D|$. □

Recall that $\mathsf{ESW}(Q, S)$ can be computed in $O(|S|)$ time using Algorithm 1. The remaining operations (lines 3-6) take $O(|S|)$ time to construct D. Hence, we obtain:

Theorem 4.3. *For any head-dominated SJFCQ* Q *and a set* $S \subseteq \mathsf{dom}(\mathsf{head}(Q))$ *of tuples such that* $\mathsf{ESW}(Q, S) =$ **true**, *Algorithm 2 returns a solution to* $\mathsf{SSW}(Q, S)$ *in polynomial time in terms of* $|S|$.

If $S = Q(\bar{D})$ for a given $\bar{D}$, then a faster algorithm exists for some queries as shown in Appendix C.

Remark. While we focus on data complexity in this paper, our algorithms for the ESW problem from Theorem 3.3, and the SSW problem (for head-dominated SJFCQs) from Theorem 4.3, run in polynomial time in terms of $|S|$ even in combined complexity.

5 SSW for SJFCQs Without the Head-domination Property

This section shows the hardness of SSW for Berge-acyclic SJFCQs without the head-domination property. In Section 5.1, we prove that SSW cannot be solved in polynomial time for the commonly studied path queries through a reduction from the *vertex cover* problem: Given an undirected graph $G = (V, E)$ with vertices V and edges E, it asks to find a smallest subset $\rho_G \subseteq V$ of vertices (called a "cover") such that each edge $e \in E$ has at least one endpoint in ρ_G. Throughout all the proofs, we assume that G is simple and each vertex in G is incident to at least two other vertices. We can safely make this assumption since the vertex cover problem is NP-Hard even for cubic graphs [19].

In Section 5.1, we show that for any fixed constant k, the SSW problem is not poly-time solvable for k-path SJFCQs unless unless P = NP. Given an arbitrary graph $G(V, E)$, we can construct within polynomial time with respect to the graph size ($|E|$ and $|V|$), an input set S of $3 \cdot |E|$ tuples to the SSW on a k-path SJFCQ. Any solution to the SSW problem can be used to construct an optimal vertex cover for the graph G in polynomial time with respect to the graph size. In Section 5.2, we extend the hardness result to *augmented path queries* that will serve as the *hard core* of general Berge-acyclic SJFCQs. Finally, in Section 5.3, we turn to Berge-acyclic SJFCQs. As any Berge-acyclic SJFCQ Q without head-domination property always admits an augmented path, we can show a reduction from SSW over an augmented path query to SSW over Q in polynomial time. In this way, we show that SSW is not poly-time solvable for Berge-acyclic SJFCQs without head-domination property unless P = NP.

To help the presentation, we introduce the following terminologies. A (synthetic) witness is *reduced* if it does not contain dangling tuples (see Section 1.1). As SSW computes the smallest synthetic witness, we only consider reduced synthetic witnesses and refer to them simply as "synthetic witnesses". For a database D, the *active domain* of an attribute A, noted as $\mathsf{adom}^D(A)$, is defined as the set of values from $\mathsf{dom}(A)$ that appear in at least one tuple of D. When the context is clear, we will drop the superscript D in $\mathsf{adom}^D(A)$. For a SJFCQ Q and a database D, we define the *instance hypergraph* H^D of D as follows. For every attribute $A \in \mathtt{attr}(Q)$ and every value $v \in \mathsf{adom}^D(A)$, we define a vertex $\gamma(A, v)$. Each tuple $t \in D$ from a relation R is encoded as a hyperedge that includes the vertices $\{\gamma(A, v) \mid A \in \mathtt{attr}(R), v = \pi_A t\}$. An *instance path* between a pair of vertices b_1, b_k in H^D is a sequence of vertices $b_1, b_2, \cdots, b_k$ such that there exists $k-1$ hyperedges $t_1, t_2, \cdots, t_{k-1}$ in H^D such that for every $i \in [k-1]$, $b_i, b_{i+1} \in t_i$, and $t_1, t_2, \cdots, t_{k-1}$ are from $k-1$ different relations. A pair of vertices are *incident* if they appear in at least one common hyperedge in H^D.

5.1 Path Queries

In this part, we first focus on the class of *path queries*, defined as below:

$$Q_{\text{path}}(A_1, A_{k+1}) :- R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie \cdots \bowtie R_k(A_k, A_{k+1}).$$

We show the hardness proof for the path query of length k, where $k \geq 2$ is an arbitrary constant integer number. By slightly abusing the notation, we may use Q_{path} to refer to both the class of path queries or a specific in this class.

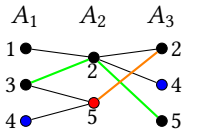

Fig. 1. The instance graph constructed.

Example 5.1. Consider a path query with $k = 2$, $Q(A_1, A_3) :- R_1(A_1, A_2) \bowtie R_2(A_2, A_3)$, over an instance with two relations $R_1 = \{(1, 2), (3, 2), (3, 5), (4, 5)\}$ and $R_2 = \{(2, 2), (2, 4), (2, 5), (5, 2)\}$. The instance hypergraph (graph), of the above example is shown in Figure 1. We note that for every path query and any database, the instance hypergraph is in fact a graph. For simplicity, the node $\gamma(A, v)$ is shown as node v under the attribute A. For instance, the vertex marked as red in the figure corresponds to $\gamma(A_2, 5)$. Each tuple in R_1 or R_2 corresponds to an edge. For example, the orange edge corresponds to the tuple $(5, 2)$ in R_2. A key property of this graph that we mainly use in the proofs, is that any tuple (ϕ_1, ϕ_2) in the results of the query $Q(A_1, A_3) :- R_1(A_1, A_2) \bowtie R_2(A_2, A_3)$ corresponds to an instance path from a node under A_1 to a node under A_3. More formally, we have the tuple (ϕ_1, ϕ_2) in the results, if and only if there is a path from the node $\gamma(A_1, \phi_1)$ to the node $\gamma(A_3, \phi_2)$. For example, for the given instance, we have the tuple $(3, 5)$ in the results of the query because the green path shown in the figure exists. However, the tuple $(4, 4)$ is not in the results, since there is no path between the blue nodes shown in the graph. The green path shows that the tuple $(3, 2)$ from R_1 can be joined to the tuple $(2, 5)$, to create the tuple $(3, 5)$ in the results.

Next, we show that if there is a poly-time algorithm, called $\mathcal{A}$, that can compute $\mathsf{SSW}(Q_{\text{path}}, S)$ for an arbitrary set $S \subseteq \mathsf{dom}(\mathsf{head}(Q_{\text{path}}))$ of tuples in polynomial time in terms of $|S|$, there exists a poly-time algorithm $\mathcal{A}'$ for the vertex cover problem. We obtain:

Theorem 5.2. SSW *is not poly-time solvable for any path query* Q_{path}, *unless* P = NP.

Step 1: Construct S for Q_{path} from input graph G. We are given a graph $G = (V, E)$. Let $V = \{v_1, v_2, \cdots, v_{|V|}\}$ and $E = \{e_1, e_2, \cdots, e_{|E|}\}$, where each e_i is a set of two vertices from V. For an example, consider the triangle graph in Figure 2. We construct a set $S \subseteq \mathsf{dom}(A_1) \times \mathsf{dom}(A_{k+1})$ of tuples as $S = \{(e_i, v_j) \in E \times V : v_j \in e_i\} \cup \{(e_i, *) : e_i \in E\}$, for some special value $*$. This step takes polynomial time in terms of $|G|$, and the resulting S is polynomial in terms of the size of G. Recall that G

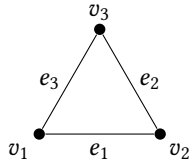

Fig. 2. An input graph $G = (V, E)$ with $V = \{v_1, v_2, v_3\}$ and $E = \{e_1, e_2, e_3\}$

is simple and each vertex is connected to at least two other vertices. For any reduced synthetic witness for (Q_{path}, S), we slightly abuse the notation, and we use V to denote the values of $\mathsf{adom}(A_{k+1})$ that correspond to the vertices of the graph G and E to denote the values of $\mathsf{adom}(A_1)$ that correspond to the edges of the graph G. Hence, we always have $\mathsf{adom}(A_{k+1}) = V \cup \{*\}$ and $\mathsf{adom}(A_1) = E$. Furthermore, notice that any instance path from a value in $\mathsf{adom}(A_1)$ to a value in $\mathsf{adom}(A_{k+1})$ creates an output tuple. Intuitively, as we will see later, the $*$ value acts like a selector of a vertex in the vertex cover. The set S for the triangle graph is shown in Figure 3.

Let D be any reduced witness of (Q_{path}, S). Before proceeding with the technical details, we review some properties of the instance graph H^D that can be constructed from D. For every tuple $(x, y) \in S$, there exists a path from the value $x \in \mathsf{adom}^D(A_1)$ to the value $y \in \mathsf{adom}^D(A_{k+1})$ in H^D. Moreover, if there exists a path from a value $x \in \mathsf{adom}^D(A_1)$ to a value $y \in \mathsf{adom}^D(A_{k+1})$ in H^D, it implies $(x, y) \in Q_{\text{path}}(D)$, and hence $(x, y) \in S$. For simplicity, we will use the terms "value", "node", and "vertex" in H^D interchangeably. Let $a^D_{i,j}$ denote the j-th value in $\mathsf{adom}^D(A_i)$, for $i = 2, \ldots, k$. If D is clear from the context we use $a_{i,j}$. By saying a value $a_{k,j}$ is incident to a value v in V, we mean that there exists a tuple $(a_{k,j}, v)$ in relation R^D_k under witness D, or equivalently, there exists an edge connecting node $a_{k,j}$ and node v in the instance graph. From the construction of S, we note that for every $e_p = (v_i, v_j) \in E$, there exists a path from $e_p \in \mathsf{adom}^D(A_1)$ to $v_i \in \mathsf{adom}^D(A_{k+1})$, a path from $e_p \in \mathsf{adom}^D(A_1)$ to $v_j \in \mathsf{adom}^D(A_{k+1})$ and a path from $e_p \in \mathsf{adom}^D(A_1)$ to $* \in \mathsf{adom}^D(A_{k+1})$, in the instance graph H^D. Furthermore, it is straightforward to see that there is no path from $e_p \in \mathsf{adom}^D(A_1)$ to any other value in $\mathsf{adom}^D(A_{k+1})$.

