Multi-Tuple Deletion Propagation: Approximations and Complexity

Benny Kimelfeld
IBM Research–Almaden
San Jose, CA 95120, USA
kimelfeld@us.ibm.com

Jan Vondrák
IBM Research–Almaden
San Jose, CA 95120, USA
jvondrak@us.ibm.com

David P. Woodruff
IBM Research–Almaden
San Jose, CA 95120, USA
dpwoodru@us.ibm.com

ABSTRACT

This paper studies the computational complexity of the classic problem of deletion propagation in a relational database, where tuples are deleted from the base relations in order to realize a desired deletion of tuples from the view. Such an operation may result in a (sometimes unavoidable) side effect: deletion of additional tuples from the view, besides the intentionally deleted ones. The goal is to minimize the side effect. The complexity of this problem has been well studied in the case where only a single tuple is deleted from the view. However, only little is known within the more realistic scenario of multi-tuple deletion, which is the topic of this paper. The class of conjunctive queries (CQs) is among the most well studied in the literature, and we focus here on views defined by CQs that are self-join free (sjf-CQs).

Our main result is a trichotomy in complexity, classifying all sjf-CQs into three categories: those for which the problem is in polynomial time, those for which the problem is NP-hard but polynomial-time approximable (by a constant-factor), and those for which even an approximation (by any factor) is NP-hard to obtain. A corollary of this trichotomy is a dichotomy in the complexity of deciding whether a side-effect-free solution exists, in the multi-tuple case. We further extend the full classification to accommodate the presence of a constant upper bound on the number of view tuples to delete, and the presence of functional dependencies. Finally, we establish (positive and negative) complexity results on approximability for the dual problem of maximizing the number of view tuples surviving (rather than minimizing the side effect incurred in) the deletion propagation.

1. INTRODUCTION

The practical need to allow for, and sometimes restrict to, database accesses through views gives rise to the view update problem: properly translate an update operation on the view to an update of the source relations. The core problem is that of underspecification—an update (e.g., tuple deletion) on the view can be realized by multiple, possibly very different, updates on the source relations. Therefore, a great deal of research has been devoted to designing syntax, semantics and restrictions to enable database systems to expose interfaces for updates through views. For example, Keller [15] defines a collection of (five) criteria qualifying the coherence of update translators, notions of complement views have been proposed for specifying update disambiguation and inversion [2, 8], and the relational lenses [3, 4] have been proposed as a language where view definitions include clear update policies to begin with.

The work reported in this paper is in another line of research within the view-update problem: underspecification is allowed, and the goal is to translate the update with as little as possible side effect [5–7, 9, 13, 16, 17]. A motivating scenario is in the context of database debugging: the user points out wrong tuples (false positives) or missing tuples (false negatives) in the result, and the proposed updated database serves as a suggestion of eliminating the errors while minimizing the side effect (e.g., on other results). Kimelfeld et al. [17] describe a use case of this motivation in the context of information extraction.

Our focus here is on the special case of deletion propagation. There, we are given undesired tuples in the view, which is defined by a monotonic query, and the goal is to delete tuples (facts) from the base relations, so that the undesired tuples are no longer in the view. The resulting database is called a solution. The side effect is the set of view tuples that are not among the undesired ones, and yet, are also deleted due to the deletion from the base relations. If there is no side effect (i.e., only the undesired tuples are deleted from the view), then the solution is side-effect free. However, it is possible that no side-effect-free solution exists. Hence, the task is relaxed to that of minimizing the side effect [5–7, 17]: delete tuples from the base relations so that the undesired tuples disappear from the view and the side effect is of minimal cardinality. Such a solution is said to be optimal. Note that the measure of quality we adopt, the view side effect, is different from the source side effect [5–7], where an optimal solution is one with a minimal number of missing facts.

This paper continues the research on the theoretical computational complexity of deletion propagation for views defined by conjunctive queries [5–7, 16, 17]. The ultimate goal is, naturally, to gain the insights needed for designing efficient solutions with quality guarantees. With the exception of Cong et al. [6], the past research has been restricted to the special case where only a single tuple is deleted from the view. Only a little is known about the computational
complexity in the (usually more realistic) case of multi-tuple deletion, which is the topic of this paper. We first shortly review the work on the single-tuple case. Later in this section, we relate the work here to the results of Cong et al. [6] on multi-tuple deletion.