Intuition behind next steps. Let D be an optimum synthetic witness returned by $\mathcal{A}$ on $\mathsf{SSW}(Q_{\text{path}}, S)$, where S is constructed by **Step 1**. In **Step 2**, we show that D has some useful properties. More specifically, in Lemmas 5.3, 5.4 we show that any value $a_{k,j}$ in $\mathsf{adom}^D(A_k)$ is incident to either i) exactly one value in V, or ii) $*$ and exactly one value in V, or iii) $*$ and two values of V. Furthermore, we show that $|\mathsf{adom}(A_k)| \geq |V|$. Although this gives us some structure, it is still not enough to construct the vertex cover. In **Step 3**, these properties are used to derive from D a new optimum witness D' (in polynomial time) that has some even nicer properties. In **Step 3**, we show the construction of D'. In Lemmas 5.5 and 5.7, we prove that the newly constructed witness D' is an optimum solution for SSW on Q_{path} such that every value $a_{k,j} \in \mathsf{adom}^{D'}(A_k)$ is either i) incident to exactly one value in V or ii) it is incident to exactly one value in V and $*$. Furthermore, we show that every value v in V is incident to exactly one value in $\mathsf{adom}^{D'}(A_k)$. These properties allow us to select a vertex cover for G.

In **Step 4**, we show how to construct an optimal vertex cover ρ_G for G based on D'. In the new witness D', we know that a value v_j from V is incident to exactly one value in $\mathsf{adom}^{D'}(A_k)$. Say that $a_{k,j}$ is this value. If $a_{k,j}$ is also incident to $*$ then we select v_j in the vertex cover we construct. If $a_{k,j}$ is only incident to v_j and not incident to $*$ then we do not select v_j in the vertex cover solution (notice that $a_{k,j}$ cannot be incident to two values from V). This is why we characterize $*$ as the selector of the vertex cover solution. Let $\rho_G \subseteq V$ be the set computed as described above. In Lemma 5.8, we show that the constructed solution for vertex cover $\rho_G \subseteq V$ is an optimal vertex cover. By contradiction, we first show that ρ_G is a vertex cover. If it was not a vertex cover, then there would be an edge $e_p = (v_1, v_2)$ such that both v_1, v_2 are not in ρ_G. However, $(e_p, *) \in S$ so there should be a path from e_p to $*$ in the instance graph. This path should pass a value/node $a_{k,j}$ in $\mathsf{adom}^{D'}(A_k)$. By the properties of D', we know that $a_{k,j}$ cannot be incident only to $*$ and it should be incident to either v_1 or v_2. Lastly, we argue that ρ_G is minimal. If there was a different vertex cover ρ'_G with $|\rho'_G| < |\rho_G|$ then removing the tuples $(a_{k,j}, *)$ for each $v_j \in \rho_G$ from D' and adding tuples

A_1	A_4
e_1	$*$
e_1	v_1
e_1	v_2
e_2	$*$
e_2	v_2
e_2	v_3
e_3	$*$
e_3	v_1
e_3	v_3

R_1

A_1	A_2
e_1	$a_{2,1}$
e_1	$a_{2,3}$
e_2	$a_{2,1}$
e_2	$a_{2,4}$
e_3	$a_{2,2}$
e_3	$a_{2,5}$

R_2

A_2	A_3
$a_{2,1}$	$a_{3,1}$
$a_{2,2}$	$a_{3,1}$
$a_{2,2}$	$a_{3,2}$
$a_{2,3}$	$a_{3,3}$
$a_{2,4}$	$a_{3,4}$
$a_{2,4}$	$a_{3,5}$
$a_{2,5}$	$a_{3,5}$

R_3

A_3	A_4
$a_{3,1}$	$*$
$a_{3,2}$	v_1
$a_{3,3}$	$*$
$a_{3,3}$	v_1
$a_{3,3}$	v_2
$a_{3,4}$	v_2
$a_{3,5}$	$*$

Fig. 3. A set of input tuples S (left) and a non-optimal synthetic witness (right) for (Q_{path}, S), where Q_{path} is a path query with $k = 3$.

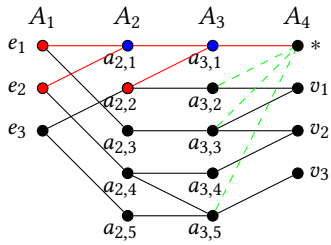

Fig. 4. Instance hypergraph of the witness in Figure 3, where solid lines are edges. A smaller witness can be obtained by removing red edges and adding dashed green edges.

$(a_{k,h}, *)$ for each $v_h \in \rho'_G$ would lead to a smaller a synthetic witness for $\mathsf{SSW}(Q_{\text{path}}, S)$, coming to a contradiction.

Step 2: Compute a solution D to $\mathsf{SSW}(Q_{\text{path}}, S)$. We invoke the (hypothetical) algorithm $\mathcal{A}$ to compute a solution to $\mathsf{SSW}(Q_{\text{path}}, S)$ denoted by D within polynomial time in terms of $|S|$ and also in $|G|$ (recall that our goal is to show that if such algorithm exists then there exists a polynomial time algorithm for the vertex cover problem). Before proceeding to the next step, we show some important properties of any synthetic witness (possibly non-optimal) to (Q_{path}, S) in Lemma 5.3 and any solution (optimal witness) to $\mathsf{SSW}(Q_{\text{path}}, S)$ in Lemma 5.4.

Lemma 5.3. *In any reduced synthetic witness $\hat{D}$ of (Q_{path}, S), (1) in the graph $H^{\hat{D}}$ each value $a_{k,i} \in \mathsf{adom}^{\hat{D}}(A_k)$ is incident to at most two values in V, and (2) $|\mathsf{adom}^{\hat{D}}(A_k)| \geq |V|$.*

Proof. As each edge contains two vertices in G, each value $a_{k,i} \in \mathsf{adom}(A_k)$ is incident to at most 2 values in V. We next show for each value $v_i \in V$ that is only incident to one value $a_{k,j} \in \mathsf{adom}(A_k)$, it must be the case that $a_{k,j}$ is not incident to any other value in V. For the sake of contradiction, assume that $a_{k,j}$ is incident to v_i and some other value v_j. Then there exists exactly one $e_p \in \mathsf{adom}(A_1)$ with an instance path to $a_{k,j}$ as otherwise, G would contain duplicate edges. Thus, v_i appears in only one edge e_p, resulting in a contradiction, since any vertex in G is contained in at least two edges. Now, we remove all tuples containing such a value in V, i.e., we remove all the tuples $(a_{k,j}, v_i) \in R_k$ where v_i is only incident to $a_{k,j}$. We also remove all tuples $r \in R_k$ such that $\pi_{A_{k+1}}(r) = *$. Let R'_k be the remaining relation. Let $\mathsf{adom}'(A_k) \subseteq \mathsf{adom}(A_k), V' \subseteq V$ be the remaining set of values that appear in some tuple in R'_k. From the analysis above, for every value in V, we remove a distinct value from $\mathsf{adom}(A_k)$, and we remove additional tuples from R_k that might remove more values from $\mathsf{adom}(A_k)$. Hence, $|\mathsf{adom}(A_k)| - |\mathsf{adom}'(A_k)| \geq |V| - |V'|$. Moreover, every value in $\mathsf{adom}'(A_k)$ appears in one or two tuples in R'_k, and every value in V' appears in at least two tuples in R'_k, hence $|\mathsf{adom}'(R_k)| \geq |V'|$. Together with $|\mathsf{adom}(A_k)| - |\mathsf{adom}'(A_k)| \geq |V| - |V'|$, we get $|\mathsf{adom}(A_k)| \geq |V|$. □

Lemma 5.4. *In any solution $\hat{D}$ to $\mathsf{SSW}(Q_{\text{path}}, S)$, each value $a_{k,i} \in \mathsf{adom}^{\hat{D}}(A_k)$ must fall into one of the following three cases in the graph $H^{\hat{D}}$: (1) only incident to one value in V; (2) incident to $*$ and one value in V; and (3) incident to $*$ and two values in V.*

Proof. We first show that if $a_{k,i}$ is incident to $*$, then $a_{k,i}$ must be incident to at least one value in V. Consider an arbitrary attribute A_i for some $i \in [k]$. A value $a_{i,j} \in \mathsf{adom}(A_i)$ is marked as

R_1

A_1	A_2
e_1	$a_{2,1}$
e_2	$a_{2,2}$
e_3	$a_{2,3}$

R_2

A_2	A_3
$a_{2,1}$	$a_{3,1}$
$a_{2,1}$	$a_{3,2}$
$a_{2,2}$	$a_{3,2}$
$a_{2,2}$	$a_{3,3}$
$a_{2,3}$	$a_{3,2}$
$a_{2,3}$	$a_{3,3}$

R_3

A_3	A_4
$a_{3,1}$	$*$
$a_{3,1}$	v_1
$a_{3,2}$	$*$
$a_{3,2}$	v_2
$a_{3,3}$	v_3

Fig. 5. A solution to $\mathsf{SSW}(Q_{\text{path}}, S)$.

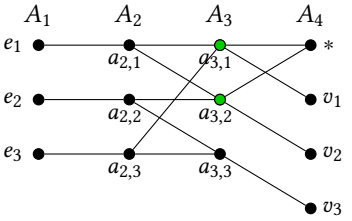

Fig. 6. Instance graph of Figure 5.

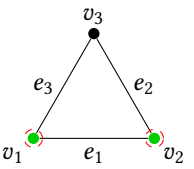

Fig. 7. An optimal vertex cover for G constructed from Figure 6.

blue if there exists no value in V having an instance path to $a_{i,j}$. Moreover, $a_{i,j}$ is marked as *red* if it is not blue and it is incident to some blue value. We note that in an optimal witness, every value in $\mathsf{adom}(A_1)$ has an instance path to a value in V so if a blue value exists, then at least one value should be colored red. We next show that in an optimal synthetic witness such as $\hat{D}$, no value will be marked as blue, and hence no value will be marked as red. For the sake of contradiction, assume that a blue value exists. In this case, as we argued, at least one red value exists. Consider any red value $a_{i,j} \in \mathsf{adom}(A_i)$ for some $i \in [k-1]$. By definition of red values, there must exist some value in V, with an instance path to $a_{i,j}$. Let $a_{k,h} \in \mathsf{adom}(A_k)$ be the value on this path. We remove all tuples from R_i containing both $a_{i,j}$ and a blue value in $\mathsf{adom}(A_{i+1})$, and add the tuple $(a_{k,h}, *)$ to R_k instead. At last, we remove every tuple that contains a blue value. By adding the tuple $(a_{k,h}, *)$, we make sure that $a_{i,j}$ has an instance path to $*$. Thus, after these operations, all the red values will have an instance path to $*$. By definition, every value $e_p \in \mathsf{adom}(A_1)$ has an instance path to a value in V, so e_p is not a blue value. Therefore, if there exists a blue value in the instance path from e_p to $*$, there is an instance sub-path from e_p to a red value. Since we made sure that every red value has a path to $*$, and e_p has a path to a red value, we know that e_p will maintain a path to $*$ after these operations. It can be easily checked that if there exists at least one blue value then the resulting database instance is strictly smaller than $\hat{D}$, contradicting that $\hat{D}$ is a solution to $\mathsf{SSW}(Q_{\text{path}}, S)$. For an example, see Figures 3, and 4. Hence, no blue value exists and every value in $\mathsf{adom}(A_k)$ is incident to some value in V.

At last, we show that if $a_{k,i}$ is incident to two values $v_1, v_2 \in V$, then $a_{k,i}$ is also incident to $*$. Note that in this case, exactly one value in $\mathsf{adom}(A_1)$ has an instance path to $a_{k,i}$ since G does not have duplicate edges. Let e be the value in $\mathsf{adom}(A_1)$ such that there exists an instance path from e to $a_{k,i}$. Notice that $e = \{v_1, v_2\} \in E$, and hence, the tuple $(e, *)$ must appear in $Q_{\text{path}}(\hat{D})$. So either $a_{k,i}$ is incident to $*$ or there exists another path that starts from $e \in \mathsf{adom}(A_1)$ and passes through a value $a_{k,j} \neq a_{k,i}$ such that $a_{k,j}$ is incident to $*$. If $a_{k,i}$ is incident to $*$, then the lemma follows. Otherwise, from the first property, we know that $a_{k,j}$ must also be incident to some value in V. This value must be either v_1 or v_2, since $e = \{v_1, v_2\}$. Wlog, assume that $a_{k,j}$ is incident to v_1. Since, e has an instance path to both $a_{k,i}$ and $a_{k,j}$, it is easy to see that we can remove the tuple $(a_{k,i}, v_1)$ from $\hat{D}$, and this contradicts the optimality of $\hat{D}$. □

Step 3: Transform D into D' with desired properties. We next transform D into another solution D' for $\mathsf{SSW}(Q_{\text{path}}, S)$ with desirable properties. Our goal is to eliminate the possibility that some value in $\mathsf{adom}(A_k)$ is incident to two values in V. Suppose $a_{k,i_1} \in \mathsf{adom}(A_k)$ is incident to $v_{j_1}, v_{j_2} \in V$. There is exactly one value $e_1 \in \mathsf{adom}(A_1)$ with an instance path to a_{k,i_1}, since there are no duplicated edges in G. Thus, there exists a unique value $a_{k-1,l_1} \in \mathsf{adom}(A_{k-1})$ which is incident to a_{k,i_1}. From Lemma 5.4, a_{k,i_1} is also incident to $*$. Each vertex in G appears in at least two edges. So, there exists another value $a_{k,i_2} \in \mathsf{adom}(A_k)$ incident to v_{j_1}.

We distinguish two more cases:

- If a_{k,i_2} is not incident to any other values in V except v_{j_1}, we remove the tuple (a_{k,i_1}, v_{j_1}) from R_k and add tuple (a_{k-1,l_1}, a_{k,i_2}) to R_{k-1}. This does not change the size of the witness.
- If a_{k,i_2} is incident to some other value $v_{j_3} \in V$, by the same argument as before, we know that only one value $e_2 \in \mathsf{adom}(A_1)$ has a path to a_{k,i_2}, and from Lemma 5.4, we have that a_{k,i_2} is incident to $*$. Let $a_{k-1,l_2} \in \mathsf{adom}(A_{k-1})$ be the unique value incident to a_{k,i_2}. From Lemma 5.4, a_{k,i_1} is also incident to $*$. In this case, we remove tuples $(a_{k,i_2}, *)$, (a_{k,i_2}, v_{j_1}), and (a_{k,i_1}, v_{j_2}), and add the three tuples (a_{k-1,l_2}, a_{k,i_1}), (a_{k-1,l_1}, a_{k,i_3}), and (a_{k,i_3}, v_{j_2}) instead, where a_{k,i_3} is a new value we add to $\mathsf{adom}(A_k)$ in this step.

The new database D' is a valid synthetic witness to (Q_{path}, S) with $|D'| = |D|$ and $|\mathsf{adom}^{D'}(A_k)| \geq |\mathsf{adom}^{D}(A_k)|$, since we may add a new value a_{k,i_3} in the second case. See Figures 5 and 6.

By the argument above, we obtain the following:

Lemma 5.5. *Any solution D to $\mathsf{SSW}(Q_{\mathsf{path}}, S)$ can be transformed into another solution D' within polynomial time in terms of $|S|$, in which each value $a_{k,i} \in \mathsf{adom}^{D'}(A_k)$ falls into one of the two cases in the graph $H^{D'}$: (1) incident to exactly one value in V; (2) incident to $*$ and one value in V.*

Lemma 5.6. *In D', each value $v \in V$ is only incident to exactly one value in $\mathsf{adom}^{D'}(A_k)$.*

Proof. By contradiction, we assume a value $v \in V$ is incident to two distinct values a_{k,i_1}, a_{k,i_2} in $\mathsf{adom}(A_k)$. Both a_{k,i_1} and a_{k,i_2} are only incident to v from V. Wlog, assume that a_{k,i_1} is incident to $*$. If none of a_{k,i_1}, a_{k,i_2} are incident to $*$, we still consider a_{k,i_1} next. Let $L \subseteq \mathsf{adom}(A_{k-1})$ be the set of values incident to a_{k,i_2}. We remove all tuples $\{(a, a_{k,i_2}) \mid a \in L\}$ from R_{k-1}, the tuple (a_{k,i_2}, v) from R_k, and the tuple $(a_{k,i_2}, *)$ if it exists. Next, we insert tuples $\{(a, a_{k,i_1}) \mid a \in L\}$ to R_{k-1}. Let D'' be the resulting database, which is a synthetic witness to (Q_{path}, S) with $|D''| \leq |D'| - 1 < |D|$, contradicting the fact that D is a solution to $\mathsf{SSW}(Q_{\mathsf{path}}, S)$. □

By Lemmas 5.5 and 5.6, $|V| = |\mathsf{adom}^{D'}(A_k)|$. As analyzed, $|\mathsf{adom}^{D'}(A_k)| \geq |\mathsf{adom}^{D}(A_k)|$. By Lemma 5.3, $|\mathsf{adom}^{D}(A_k)| \geq |V|$. Hence, we obtain:

Lemma 5.7. *In any solution D to $\mathsf{SSW}(Q_{\mathsf{path}}, S)$, $|\mathsf{adom}^{D}(A_k)| = |V|$.*

The two lemmas above, will later allow us to show the hardness of SSW over a more general class of queries in the next section.

Step 4: Construct an optimal vertex cover ρ_G for G. For each $v_j \in V$, if there exists a value $a_{k,i} \in \mathsf{adom}^{D'}(A_k)$, such that $a_{k,i}$ is incident to both v_j and $*$ in the graph $H^{D'}$, we include v_j in the vertex cover denoted by ρ_G. This step takes polynomial time in terms of $|G|$. See Figures 6, 7 for an example of constructing a vertex cover from D'.

Lemma 5.8. *ρ_G is an optimal vertex cover for G.*

Proof. First, ρ_G is a vertex cover of G. By contradiction, assume there exists an edge $e_p = (v_1, v_2) \in E$ with $v_1, v_2 \notin \rho_G$. By Lemma 5.4, there is no value $a_{k,i} \in \mathsf{adom}(A_k)$, such that $a_{k,i}$ is only incident to $*$. Furthermore, by the definition of ρ_G, $a_{k,i}$ is incident to $*$ and one of the vertices v_1 or v_2 in V. So, $(e_p, *) \notin Q(D')$, coming to a contradiction. We next show that ρ_G is an optimal vertex cover. Assume a vertex cover ρ'_G for G exists, such that $|\rho'_G| < |\rho_G|$. Wlog, let $a_{k,i} \in \mathsf{adom}(A_k)$ be the value incident to $v_i \in \mathsf{adom}(A_{k+1})$ for every $v_i \in V$ (there always exist such unique values, by Lemma 5.5). By definition, for each $v_j \in \rho_G$, the value $a_{k,j}$ is incident to $*$ in addition to v_j. Let D'' be the set of tuples after removing $(a_{k,j}, *)$ for each $v_j \in \rho_G$ from D' and adding tuples $(a_{k,h}, *)$ for each $v_h \in \rho'_G$. Now, D'' is a valid synthetic witness with $|D''| < |D'|$, contradicting that D' is a solution to $\mathsf{SSW}(Q_{\mathsf{path}}, S)$. □

Putting all four steps together, we obtained a poly-time algorithm $\mathcal{A}'$ for the vertex cover problem. Hence, such a poly-time algorithm $\mathcal{A}$ does not exist for $\mathsf{SSW}(Q_{\text{path}}, S)$, unless $\mathsf{P} = \mathsf{NP}$.

Theorem 5.9. SSW *is not poly-time solvable for* Q_{path}*, unless* $\mathsf{P} = \mathsf{NP}$.

5.2 Augmented Path Queries

We next consider a slightly larger class of queries, noted as *augmented path queries*:

$$Q_{\text{apath}}(A_1, A_{k+1}) :- \left(\bowtie_{i\in[k]} R_i(A_i, A_{i+1})\right) \bowtie \left(\bowtie_{i\in[k+1]} \bowtie_{j\in[w_i]} R_{k+W_{i-1}+j}(A_i)\right),$$

where $k \geq 2$ is a constant integer, $w_1, w_2, \cdots, w_{k+1}$ are non-negative constant integers, and $W_i = \sum_{j\in[i]} w_j$ for $i \in [k]$ with $W_0 = 0$. By slightly abusing the notation, we may use Q_{apath} to refer to both the class of augmented path queries, or a specific query in this class.