Buneman et al. [5] showed NP-completeness of deciding on the existence of a side-effect-free solution (hence, NP-hardness of finding an optimal or approximately optimal solution) when the view is defined by the following self-join-free conjunctive query (abbrev. sjf-CQ here).

\[ Q_2'(y_1, y_2) : = R_1(x, y_1), R_2(x, y_2) \] (1)

Kimelfeld et al. [17] explore the space of all views defined by sjf-CQs and prove a dichotomy in the (data) complexity of single-tuple deletion propagation for views defined in that space. They do so by defining the property of head domination of a CQ, and show that this property fully characterizes the views with a tractable deletion propagation. More precisely, they prove the following dichotomy in complexity for the case where the view is defined by an sjf-CQ Q. If Q has head domination, then finding an optimal solution is in polynomial time; otherwise, it is NP-hard to find an optimal solution, and in fact any (finite) approximation thereof. Cong et al. [6] considered the complexity of the problem in the case of key preserving views. Later, Kimelfeld [16] generalized the results of Cong et al. [6], in the case of sjf-CQs, to a dichotomy in complexity that takes general functional dependencies into account.

We now proceed to describing the results we establish in this paper. We first show that the previously established dichotomy results for the single-tuple deletion [16,17] no longer hold in the multi-tuple case. Instead, we prove a trichotomy in complexity. More precisely, we show a classification of the sjf-CQs into the following three categories.

1. Those where finding an optimal solution is in polynomial time.
2. Those where an optimal solution is NP-hard to obtain, but for some constant k, a k-optimal solution (i.e., the side effect is at most k times the minimum) can be found in polynomial time.
3. Those where finding an α-optimal solution is NP-hard for all positive constants/variables α (since deciding on the existence of a side-effect-free solution is already NP-hard).

Although the dichotomy of [17] no longer holds in the multi-tuple case, our results do imply that, in some strong sense, the notion of head domination is robust enough to carry over to the general case. In particular, we parameterize their head domination into level-k head domination, for an integer k ≥ 1 (exact definitions are in Section 2). We then show that when k = 1, we are in Case 1 above. And when k > 1, we are in Case 2. Consequently, Case 3 captures exactly those sjf-CQs that are hard (to approximate) already in the single-tuple case (namely, no head domination).

As part of our proof of the trichotomy, we define a generalization of the minimal hitting-set problem, the focused hitting-set problem, where the goal is to hit all the sets in one collection while hitting as few as possible sets in another collection. We show how to reduce deletion propagation with level-k head domination to that problem, where each set has k elements. We present a k-approximation algorithm for that variant of focused hitting set, through rounded linear programming.

Cong et al. [6] studied complexity aspects of multi-tuple deletion. Their variant of deletion propagation is different from ours—they aim at minimizing the side effect, and at the same time, do so using the minimal number of deleted source tuples. They show that the problem is NP-hard under combined (query-and-data) complexity when the view is defined by a CQ, even if the CQ preserves keys (i.e., the head variables include a key for each relation symbol involved in the query). Our results here show that finding an optimal solution can be NP-hard already under data complexity, and even if we do not impose any requirement on the source side effect. In fact, as mentioned earlier, our results show precisely for which sjf-CQs the problem is in polynomial time, hard but approximable, and inapproximable.

In addition to the above trichotomy in complexity, we study additional variants of the problem. In particular, we consider the bounded deletion variant of our problem, where the number of deleted tuples is bounded by some constant. We show that, unlike the unbounded case, the dichotomy of Kimelfeld et al. [17] continues to hold. We also generalize the results described thus far to take functional dependencies into account.

The negative part of our results indicates that, quite often, reducing the side effect in deletion propagation is theoretically intractable if we desire approximation guarantees w.r.t. the smallest possible side effect. A natural question is then whether there are efficient algorithms for other types of quality guarantees. For that, we consider the maximization variant of our problem, where the goal is to maximize the remaining view rather than to minimize the side effect. The two versions, side-effect minimization and view maximization, are the same if only optimal solutions are considered. However, the two imply different notions of approximation guarantees, and consequently, involve different complexities. A known polynomial-time approximation for sjf-CQs guarantees the factor h, which is the minimal number of atoms that cover all the head variables. In fact, there is always a solution (obtainable through a simple algorithm due to Buneman et al. [5]) that retains at least 1/h of the non-deleted tuples [17]. In contrast, here we show that in the case of multi-tuple deletion, such a solution does not necessarily exist. We further show that for unbounded deletion, a very general condition implies that no polynomial-time approximation algorithm guarantees any constant approximation factor, under a reasonable (yet sub-standard) complexity assumption made by Ambühl et al. [1]. Nevertheless, we establish positive results for maximization in the case of bounded deletion.

The paper is organized as follows. In Section 2 we give preliminary definitions and background. The focused hitting-set problem is introduced in Section 3, where we also give an approximation algorithm. We define leveled head domination in Section 4. In Section 5 we give the complexity results for minimizing the side effect. The maximization variant of the problem and the incorporation of functional dependencies are discussed in Section 6. Finally, we make concluding remarks in Section 7.

2. FORMAL SETTING

In this section we present the formal setting we will build upon throughout the paper.
2.1 Schemas and Instances

We fix an infinite set \( \text{Const} \) of constants. We denote constants by lowercase letters from the beginning of the Latin alphabet (e.g., \( a, b \) and \( c \)). A \textit{schema} is a finite sequence \( \mathbf{R} = (R_1, \ldots, R_m) \) of distinct relation symbols, where each \( R_i \) has an arity \( r_i > 0 \). An \textit{instance} \( I \) (over \( \mathbf{R} \)) is a sequence \( (R'_1, \ldots, R'_n) \), such that each \( R'_i \) is a finite relation of arity \( r_i \) over \text{Const} (i.e., \( R'_i \) is a finite subset of \( \text{Const}^{r_i} \)). If \( c \in \text{Const} \), then \( R_i(c) \) is called a \textit{fact}, and it is a fact of the instance \( I \) if \( c \in R'_i \). Notationally, we view an instance as the set of its facts. For example, we may write \( R(c) \in I \) to say that \( c \) is in \( I \). As another example, \( J \subseteq I \) means that \( J \) is a \textit{subinstance} of \( I \), that is, \( R'_i \subseteq R_i \) for all \( i = 1, \ldots, m \).

2.2 Conjunctive Queries

We fix an infinite set \( \text{Var} \) of \textit{variables}. We assume that \( \text{Var} \) and \( \text{Const} \) are disjoint sets. We denote variables by lowercase letters from the end of the Latin alphabet (e.g., \( x, y \) and \( z \)). We use the Datalog style for denoting a conjunctive query (abbrev. CQ); that is, a CQ over a schema \( \mathbf{R} \) is an expression of the form \( Q(y) : = \Psi(x, y, c) \), where \( x \) and \( y \) are disjoint tuples of variables (from \( \text{Var} \)), \( c \) is a tuple of constants (from \( \text{Const} \)), and \( \Psi(x, y, c) \) is a conjunction of atomic formulas \( \varphi(x, y, c) \) over \( R \); an atomic formula is also called an \textit{atom}. We may write just \( Q(y) \), or even just \( Q \), if \( \Psi(x, y, c) \) is irrelevant. We denote by \( \Psi(x, y, c) \) the set of all \( Q \) of atoms of \( Q \). We usually write \( \Psi(x, y, c) \) by simply listing the atoms of \( Q \). We require every variable of \( y \) to occur at least once in \( \Psi(x, y, c) \). The \textit{arity} of \( Q \), denoted \( \text{arity}(Q) \), is the length of the tuple \( y \).

When we mention a CQ \( Q \), we usually avoid specifying the underlying schema \( \mathbf{R} \), and rather assume that this schema is the one that consists of the relation symbols that appear in \( Q \) (and each symbol has the arity it takes in \( Q \)). When we want to refer to that schema, we denote it by \( \text{schema}(Q) \).

Let \( Q(y) : = \Psi(x, y, c) \) be a CQ. A variable in \( x \) is called an \textit{existential} variable, and a variable in \( y \) is called a \textit{head} variable. We use \( \text{Var}_2(Q) \) and \( \text{Var}_h(Q) \) to denote the sets of existential variables and head variables of \( Q \), respectively. Similarly, if \( \varphi \) is an atom of \( Q \), then \( \text{Var}_2(\varphi) \) and \( \text{Var}_h(\varphi) \) denote the sets of existential and head variables, respectively, that occur in \( \varphi \). We denote by \( \text{Var}(Q) \) and \( \text{Var}(\varphi) \) the sets of all variables that occur in \( Q \) and \( \varphi \), respectively (that is, the unions \( \text{Var}_2(Q) \cup \text{Var}_h(Q) \) and \( \text{Var}_2(\varphi) \cup \text{Var}_h(\varphi) \), respectively).