Intuitively, an augmented path query is similar to a path query, but for each $i \in [k+1]$, there are w_i additional relations only containing the single attribute A_i. Hence, in the size of a synthetic witness, the number of tuples in these augmented relations, i.e., the size of the active domain of all attributes, should also be taken into consideration.

Theorem 5.10. SSW *is not poly-time solvable for any augmented path query* Q_{apath}*, unless* $\mathsf{P} = \mathsf{NP}$.

Our hardness proof of SSW for Q_{apath} exactly follows that of Q_{path}. We start with the same reduction from the vertex cover and build the set of tuples S as **Step 1**. Lemma 5.3 and Lemma 5.4 hold for Q_{apath}. In **Step 2**, suppose a poly-time algorithm for Q_{apath} can compute an arbitrary solution D to (Q_{apath}, S). Implied by Lemma 5.7, Lemma 5.5 also holds for Q_{apath} correspondingly in **Step 3**. The optimal vertex cover can be computed in **Step 4**.

5.3 Berge-acyclic SJFCQs

We start with an important structural property of Berge-acyclic SJFCQs below:

Definition 5.11 (Free Path). In a SJFCQ Q, a *free path* is a sequence of distinct attributes $P = \langle A_1, A_2, \cdots, A_{k+1}\rangle$ from $\mathtt{attr}(Q)$, and a sequence of distinct relations $Y = \langle R_1, R_2, \cdots, R_k\rangle$ from $\mathtt{rels}(Q)$, such that:

- $A_1, A_{k+1} \in \mathsf{head}(Q)$ and for each $i \in \{2, 3, \cdots, k\}$, $A_i \in \mathtt{attr}(Q) - \mathsf{head}(Q)$;
- For each $i \in [k]$, $A_i, A_{i+1} \in \mathtt{attr}(R_i)$.

Lemma 5.12 ([22]). *All Berge-acyclic SJFCQs without the head-domination property have a free path.*

Overview. Given any Berge-acyclic query without the head-domination property Q, we first find a specific augmented path query Q_{apath}, based on Q. By Theorem 5.10, we know that SSW is not poly-time solvable for Q_{apath}, unless $\mathsf{P} = \mathsf{NP}$. Next, we present a polynomial time reduction from SSW over Q_{apath} to SSW over Q. To establish this reduction, we rely on two key properties that hold for Q. First, Q always contains a free path. Second, every relation in Q that is not part of this free path contains at most one attribute from the free path. These properties allow us to define the augmented path query Q_{apath} based on the free path, and further a polynomial-time reduction from SSW over Q_{apath} to SSW over Q. Since SSW is not poly-time solvable for Q_{apath} unless $\mathsf{P} = \mathsf{NP}$, the same hardness result extends to SSW over all Berge-acyclic SJFCQs without the head-domination property.

Poly-time reduction. Consider an arbitrary Berge-acyclic SJFCQ Q without the head-domination property. Let $P = \langle A_1, A_2, \cdots, A_{k+1}\rangle$ from $\mathtt{attr}(Q)$, and $Y = \langle R_1, R_2, \cdots, R_k\rangle$ from $\mathtt{rels}(Q)$ be the free path of Q. For Berge-acyclic SJFCQs (see Section 2), we know that $|\mathtt{attr}(R_h) \cap P| \leq 1$ for

each relation $R_h \in \texttt{rels}(Q) - Y$ and $\texttt{attr}(R_i) \cap \texttt{attr}(R_{i+1}) = \{A_{i+1}\}$ for each $i \in [k-1]$, by the definition of the free path. Let

$$Q_{\text{apath}}(A_1, A_{k+1}) :- \left(\bowtie_{i\in[k]} T_i(A_i, A_{i+1})\right) \bowtie \left(\bowtie_{R_h\in\texttt{rels}(Q)-Y, P\cap\texttt{attr}(R_h)\neq\emptyset} T_h(P \cap \texttt{attr}(R_h)\right),$$

be the specific augmented path from which we will reduce. Let $S_1 \subseteq \texttt{dom}(\texttt{head}(Q_{\text{apath}}))$ be the set of input tuples. We construct a set $S_2 \subseteq \texttt{dom}(\texttt{head}(Q))$ of tuples as follows. For each $t_1 \in S_1$, we add a tuple t_2 to S_2, such that $\pi_{A_1,A_{k+1}} t_2 = t_1$ and $\pi_A t_2 = *$, for each attribute $A \in \texttt{head}(Q) - \{A_1, A_{k+1}\}$ for a special character $*$. Let D_2 be any solution to $\texttt{SSW}(Q, S_2)$. We construct a database D_1 as follows. For each $i \in [k]$, we set $T_i = \pi_{A_i,A_{i+1}}(R_i)$. For each $i \in [k]$, we set $T_h = \pi_{A_i} R_i$ for all $T_h \in \texttt{rels}(Q_{\text{apath}})$ such that $\texttt{attr}(T_h) = \{A_i\}$, and $T_h = \pi_{A_{k+1}} R_k$ for all $T_h \in \texttt{rels}(Q_{\text{apath}})$ such that $\texttt{attr}(T_h) = \{A_{k+1}\}$. We show that D_1 is a synthetic witness for (Q_{apath}, S_1) and prove its optimality.

Lemma 5.13. *$Q_{\text{apath}}(D_1) = S_1$ and D_1 is a solution to $\texttt{SSW}(Q_{\text{apath}}, S_1)$.*

Proof. The proof of the first part of the lemma can be found in Appendix D.

We show the second part of the lemma; D_1 is a solution to $\texttt{SSW}(Q_{\text{apath}}, S_1)$. By contradiction, assume that there exists a synthetic witness $\hat{D}_1$ to (Q_{apath}, S_1), such that $|\hat{D}_1| < |D_1|$. We construct a database $\hat{D}_2$ such that $\hat{D}_2$ is a witness to (Q, S_2), and $|\hat{D}_2| < |D_2|$, which contradicts the optimality of D_2. For simplicity, let $\hat{T}_i, \hat{R}_i$ to denote $T_i^{\hat{D}_1}, R_i^{\hat{D}_2}$ respectively. Consider any $i \in [k]$. For each tuple $t \in \hat{T}_i$, we build a tuple t' such that $\pi_{A_i,A_{i+1}}(t') = t$ and $\pi_A(t') = *$, for each attribute $A \in \texttt{attr}(R_i) - \{A_i, A_{i+1}\}$, and add t' to $\hat{R}_i$. Consider an arbitrary relation $R_h \in \texttt{rels}(Q) - Y$. We distinguish two cases:

(1) $\texttt{attr}(R_h) \cap P = \emptyset$. In this case, we add one tuple $(*, *, \cdots, *)$ to $\hat{R}_h$.
(2) $\texttt{attr}(R_h) \cap P = \{A_j\}$, for some $A_j \in P$. In this case, for each value $a \in \texttt{adom}^{\hat{D}}(A_j)$, we add a tuple t_a with $\pi_{A_j} t_a = a$, and $\pi_A t_a = *$ for each attribute $A \in \texttt{attr}(R_h) - \{A_j\}$.

It is straightforward to see $Q(\hat{D}_2) = S_2$ since $Q_{\text{apath}}(\hat{D}_1) = S_1$. Next we show $|\hat{D}_2| < |D_2|$.

For each $i \in [k]$, we have $|\hat{R}_i| = |\hat{T}_i|$ (by construction) and $|R_i| \geq |T_i|$ (by definition). Consider an arbitrary relation $R_h \in \texttt{rels}(Q) - Y$. We distinguish two cases:

(1) $\texttt{attr}(R_h) \cap P = \emptyset$. In this case, we have $|\hat{R}_h| = 1$, by the construction of $\hat{D}_2$. Moreover, we have $|R_h| \geq 1$, since each relation should contain at least one tuple.
(2) $\texttt{attr}(R_h) \cap P = A_j$, for some $A_j \in P$. In this case, we have $|\hat{R}_h| = |\texttt{adom}^{\hat{D}_1}(A_j)|$, by the construction of $\hat{D}_2$. Moreover, we have $|R_h| \geq |\texttt{adom}^{D_1}(A_j)|$; otherwise, there exists a value $a \in \texttt{adom}^{D_1}(A_j)$ such that there exists no tuple $t \in R_h$ with $\pi_{A_j}(t) = a$. This implies that there exists a dangling tuple in D_2, which contradicts the optimality of D_2.

Putting everything together, we have: $|D_2| \geq |D_1| + |\{R_h \in \texttt{rels}(Q) : \texttt{attr}(R_h) \cap P = \emptyset\}|$ and $|\hat{D}_2| = |\hat{D}_1| + |\{R_h \in \texttt{rels}(Q) : \texttt{attr}(R_h) \cap P = \emptyset\}|$. As we assumed $|\hat{D}_1| < |D_1|$, we obtain $|\hat{D}_2| < |D_2|$, contradicting the optimality of D_2. □

Theorem 5.14. *For a Berge-acyclic SJFCQ Q, if Q has head-domination property, then* SSW *is poly-time solvable; otherwise* SSW *is not poly-time solvable, unless* P = NP.

5.4 Beyond Berge-acyclic SJFCQs

In Section 4, we showed that SSW is solvable in polynomial time for all conjunctive queries (SJFCQs) with the head-domination property. In Section 5, we established that for the remaining class of Berge-acyclic SJFCQs, SSW is not polynomial-time solvable unless P = NP. To demonstrate that at least one cyclic query exists for which SSW is intractable, we now analyze the hardness of a simple cyclic query that lacks the head-domination property.