The focus of this paper is on CQs that are \textit{self-join free}, which means that each relation symbol occurs at most once in the CQ. We use sjf\text{-CQ} as an abbreviation of \textit{self-join-free CQ}. If \( R \) is a relation symbol that occurs in the sjf\text{-CQ} \( Q(y) : = \Psi(x, y, c) \), then the unique atom over \( R \) is denoted by \( \varphi_R(x, y, c) \) or just \( \varphi_R \). Similarly, \( R(x, y) \) denotes the relation symbol of the atom \( \varphi \).

Example 2.1. We give here examples of sjf\text{-CQs} that we later reference. First, consider the CQ \( Q_2^1 \) in (1). The atoms of \( Q_2^1 \) are \( \varphi_1 = R_1(x, y_1) \) and \( \varphi_2 = R_2(x, y_2) \). In the examples of this article, \( R_i \) and \( R_j \) are assumed to be different symbols when \( i \neq j \). In particular, \text{schema}(Q_2^1) consists of two distinct binary relation symbols: \( R_1 \) and \( R_2 \). Hence, \( Q_2^1 \) has no self joins (but it would have a self join if we replaced the symbol \( R_2 \) with \( R_1 \)). There is only one existential variable in \( Q_2^1 \), namely \( x \), and the two head variables are \( y_1 \) and \( y_2 \). Hence \( \text{Var}_2(Q_2^1) = \{ x \} \) and \( \text{Var}_h(Q_2^1) = \{ y_1, y_2 \} \). Furthermore, \( \text{Var}_2(\varphi_1) = \{ x \} \) and \( \text{Var}_h(\varphi_1) = \{ y_1 \} \).

We will frequently refer to the following special sjf\text{-CQ}:

\[
Q_2^1(y_1, y_2) : = R_1(y_1), R_2(y_2)
\]

More generally, for a natural number \( k \):

\[
Q_k^1(y_1, \ldots, y_k) : = R_1(y_1), \ldots, R_k(y_k)
\]

That is, \( Q_k^1 \) is simply the Cartesian product of \( k \) distinct unary relations. Observe that \( Q_k^1 \) has no existential variables.

Finally, our running example for this section is the CQ \( Q_0 \), defined as follows.

\[
Q_0(y_1, y_2, y_3, y_4) : = R_1(x_1, y_1), R_2(x_1, y_2), R_3(x_2, y_1), R_4(x_3, y_1, y_2), R_5(x_2, x_3), R_6(y_1, y_4)
\]

The CQs in the examples of this paper do not have constants. Nevertheless, all the definitions and results have no restriction on constants. □


2.3 Deletion Propagation

The focus of this paper is on (but not restricted to) the problem of minimizing the \textit{view side effect} [5] when propagating the deletion of multiple answers (a.k.a. \textit{group deletion} [6]) back to the source relations. Formally, for a CQ \( Q \) the problem MinVSE(\( Q \)) is defined as follows. The input consists of an instance \( I \) over schema(\( Q \)), and a set \( A \subseteq \text{schema}(I) \) of answers. A solution \( (I, A) \) is an instance \( J \subseteq I \) such that \( Q \) is \( J \) and \( A = \emptyset \). The side effect (of \( J \), denoted \text{seff}(J) \), is the set \( \text{seff}(J) \setminus (A \cup Q(J)) \), that is, the set of answers that are deleted in addition to \( A \). (Note that the notation \text{seff}(J) assumes that \( Q \), \( I \) and \( A \) are known from the context, which will be the case whenever we will use this notation.) The goal is to find an \textit{optimal} solution, which is a solution \( J \) that minimizes the side effect; that is, \( J \) is such that \( \text{seff}(J) \leq \text{seff}(K) \) for all solutions \( K \).

As we discuss later, finding an optimal solution for the problem MinVSE(\( Q \)) may be intractable. Often, though, we can settle for \textit{approximations}, which are defined as follows. For a number (or numeric function) \( \alpha \geq 1 \), a solution \( J \) is said to be \( \alpha\text{-optimal} \) if \( \lim_{\alpha \to \infty} \text{seff}(J) \leq \alpha \cdot \text{seff}(K) \) for all solutions \( K \).
A smallest hitting set for \( C \) is \( \{0\} \), but this set hits all 4 sets in \( D \). A most focused hitting set is \( \{13, 23, 33\} \) (marked by the grey circles in Figure 1), which hits only one set in \( D \). Finally, \( \{12, 22, 32\} \) is a 2-optimal hitting set, as it hits (no more than) twice the minimum number of sets in \( D \).