Beyond Berge-acyclic SJFCQs, we show the hardness of SSW on the pyramid query:

$$Q_{\text{pyramid}}(A,B,C) :- R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(C,A) \bowtie R_4(A,F) \bowtie R_5(B,F) \bowtie R_6(C,F),$$

which is the simplest cyclic query without the head-domination property. Notably, the pyramid query was previously used in [22] to establish the inapproximability of SWP within a sub-logarithmic factor for cyclic SJFCQs without the head-domination property. We prove the hardness of SSW for Q_{pyramid} via a reduction from the vertex cover problem. Let $G(V,E)$, be the input graph to the vertex cover problem as before. We construct the set $S \subseteq \mathsf{dom}(A) \times \mathsf{dom}(B) \times \mathsf{dom}(C)$ of tuples as $S = \{(e_i, v_j, c_1) : e_i \in E, v_j \in e_i\} \cup \{(e_i, *, c_1) : e_i \in E\} \cup \{(e_i, v_j, c_2) : e_i \in E, v_j \in V\}$, where $*, c_1, c_2$ are special characters, and show that having a solution for $\mathsf{SSW}(Q_{\text{pyramid}}, S)$, we can find a minimum vertex cover for G. The idea behind the construction of S, intuitively, is to force the relations R_1, R_2, and R_3, to have as many tuples as possible, so that they become less important and enable us to focus on the subquery $Q(A,B) :- R_4(A,F) \bowtie R_5(B,F) \bowtie R_6(C,F)$, which is a Berge-acyclic query. Since the process is similar to what we showed in previous sections, we show all the proofs and details in Appendix D, and conclude with the following theorem:

Theorem 5.15. SSW *is not poly-time solvable for* Q_{pyramid}, *unless* P = NP.

6 Approximating SSW

In this section, we briefly discuss the approximation version of SSW. A minor modification of Algorithm 1 returns an approximation solution for the SSW with theoretical guarantees. More specifically, in Algorithm 1, instead of returning **TRUE** or **FALSE** (in lines 7 and 8), we return D for the input (Q,S). We analyze the approximation ratio of this algorithm. First, by the construction of D, it holds that $|D| \leq |S| \cdot |\mathtt{rels}(Q)|$. Let D^* be a solution for $\mathsf{SSW}(Q,S)$. It is known that a SJFCQ Q over a database with $|D^*|$ tuples can have at most $O(|D^*|^{\rho^\star(Q)})$ results, where $\rho^\star(Q)$ is the fractional edge covering number[4] of Q [2]. Hence, it follows that $|D^*|^{\rho^\star(Q)} \geq c \cdot |S|$ for some constant c. Finally, we have $\frac{|D|}{|D^*|} \leq \frac{|S| \cdot |\mathtt{rels}(Q)|}{c^{1/\rho^\star(Q)} |S|^{1/\rho^\star(Q)}} = O(|S|^{1-1/\rho^\star(Q)})$. On the other hand, we define $\beta = \max_{R_i \in \mathtt{rels}(Q)} |\pi_{\mathsf{head}(R_i)} S|$. As $|D^*| \geq \beta$, we have $\frac{|D|}{|D^*|} \leq \frac{|S| \cdot |\mathtt{rels}(Q)|}{\beta} = O\left(\frac{|S|}{\beta}\right)$. We conclude:

Theorem 6.1. *For an arbitrary SJFCQ* Q *and a set* $S \subseteq \mathsf{dom}(\mathsf{head}(Q))$ *of tuples such that* $\mathsf{ESW}(Q,S)$ *returns* **TRUE**, *there is an algorithm that can find an* $O(\min\{|S|^{1-1/\rho^\star(Q)}, \frac{|S|}{\beta}\})$*-approximated solution to* $\mathsf{SSW}(Q,S)$ *in polynomial time in terms of* $|S|$, *where* $\rho^\star(Q)$ *is the fractional edge covering number of* Q, *and* $\beta = \max_{R_i \in \mathtt{rels}(Q)} |\pi_{\mathsf{head}(R_i)} S|$.

Remark. Recall the equivalence between SSW for the matrix query Q_{matrix} and the weighted edge covering problem discussed in Section 1.3. Any approximation algorithm (or inapproximability result) for SSW over the matrix query directly translates to an approximation algorithm (or inapproximability result) for the weighted edge covering problem. Unfortunately, despite being studied for nearly 50 years, the weighted edge covering problem has no known non-trivial approximation algorithms or inapproximability results [41, 43].

Therefore, designing constant-factor or $O(\log(|S|))$ approximation algorithms (or proving non-trivial inapproximability results) for SSW over the matrix query would yield significant progress in graph theory by providing new approximation algorithms or inapproximability results for this long-standing open problem.

[4]For a SJFCQ 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 W of Q.

7 Conclusions and Future Work

In this paper, we study the problem of constructing minimal synthetic witnesses for SJFCQs to reproduce the desired output. It is still an open problem whether we can get a full dichotomy for the SSW over all CQs, or even the class of α-acyclic CQs. As we have seen, proving the hardness is not trivial even for the simplest matrix query, since it is equivalent to the weighted biclique covering problem [41, 43]. To the best of our knowledge, all the known NP-hardness proofs [30, 36, 44] for this classic problem are non-trivial and rely on the structure of a bipartite graph – in other words, on having 2 relations in our setting. We presented a new proof via a reduction from the vertex cover problem, which holds for an arbitrary constant k. The concepts introduced here may prove valuable for future research aimed at fully characterizing the problem's complexity. A major distinction between the definition of SSW problem with other related witness problems (for example, smallest witness problem SWP [22], and deletion propagation [27]) is that, in other problems, the optimum solution is always a subset of the database input. In contrast, for SSW, the infinite domain of the SSW problem constitutes the primary obstacle to showing the hardness results for more complex queries and fully capturing its complexity.

Finally, we highlight several interesting yet challenging open questions for future investigation:

- *Full Dichotomy of* SSW *for CQs.* An immediate next step is to develop a complete characterization of the hardness for all CQs without the head-domination property.
- *Better Approximation Algorithm.* It is intriguing to investigate whether the approximation factor of the θ-SSW problem can be improved for certain specific queries by leveraging recent advances in the weighted biclique covering problem.
- *Finite Domains.* Our results assume that the values for synthetic witnesses can be freely chosen from an infinite domain. What would change if we imposed constraints on the available domain values? The results of [34] for the more general view insertion problem suggest that this impacts the hardness of the ESW problem.
- *Compressing S by optimizing both Q and D.* One application of synthetic witnesses is data compression: instead of sending a table S, we can transmit Q along and a witness D that enables the receiver to reconstruct S. A natural generalization of SSW is to allow Q to be determined as part of the optimization process.

References

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

[2] Albert Atserias, Martin Grohe, and Dániel Marx. 2013. Size bounds and query plans for relational joins. *SIAM J. Comput.* 42, 4 (2013), 1737–1767. doi:10.1137/110859440

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

[4] Claude Berge. 1984. *Hypergraphs: combinatorics of finite sets*. Vol. 45. Elsevier.

[5] Carsten Binnig, Donald Kossmann, Eric Lo, and M. Tamer Özsu. 2007. QAGen: generating query-aware test databases. In *SIGMOD*, Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou (Eds.). 341–352. doi:10.1145/1247480.1247520

[6] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. 2002. On Propagation of Deletions and Annotations through Views. In *PODS*. 150–158. doi:10.1145/543613.543633

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

[8] Balder ten Cate and Victor Dalmau. 2022. Conjunctive queries: Unique characterizations and exact learnability. *TODS* 47, 4 (2022), 1–41. doi:10.1145/3559756

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

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

[11] Gao Cong, Wenfei Fan, and Floris Geerts. 2006. Annotation propagation revisited for key preserving views. In *CIKM*. 632–641. doi:10.1145/1183614.1183705

[12] Stephen A Cook. 2023. The complexity of theorem-proving procedures. In *Logic, automata, and computational complexity: The works of Stephen A. Cook*. ACM, 143–152. doi:10.1145/3588287

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

[14] Graham Cormode and Ke Yi. 2020. *Small summaries for big data*. Cambridge University Press. doi:10.1017/9781108769938

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

[16] Tomás Feder and Rajeev Motwani. 1991. Clique partitions, graph compression and speeding-up algorithms. In *STOC*. 123–133. doi:10.1145/103418.103424

[17] Cibele 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. doi:10.14778/2850583.2850592

[18] Cibele 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. doi:10.1145/3375395.3387647

[19] M. R. Garey, D. S. Johnson, and L. Stockmeyer. 1974. Some simplified NP-complete problems. (1974), 47–63. doi:10.1145/800119.803884

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

[21] M. Herschel and M. Hernandez. 2010. Explaining Missing Answers to SPJUA Queries. *PVLDB* 3, 1 (2010), 185–196. doi:10.14778/1920841.1920869

[22] Xiao Hu and Stavros Sintos. 2024. Finding Smallest Witnesses for Conjunctive Queries. In *ICDT*. doi:10.4230/LIPIcs.ICDT.2024.24

[23] Xiao Hu, Shouzhuo Sun, Shweta Patwa, Debmalya Panigrahi, and Sudeepa Roy. 2020. Aggregated deletion propagation for counting conjunctive query answers. *PVLDB* 14, 2 (2020), 228–240. doi:10.14778/3425879.3425892

[24] Xiao Hu and Ke Yi. 2016. Towards a worst-case I/O-Optimal algorithm for acyclic joins. In *PODS*. 135–150. doi:10.1145/2902251.2902292

[25] Batya Kenig, Pranay Mundra, Guna Prasaad, Babak Salimi, and Dan Suciu. 2020. Mining Approximate Acyclic Schemes from Relations. In *SIGMOD*. 297–312. doi:10.1145/3318464.3380573

[26] Batya Kenig and Nir Weinberger. 2023. Quantifying the Loss of Acyclic Join Dependencies. In *PODS*. 329–338. doi:10.1145/3584372.3588658

[27] Benny Kimelfeld, Jan Vondrák, and Ryan Williams. 2012. Maximizing Conjunctive Views in Deletion Propagation. *TODS* 37, 4 (2012), 24:1–24:37. doi:10.1145/1989284.1989308

[28] Benny Kimelfeld, Jan Vondrák, and David P Woodruff. 2013. Multi-tuple deletion propagation: Approximations and complexity. *PVLDB* 6, 13 (2013), 1558–1569. doi:10.14778/2536258.2536267

[29] Xi Liang, Stavros Sintos, Zechao Shang, and Sanjay Krishnan. 2021. Combining aggregation and sampling (nearly) optimally for approximate query processing. In *SIGMOD*. 1129–1141. doi:10.1145/3448016.3457277

[30] Xuemin Lin. 2000. On the computational complexity of edge concentration. *Discrete Applied Mathematics* 101, 1-3 (2000), 197–205. doi:10.1016/S0166-218X(99)00207-3

[31] Neha Makhija and Wolfgang Gatterbauer. 2023. A Unified Approach for Resilience and Causal Responsibility With Integer Linear Programming (ILP) and LP Relaxations. *Proc. ACM Manag. Data* 1, 4 (2023), 228:1–228:27. doi:10.1145/3626715

[32] 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. doi:10.1145/3651605

[33] Neha Makhija and Wolfgang Gatterbauer. 2024. A Unified and Practical Approach for Generalized Deletion Propagation. *arXiv preprint arXiv:2411.17603* (2024). doi:10.48550/arXiv.2411.17603

[34] Dongjing Miao, Zhipeng Cai, Xianmin Liu, and Jianzhong Li. 2016. On the Complexity of Insertion Propagation with Functional Dependency Constraints. In *COCOON*, Vol. 9797. 623–632. doi:10.1007/978-3-319-42634-1_50

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

[36] Barsha Mitra, Shamik Sural, Jaideep Vaidya, and Vijayalakshmi Atluri. 2016. A survey of role mining. *CSUR* 48, 4 (2016), 1–37. doi:10.1145/2871148

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

[38] Jeff M Phillips. 2017. Coresets and sketches. In *Handbook of discrete and computational geometry*. CRC press, 1269–1288. doi:10.48550/arXiv.1601.00617

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

[40] Evgeny Sherkhonov and Maarten Marx. 2017. Containment of acyclic conjunctive queries with negated atoms or arithmetic comparisons. *Inform. Process. Lett.* 120 (2017), 30–39. doi:10.1016/j.ipl.2016.12.005

[41] Tamás G Tarjan. 1975. Complexity of Lattice-Configurations. (1975).