Note that \( \text{FcsHit} \) generalizes \( \text{MinHit} \) through the following approximation-preserving reduction: from the input \((X, C, \alpha)\) for \( \text{MinHit} \) generate the input \((X, C, D)\) for \( \text{FcsHit} \) with \( D \) being the set of all singletons \( \{x\} \) where \( x \in X \).

Let \( k \) be a natural number. The problems \( \text{MinHit}(k) \) and \( \text{FcsHit}(k) \) are similar to \( \text{MinHit} \) and \( \text{FcsHit} \), respectively, except that now we assume that the size of each set in \( C \) or \( D \) is at most \( k \).

A special case of \( \text{MinHit}(k) \) is vertex cover in a \( k \)-uniform \( k \)-partite hypergraph: this is a vertex cover problem where the vertex set \( X \) is partitioned into \( k \) pairwise-disjoint sets \( s_1, \ldots, s_k \) such that each set (hyperedge) in \( C \) has exactly one element from each \( s_i \). The focused version of this problem is similarly a special case of \( \text{FcsHit}(k) \) where, again, each set in \( C \cup D \) has exactly one element from each \( s_i \). We denote these problems by \( \text{MinUVC}(k) \) and \( \text{FcsUVC}(k) \), respectively.

### 3.1 Approximation Algorithm

As well known, \( \text{MinHit}(k) \) can be \( k \)-approximated in polynomial time. Such an approximation is achieved by greedily selecting pairwise-disjoint sets from \( C \) until the union of these sets is a hitting set, which is taken as the solution. This simple greedy approach fails when the sets of \( C \) are weighted. Moreover, there does not seem to be any corresponding greedy approach to \( k \)-approximate \( \text{FcsHit}(k) \); one explanation of that is in the fact that focused hitting set can encode weighted hitting set. So, in the proof of the following lemma we adopt an approach that works for weighted vertex cover (which is a special case of weighted hitting set), namely linear programming with rounding [25]. As a result, we get that \( \text{FcsHit}(k) \) can also be \( k \)-approximated in polynomial time.

**Lemma 3.2.** For \( \text{FcsHit}(k) \), a \( k \)-optimal hitting set can be found in polynomial time.

**Proof.** Consider the input \((C, D)\) for \( \text{FcsHit}(k) \). We define the following linear program, with a variable \( y_s \) for each \( s \in D \) and \( x_e \) for each \( e \in \cup(C \cup D) \).

\[
\begin{align*}
\text{Minimize} & \quad \sum_{s \in D} y_s \quad \text{s.t.} \quad \forall s \in C \quad \sum_{e \in s} x_e &\geq 1 \quad (5) \\
& \quad \forall s \in D, e \in s \quad y_s \geq x_e \quad (6) \\
& \quad \forall e \in \cup(C \cup D) \quad x_e \geq 0 \quad (7)
\end{align*}
\]

Given a solution \( x, y \), we use the following rounding to construct a hitting set \( H \).

\[
H \equiv \{ e \in \cup C \mid x_e \geq 1/k \}
\]

Note that conditions (5) and (7), along with the fact that each \( s \in C \) satisfies \( |s| = k \), implies that \( H \) is indeed a hitting set (that is, for each \( s \in C \) there is \( e \in \cup C \) with \( x_e \geq 1/k \).
Let $H_{\text{opt}}$ be an optimal hitting set (i.e., a hitting set for $C$ that hits as few as possible members of $D$).

If $H'$ is a hitting set for $C$, then let $\mathbf{x}', \mathbf{y}'$ be the solution for the linear program assigning $x'_e$ one if $e \in H'$ or zero otherwise, and $y'_1 = 1$ if $s$ is hit by $H'$ or zero otherwise. It is easy to see that $\mathbf{x}', \mathbf{y}'$ is legal solution, and that $|H' \cap D| = \sum_{s \in D} y'_s$. It follows that $\sum_{s \in D} y_s \leq |H_{\text{opt}} \cap D|$. Let $\mathbf{y}$ be the solution obtained by replacing every $y_s$ with $1$ in case $s$ is hit by $H$. Observe that, due to (6) we have that $y_s \leq k y_s$ for all $s \in D$. Hence, we get the following:

$$|H \cap D| \leq \sum_{s \in D} y_s \leq \sum_{s \in D} k \cdot y_s = k \sum_{s \in D} y_s \leq k |H_{\text{opt}} \cap D|$$

Hence, our solution $H$ is $k$-optimal, as claimed. \[\square\]

4. LEVELED HEAD DOMINATION

In this section, we introduce the notion of level-$k$ head domination (where $k$ is a natural number). But first, we recall the definition of (ordinary) head domination [17].