[42] Javier Tuya, Claudio de la Riva, Maria Jose Suarez-Cabal, and Raquel Blanco. 2016. Coverage-aware test database reduction. *IEEE Transactions on Software Engineering* 42, 10 (2016), 941–959. doi:10.1109/TSE.2016.2519032

[43] Zsolt Tuza. 1984. Covering of graphs by complete bipartite subgraphs; complexity of 0–1 matrices. *Combinatorica* 4 (1984), 111–116. doi:10.1007/BF02579163

[44] Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, and Haibing Lu. 2009. Edge-rmp: Minimizing administrative assignments for role-based access control. *Journal of Computer Security* 17, 2 (2009), 211–235. doi:10.3233/JCS-2009-0341

[45] Jane Xu, Waley Zhang, Abdussalam Alawini, and Val Tannen. 2018. Provenance Analysis for Missing Answers and Integrity Repairs. *IEEE Data Engineering Bulletin* (2018), 39.

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

A Missing Material for Section 1

Multiple minimal synthetic witnesses. First, observe that for any query with existential attributes, there will be infinitely few minimal synthetic witnesses as we assume an infinite domain of values that can be used in synthetic witnesses. Thus, given a synthetic witness, we can construct new synthetic witnesses D of equal size by picking any value c in $\mathtt{adom}^D(A)$ for some attribute $A \in \mathtt{head}(Q) - \mathtt{attr}(Q)$ and replacing every A attribute value in D that is equal to c with a fresh value $c' \notin \mathtt{adom}(D)$. However, the fact that we can rename domain values for body attributes is not the only cause for multiple synthetic witnesses to exist. Consider the following query:

$$Q(A, C) :- R_1(A, B) \bowtie R_2(B, C).$$

For $S = \{(a, 1), (a, 2), (b, 2)\}$, there are two minimal synthetic witnesses:

$$\underline{D_1} : R_1^{D_1} = \{(a, x), (b, y)\} \qquad \underline{D_2} : R_1^{D_2} = \{(a, x), (a, y), (b, y)\}$$
$$R_2^{D_1} = \{(x, 1), (x, 2), (y, 2)\} \qquad R_2^{D_2} = \{(x, 1), (y, 2)\}$$

B Missing Material for Section 3

Proof of Lemma 3.2. From Lemma 3.1, we know that $S \subseteq Q(D)$. It suffices to show that if there exists a tuple $h \in Q(D)$ such that $h \notin S$ then $\mathtt{ESW}(Q, S) = \textbf{false}$. By contradiction, assume that $h \in Q(D) - S$. Let $\{h_i \in R_i : i \in [m]\}$ be a set of tuples such that $\pi_{\mathsf{head}(Q)}(\bowtie_{i\in[m]} h_i) = h$. For simplicity, whenever we write x_i for a tuple $x \in S$, we denote the tuple that our algorithm adds in relation R_i given the tuple x. As $h \in Q(D)$ and by the construction of D based on the tuples in S, it must be the case that there exists a set of tuples $T = \{t^{(1)}, \ldots, t^{(\ell)}\} \subseteq S$, such that for every index $i \in [m]$, there exists an index $j_i \in [\ell]$ such that $h_i = t_i^{(j_i)}$, $\bigcup_{i\in[m]}\{j_i\} = [\ell]$, and $\pi_{\mathsf{head}(Q)}(\bowtie_{i\in[m]} t_i^{(j_i)}) = h$. As $h \in Q(D)$ and $h \notin S$, we have $|T| \geq 2$. If not (and cardinality of T was 1) then $h \in S$ which is a contradiction. Since for every $t \in S$ we create the tuples $\{t_i \mid i \in [m]\}$ with a distinct value for the body attributes unique to t, we know that for any pair $i_1, i_2 \in [m]$ with $j_{i_1} \neq j_{i_2}$, it holds that $(\mathtt{attr}(R_{i_1}) \cap \mathtt{attr}(R_{i_2})) - \mathtt{head}(Q) = \emptyset$ as otherwise, $t_{i_1}^{(j_{i_1})}$ does not join with $t_{i_2}^{(j_{i_2})}$, since they would have different values on body attributes. If $j_{i_1} = j_{i_2}$, then again from $\pi_{\mathsf{head}(Q)}(\bowtie_{i\in[m]} t_i^{(j_i)}) = h$ follows that for every $A \in \mathtt{head}(R_{i_1}) \cap \mathtt{head}(R_{i_2})$, $\pi_A(t_{i_1}^{(j_{i_1})}) = \pi_A(t_{i_2}^{(j_{i_2})})$. Intuitively, any synthetic witness for $S \supseteq T$ must also create the tuple h. Assume a synthetic witness D' exists for (Q, S). For every $t^{(p)} \in T \subseteq S$, let $\hat{t}_i^{(p)}$ be a tuple in $R_i^{D'}$ such that $t^{(p)} = \pi_{\mathsf{head}(Q)}(\bowtie_{i\in[m]} \hat{t}_i^{(p)})$. Hence, by definition, for every $i \in [m]$, we have $\pi_{\mathsf{head}(R_i)}(h_i) = \pi_{\mathsf{head}(R_i)}(\hat{t}_i^{(j_i)})$. By the analysis above, $h = \pi_{\mathsf{head}(Q)}(\bowtie_{i\in[m]} \hat{t}_i^{(j_i)})$ and we get the contradiction $h \in Q(D')$. □

C Missing Material for Section 4

We start by giving the formal definition of α-acylic queries.

Definition C.1 (α-acyclic CQs [3, 15]). A CQ Q is acyclic if there exists a tree $\mathbb{T}$ (also called *join tree*) 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 $A \in \mathbb{A}$, the set of nodes containing A forms a connected subtree of $\mathbb{T}$.

Although in section 4, we showed an algorithm that can solve SSW for all SJFCQs with the head-domination property, motivated by some applications, the set S of input tuples itself may be stored as a result of a SJFCQ, and we may not have direct access to it. For example, a database D

is stored on a server, and the server needs to send the results of a head-dominant SJFCQ Q over D to a client. One way to perform this task is first to compute the set of the results $S = Q(D)$, and then get a synthetic witness $D^* = \texttt{SSW}(Q, S)$ by calling Algorithm 2 on S. However, it may be computationally expensive to compute the results of $Q(D)$, since $|Q(D)|$ can be polynomially larger than $|D|$. Next, we show that it is possible to build the smallest synthetic witness directly from D without computing $Q(D)$. As discussed later, for a large class of queries, this algorithm is considerably faster.

Instead of computing the join results, we remove all dangling tuples from D. Then, we start from an empty database $D^* = \emptyset$ and for any tuple $t \in R_i^D$, for all $R_i \in \texttt{rels}(Q)$, we build a tuple t', such that $\pi_{\texttt{head}(R_i)}(t') = \pi_{\texttt{head}(R_i)}(t)$, and $\pi_A(t') = *$, for all $A \in \texttt{attr}(R_i) - \texttt{head}(R_i)$ and add t' to $R_i^{D^*}$. Let $\texttt{ModifiedSSW}(Q, D)$, denote the resulting database D^* after running this algorithm, and $\texttt{EasySSW}(Q, Q(D))$ denote the output of the original algorithm described in 4. The pseudo-code for ModifiedSSW is given in Algorithm 3, and the pseudo-code for EasySSW is given in Algorithm 2. Note that in line 2 of Algorithm 3, we can use any known algorithm for removing dangling tuples, as it is discussed later.

Lemma C.2. *The results of* $\texttt{EasySSW}(Q, Q(D))$ *and* $\texttt{ModifiedSSW}(Q, D)$ *are exactly the same, for any SJFCQ* Q*, and any given database instance* D*.*

Proof. Let D_1 and D_2 denote the resulting outputs of $\texttt{EasySSW}(Q, S)$ and $\texttt{ModifiedSSW}(Q, S)$, respectively. Let $t \in R_i^{D_1}$ be and arbitrary tuple from an arbitrary relation $R_i^{D_1}$ for some relation $R_i \in \texttt{rels}(Q)$. By the definition of EasySSW, we know that there exists a tuple $p \in R_i^D$ such that $\pi_{\texttt{head}(R_i)}(p) = t$. It is easy to verify that p is a non-dangling tuple, and in ModifiedSSW we will not remove it, and hence, there exists a tuple $p' \in R_i^{D_2}$, such that $\pi_{\texttt{head}(R_i)}(p') = \pi_{\texttt{head}(R_i)}(p) = \pi_{\texttt{head}(R_i)}(t)$, and by the definition of EasySSW and ModifiedSSW, we know that $\pi_A(p') = \pi_A(t) = *$ for all $A \in \texttt{attr}(R_i) - \texttt{head}(R_i)$. Therefore, we have $p' = t$ and can deduce that $D_1 \subseteq D_2$.

For the other direction, let $t \in R_i^{D_2}$ be and arbitrary tuple from an arbitrary relation $R_i^{D_2}$ for some relation $R_i \in \texttt{rels}(Q)$. By the definition of ModifiedSSW, we know that there exists a tuple $p \in R_i^D$ such that $\pi_{\texttt{head}(R_i)}(p) = t$ and p is a non-dangling tuple, so, there exists a tuple $p' \in Q(D)$, such that $\pi_{\texttt{head}(R_i)}(p') = \pi_{\texttt{head}(R_i)}(t)$. Thus, by the definition of EasySSW, we know that there exists a tuple $t' \in R_i^{D_1}$, such that $\pi_{\texttt{head}(R_i)}(t') = \pi_{\texttt{head}(R_i)(p')}$ and hence $\pi_{\texttt{head}(R_i)}(t') = \pi_{\texttt{head}(R_i)}(t)$. We know that $\pi_A(t) = \pi_A(t') = *$ for all $A \in \texttt{attr}(R_i) - \texttt{head}(R_i)$. Therefore, we have $t = t'$ and can deduce that $D_2 \subseteq D_1$. □

By Theorem 4.3, we know that EasySSW, outputs a solution to $\texttt{SSW}(Q, Q(D))$ for any head-dominated SJFCQ Q, and by the Lemma C.2, we know that $\texttt{ModifiedSSW}(Q, D)$ outputs the same results. Hence, we obtain:

Theorem C.3. *For an arbitrary head-dominated SJFCQ* Q *and an instance* D, $\texttt{ModifiedSSW}(Q, D)$ *returns a solution to* $\texttt{SSW}(Q, Q(D))$ *in polynomial time.*

Time and space complexity. In the algorithm EasySSW, the most expensive part is to compute the result of the SJFCQ Q. For this part, we can use a known algorithm for reporting the results of the SJFCQ Q, and the total time complexity of EasySSW would be the same as for this known algorithm. In the algorithm ModifiedSSW, the most expensive step is to remove the dangling tuples. The main difference between EasySSW and ModifiedSSW, is that for the first one, we need to compute $Q(D)$, and for the latter, we just need to remove the dangling tuples from D. In many cases, removing dangling tuples is less expensive than reporting all the results. For instance, it is known that for all α-acyclic queries, using the traditional Yannakakis algorithm, we can remove all the dangling tuples in $O(|D|)$ time and space. After removing dangling tuples, we go through each relation and

add the corresponding tuple to the output instance. So, after the cleaning part, the algorithm spends $O(|D|)$ time and space. We can conclude with the following corollary.

Corollary C.4. *For an arbitrary α-acyclic SJFCQ Q with the head-domination property, and an instance D, $\texttt{ModifiedSSW}(Q, D)$ returns a solution to $\texttt{SSW}(Q, Q(D))$ in $O(|D|)$ time.*

Algorithm 3: ModifiedSSW(Q, D)

```
D* ← ∅;
D ← removeDangling(D) ;
foreach i ∈ [m] do
    foreach tuple t ∈ R_i^D do
        t' ← a tuple defined on attr(R_i) such that π_A(t') = π_A(t) for each attribute
          A ∈ head(R_i) and π_A(t') = ∗ for each attribute A ∈ attr(R_i) − head(R_i);
        R_i^{D*} ← R_i^{D*} ∪ {t'};
return D*;
```

D Missing Material for Section 5

Proof of Lemma 5.13. We show the first part of the lemma; $S_1 = Q_{\text{apath}}(D_1)$.

Direction $S_1 \subseteq Q_{\text{apath}}(D_1)$. Let $t_1 \in S_1$ be an arbitrary tuple. By definition, there exists a tuple $t_2 \in S_2$ such that $\pi_{A_1,A_{k+1}} t_2 = t_1$ and $\pi_A t_2 = *$, for each attribute $A \in \texttt{head}(Q) - \{A_1, A_{k+1}\}$. By definition, for every $i \in [k]$, there exists a tuple $\hat{t}_i \in R_i^{D_2}$, such that $t_2 = \pi_{A_1,A_{k+1}}(\bowtie_{i\in[k]} \hat{t}_i)$. It holds that, $\pi_{A_{i+1}}(\hat{t}_i) = \pi_{A_{i+1}}(\hat{t}_{i+1})$, $\pi_{A_1}\hat{t}_1 = \pi_{A_1} t_2 = \pi_{A_1} t_1$, and $\pi_{A_{k+1}}\hat{t}_1 = \pi_{A_{k+1}} t_2 = \pi_{A_{k+1}} t_1$. By construction, for each $i \in [k]$, we have $\pi_{A_i,A_{i+1}}\hat{t}_i \in T_i$, $\pi_{A_i}\hat{t}_i \in T_h$ for all $T_h \in \texttt{rels}(Q_{\text{apath}})$ such that $\texttt{attr}(T_h) = \{A_i\}$, and $\pi_{A_{k+1}}\hat{t}_k \in T_h$ for all $T_h \in \texttt{rels}(Q_{\text{apath}})$ such that $\texttt{attr}(T_h) = \{A_{k+1}\}$. Hence, $t_1 = \pi_{A_1,A_{k+1}}(\bowtie_{i\in[k]} (\pi_{A_i,A_{i+1}}(\hat{t}_i))) \in Q_{\text{apath}}(D_1)$.

Direction $S_1 \supseteq Q_{\text{apath}}(D_1)$. Let $t \in Q_{\text{apath}}(D_1)$ be an arbitrary tuple. By definition, there exists a set of tuples $\{t_i \in T_i : i \in [k]\}$ such that $t = \pi_{A_1,A_{k+1}}(\bowtie_{i\in[k]} t_i)$. By definition, for every $i \in [k]$, there exists a tuple $t'_i \in R_i$ such that $\pi_{A_i,A_{i+1}} t'_i = t_i$. As D_2 is a solution to $\texttt{SSW}(Q, S_2)$, every tuple in D_2 contributes to at least one full join result of Q. As Q is Berge-acyclic, $|\texttt{attr}(R_h) \cap P| \leq 1$ for each $R_h \in \texttt{rels}(Q) - Y$ and $|\texttt{attr}(R_i) \cap \texttt{attr}(R_{i+1})| = 1$ for $i \in [k-1]$. Hence, if for each $i \in [k]$ there exists a tuple $t_i^* \in R_i$, such that $\pi_{A_{i+1}}(t_i^*) = \pi_{A_{i+1}}(t_{i+1}^*)$ for $i \in [k-1]$, leading to $\bowtie_{i\in[k]} t_i^* \neq \emptyset$, then additional relations in $\texttt{rels}(Q) - Y$ do not prune $\bowtie_{i\in[k]} t_i^*$, constructing a tuple $\hat{t}^* \in Q(D_2) = S_2$ with $\pi_{A_1}(\hat{t}^*) = \pi_{A_1}(t_1^*)$, $\pi_{A_{k+1}}(\hat{t}^*) = \pi_{A_{k+1}}(t_k^*)$, and $\pi_A(\hat{t}^*) = *$ for each attribute $A \in \texttt{head}(Q) - \{A_1, A_{k+1}\}$. Let $t' = \bowtie_i t'_i$ (notice that $\bowtie_i t'_i \neq \emptyset$). By the discussion above, it follows that there exists a tuple $\hat{t} \in Q(D_2) = S_2$ such that $\pi_{A_1}\hat{t} = \pi_{A_1}(t'_1) = \pi_{A_1}(t)$, $\pi_{A_{k+1}}\hat{t} = \pi_{A_{k+1}}(t'_k) = \pi_{A_{k+1}}(t)$, and $\pi_A\hat{t} = *$ for any other attribute $A \in \texttt{head}(Q) - \{A_1, A_{k+1}\}$. By definition, there should be a tuple $\bar{t} \in S_1$ such that $\bar{t} = \pi_{A_1,A_{k+1}}\hat{t} = t$. □

Proof of Theorem 5.15. We show a reduction from the vertex cover problem to the SSW problem for Q_{pyramid}. Let $G(V, E)$ be the input graph with the vertex set $V = \{v_1, v_2, \cdots, v_n\}$ and the edge set $E = \{e_1, e_2, \cdots, e_m\}$. We construct the set $S \subseteq \texttt{dom}(A) \times \texttt{dom}(B) \times \texttt{dom}(C)$ of tuples as follows. We set $\texttt{dom}(B) = V \cup \{*\}$ for some special value $*$, $\texttt{dom}(A) = E$, $\texttt{dom}(C) = \{c_1, c_2\}$, and $S = \{(e_i, v_j, c_1) : e_i \in E, v_j \in e_i\} \cup \{(e_i, *, c_1) : e_i \in E\} \cup \{(e_i, v_j, c_2) : e_i \in E, v_j \in V\}$. Next, we show that having a solution for $\texttt{SSW}(Q_{\text{pyramid}}, S)$, we can find the vertex cover of G in polynomial time. The intuitive idea behind constructing S, as described, is to force any witness of Q_{pyramid} to

have many tuples in R_1, R_2, R_3, so that these relations become irrelevant and we can focus only on minimizing $|R_4| + |R_5| + |R_6|$. Let $D = \mathsf{SSW}(Q_{\text{pyramid}}, S)$ be an optimal witness of S.

Lemma D.1. *Any solution for* $\mathsf{SSW}(Q_{\text{pyramid}}, S)$ *such as D, has the following structure.*

(1) $R_1^D = \{(e_i, v_j) : e_i \in E, v_j \in V\} \cup \{(e_i, *) : e_i \in E\}$.
(2) $R_2^D = \{(v_i, c_1) : v_i \in V\} \cup \{(v_i, c_2) : v_i \in V\} \cup (*, c_1)$.
(3) $R_3^D = \{(c_1, e_i) : e_i \in E\} \cup \{(c_2, e_i) : e_i \in E\}$.

Proof. It is easy to verify that $\pi_{AB}(S) \subseteq R_1^D$, because otherwise, if there exists a tuple $t \in S$ such that $\pi_{AB}(t) \notin R_1^D$, then $t \notin Q_{\text{pyramid}}(D)$, which contradicts the fact $Q_{\text{pyramid}}(D) = S$. With the same argument we have $\pi_{BC}(S) \subseteq R_2^D$ and $\pi_{CA}(S) \subseteq R_3^D$. By the construction of S, we have $\pi_{AB}(S) = \mathsf{dom}(A) \times \mathsf{dom}(B)$, and $\pi_{BC}(S) = \mathsf{dom}(B) \times \mathsf{dom}(C) - (*, c_2)$, and $\pi_{CA}(S) = \mathsf{dom}(C) \times \mathsf{dom}(A)$. For the first and the third condition, since $\pi_{AB}(S)$ and $\pi_{CA}(S)$ contain all the possible tuples in $\mathsf{dom}(A) \times \mathsf{dom}(B)$ and $\mathsf{dom}(C) \times \mathsf{dom}(A)$ respectively, and we showed that $\pi_{AB}(S) \subseteq R_1^D$ and $\pi_{CA}(S) \subseteq R_3^D$, we can directly deduce that $R_1^D = \pi_{AB}(S) = \{(e_i, v_j) : e_i \in E, v_j \in V\} \cup \{(e_i, *) : e_i \in E\}$, and $R_3^D = \pi_{CA}(S) = \{(c_1, e_i) : e_i \in E\} \cup \{(c_2, e_i) : e_i \in E\}$, and the first and third condition follows.

For the second condition, since $\pi_{BC}(S)$ contains all the tuples in $\mathsf{dom}(B) \times \mathsf{dom}(C)$ except the single tuple $(*, c_2)$, we need to show that $(*, c_2) \notin R_2^D$. By the construction of S, there is no $t \in S$, such that $\pi_{BC}(t) = (*, c_2)$, and by definition we have $Q_{\text{pyramid}}(D) = S$. Thus, if R_2^D contains the tuple $(*, c_2)$, we can remove this tuple from it without changing the query results $Q_{\text{pyramid}}(D)$ and this contradicts the optimality of D. Hence, we have $R_2^D = \pi_{BC}(S) = \{(v_i, c_1) : v_i \in V\} \cup \{(v_i, c_2) : v_i \in V\} \cup (*, c_1)$. □

Let $F_1 = \{f \in \mathsf{dom}(F) : (c_1, f) \in R_6^D\}$, and $F_2 = \{f \in \mathsf{dom}(F) : (c_2, f) \in R_6^D\}$.

Lemma D.2. *Any optimal witness of S such as D, has the following structure.*

(1) $F_1 \cap F_2 = \emptyset$.
(2) $|F_2| = 1$.

Proof. For any vertex $v_i \in V$, there exists an edge $e_j \in E$, such that $v_i \notin e_j$, otherwise we could simply report v_i as the minimum vertex cover of G. It is easy to see that there is no $f_1 \in F_1$, such that $(e_j, f_1) \in R_4^D \wedge (v_i, f_1) \in R_5^D$, since if such f_1 exists, we will have $(e_j, v_i, c_1) \in S$, which contradicts the construction of S. Moreover, we know that $(e_j, v_i, c_2) \in S$, so there should exist a value $f_2 \in F_2$ such that $(e_j, f_2) \in R_4^D \wedge (v_i, f_2) \in R_5^D$. Therefore, for any $v_i \in V$, there exists a tuple $(v_i, f) \in R_5^D$ such that $f \in F_2 - F_1$. Let $T_B = \{(v_i, f) \in R_5^D : v_i \in V, f \in F_2 - F_1\}$, by the above argument we have that $|T_B| \geq |V|$. Similarly, for any edge $e_i \in E$, we know that there exists a $v_j \in V$, such that $v_j \notin e_i$, since G has at least 2 edges. Thus, with the same argument we can show that for any $e_i \in E$, there exists a value $f \in F_2 - F_1$, such that $(e_i, f) \in R_4^D$. Let $T_A = \{(e_i, f) \in R_4^D : e_i \in E, f \in F_2 - F_1\}$ and we have $T_A \geq |E|$. We do the following operation on D to build another database instance $\hat{D}$ such that $Q_{\text{pyramid}}(\hat{D}) = S$, and show that if any of (1) or (2) does not hold, we will have $|\hat{D} < D|$, which contradicts the optimality of D.

We start by setting $\hat{D} = D$. Then, we remove all the tuples $t \in T_A$ from $R_4^{\hat{D}}$, and all the tuples $t \in T_B$ from $R_5^{\hat{D}}$. Until this stage, we have removed at least $|V| + |E|$ tuples. Next, we add a special character $*$ to $\mathsf{dom}(F)$, and add the tuple $(e_i, *)$ to $R_4^{\hat{D}}$, for every edge $e_i \in E$, and add the tuple $(v_i, *)$ to $R_5^{\hat{D}}$, for every vertex $v_i \in V$. In this stage we added $|V| + |E|$ tuples to $\hat{D}$. We also add the tuple $(c_2, *)$ to $R_6^{\hat{D}}$. At last, we remove all the tuples $(c_2, f) : f \in F_2$. After all these operations, it is easy to verify that $Q_{\text{pyramid}}(\hat{D}) = Q_{\text{pyramid}}(D) = S$.

In the first step, we removed at least $|V| + |E|$ tuples, then, in the second step we added $|V| + |E|$ tuples. In the third step, we added the single tuple $(c_2, *)$ and removed at least $|F_2|$ tuples. So, overall

we have $|\hat{D}| \leq |D| - |V| - |E| + |V| + |E| + 1 - |F_2| = |D| + 1 - |F_2|$. Thus, if $|F_2| > 1$, we have $|\hat{D}| < |D|$, which contradicts the optimality of D. Therefore, we have $|F_2| \leq 1$, and since we showed earlier that $F_2 - F_1$ is not empty, we can deduce that $|F_2| = 1$ and $F_2 \cap F_1 = \emptyset$. □

Next, we show that $Q_{\mathsf{pyramid}}(D) = R_4^D \bowtie R_5^D \bowtie R_6^D$. Let $\hat{f}$ denote the only value in F_2. We have $(*, \hat{f}) \notin R_5^D$, since otherwise we can remove this tuple and reduce the size of D, because there exists no tuple $t \in S$ such that $\pi_{BC}(t) = (*, c_2)$. By reordering the relations, we have $Q_{\mathsf{pyramid}}(D) = Q_1 \bowtie Q_2$, where $Q_1 = R_4^D \bowtie R_5^D \bowtie R_6^D$ and $Q_2 = R_1^D \bowtie R_2^D \bowtie R_3^D$. Let t be any tuple in $\mathsf{dom}(A) \times \mathsf{dom}(B) \times \mathsf{dom}(C)$, such that $\pi_{BC}(t) \neq (*, c_2)$. By Lemma D.1, we can easily see that $t \in Q_2$. Moreover, we showed that there is no tuple $t \in Q_1$, such that $\pi_{BC}(t) = (*, c_2)$. Hence, we have $Q_1 \subseteq Q_2$ and so $Q_{\mathsf{pyramid}}(D) = Q_1(D) = R_4^D \bowtie R_5^D \bowtie R_6^D$.

Let $H_1 = R_4^D \ltimes F_1$ and $H_2 = R_5^D \ltimes F_1$ and $H_3 = R_6^D \ltimes F_1$, be the set of tuples in R_4^D, R_5^D and R_6^D that have a F value in F_1. By Lemma D.2, we know that for all tuples $t \in H_3$, $\pi_C(t) = c_1$, and we have $|H_3| = F_1$. Therefore, we have

$$|R_4^D| + |R_5^D| + |R_6^D| = (|H_1| + |E|) + (|H_2| + |V|) + (|H_3| + 1) = |H_1| + |H_2| + |F_1| + |E| + |V| + 1.$$

Moreover,

$$\begin{aligned} |D| &= |R_4^D| + |R_5^D| + |R_6^D| + |R_1^D| + |R_2^D| + |R_3^D| \\ &= (|H_1| + |H_2| + |F_1| + |E| + |V| + 1) + (|E|(1 + |V|) + (2(1 + |V|) - 1) + 2|E|). \\ &= |H_1| + |H_2| + |F_1| + w, \end{aligned}$$

where w is a number that only depends on the structure of the graph G.

It is easy to check that $H_1 \bowtie H_2 \bowtie H_3 = \hat{S}$. Let $\hat{Q} = \pi_{AB}(R_4(A, F) \bowtie R_5(B, F))$, and let $\hat{S} = \{t \in S : \pi_C(t) = c_1\}$. In Theorem 5.2, we showed that $\mathsf{SSW}(\hat{Q}, \hat{S})$ is not poly-time solvable unless $\mathsf{P} = \mathsf{NP}$. Now, we show that the database instance $\hat{D} = H_1 \cup H_2$, is an optimal solution to $\mathsf{SSW}(\hat{Q}, \hat{S})$, and hence $\mathsf{SSW}(Q_{\mathsf{pyramid}}, S)$ is not poly-time solvable, since otherwise, we can go through D and find H_1 and H_2 in polynomial time and have an optimal solution $\hat{D}$ for $\mathsf{SSW}(\hat{Q}, \hat{S})$ in poly-time. First, since $H_1 \bowtie H_2 \bowtie H_3 = \hat{S}$, the instance $\hat{D}$ is a witness for $\hat{S}$. In Lemma 5.3, we showed that in any witness of $\hat{S}$, the size of the active domain of F is at least $|V|$, thus we have $|F_1| \geq |V|$. Second, let D^* be the optimal witness for $\hat{S}$. In Lemma 5.6, we showed that in polynomial time we can change D^* such that it remains an optimal witness for $\hat{S}$ and the size of the active domain of F denoted by $\hat{F}$ is exactly $|V|$. Now, we show that $|\hat{D}| = |D^*|$. Since $\hat{D}$ is a witness for $\hat{S}$ and D^* is optimal we have $|D^*| \leq |\hat{D}| = |H_1| + |H_2|$. Also, we have,

$$\begin{aligned} |H_1| + |H_2| + |F_1| &\leq |R_5^{D^*}| + |R_6^{D^*}| + |\hat{F}| \\ &= |R_5^{D^*}| + |R_6^{D^*}| + |V|, \end{aligned}$$

and since $|F_1| \geq |V|$, we have $|R_5^{D^*}| + |R_6^{D^*}| \geq |H_1| + |H_2| = |\hat{D}|$. Thus, we have $|\hat{D}| = |D^*|$ and hence $\hat{D}$, is an optimal witness for $\hat{S}$. This implies that the optimal solution D for $\mathsf{SSW}(Q_{\mathsf{pyramid}}, S)$ can not be obtained in polynomial time unless $\mathsf{P} = \mathsf{NP}$. □

Received December 2024; revised February 2025; accepted March 2025