Let $Q$ be a CQ. The existential-connectivity graph of $Q$, denoted $\mathcal{G}_{\exists}(Q)$, is the undirected graph that has atoms($Q$) as the set of nodes, and that has an edge $\{\varphi_1, \varphi_2\}$ whenever $\varphi_1$ and $\varphi_2$ have at least one existential variable in common (that is, $\varphi_1 \cap \varphi_2 \neq \emptyset$). If $P$ is a connected component of $\mathcal{G}_{\exists}(Q)$, then we denote by $\text{Var}_h(P)$ the set of all the head variables that occur in the atoms of $P$.

**Example 4.1.** Figure 2 shows the graph $\mathcal{G}_{\exists}(Q_0)$ for the CQ $Q_0$ in (4). Note that $R_1(x_2, y_1, y_2)$ and $R_4(x_3, y_1, y_2, y_3)$ do not share an edge, since they do not share existential variables (they share only head variables). The graph has three connected components (surrounded by dashed-edge polygons): $P_1$, $P_2$ and $P_3$. Observe that $\text{Var}_h(P_1) = \{y_1, y_2\}$, $\text{Var}_h(P_2) = \{y_1, y_2, y_3\}$ and $\text{Var}_h(P_3) = \{y_3, y_4\}$.

Following is the definition of the head-domination property of a CQ [17].

**Definition 4.2.** (Head Domination) A CQ $Q$ has head domination if there is a subset $\Phi$ of atoms($Q$), such that for every connected component $P$ of $\mathcal{G}_{\exists}(Q)$ there is an atom $\varphi \in \Phi$ with $\text{Var}_h(P) \subseteq \text{Var}(\varphi)$; in that case, we say that $Q$ is head dominated by $\Phi$. \[\square\]

Note that in the definition, the atom $\varphi$, which satisfies $\text{Var}_h(P) \subseteq \text{Var}(\varphi)$, is not required to be in $P$.

**Example 4.3.** Consider again the graph $\mathcal{G}_{\exists}(Q_0)$ in Figure 2. Recall that the connected components of $\mathcal{G}_{\exists}(Q_0)$ are $P_1$, $P_2$ and $P_3$. It holds that $\text{Var}_h(P_1) \subseteq \text{Var}(\varphi_{R_1})$, $\text{Var}_h(P_2) \subseteq \text{Var}(\varphi_{R_4})$ and $\text{Var}_h(P_3) \subseteq \text{Var}(\varphi_{R_6})$. (Recall that $\varphi_{R_i}$ denotes the atom with the relation symbol $R_i$). Therefore, $Q$ has head domination, and is head dominated by $\Phi' = \{\varphi_{R_1}, \varphi_{R_4}, \varphi_{R_6}\}$. Observe that $Q$ is also head dominated by $\Phi = \{\varphi_{R_4}, \varphi_{R_6}\}$.

An example of a CQ without head domination is $Q_2^*$ that we defined in (1). Indeed, $\mathcal{G}_{\exists}(Q_2^*)$ has exactly one connected component, its head variables are $y_1$ and $y_2$, and no atom of $Q$ contains both $y_1$ and $y_2$. \[\square\]

The next definition parameterizes head domination by the minimal number atoms needed for domination.

**Definition 4.4.** (Level-$k$ Head Domination) Let $k$ be a natural number. A CQ $Q$ with head domination has level-$k$ head domination if $k$ is the minimal cardinality of a set that head dominates $Q$.

**Example 4.5.** Continuing Example 4.3, the CQ $Q_0$ is head dominated by $\Phi$, which has two atoms. Since $Q_0$ is not head dominated by any single atom, we get that $Q_0$ has level-2 head domination. \[\square\]

Kimmelfeld et al. [17] proved the following dichotomy in the complexity of the problems $\text{MinVSE}_c(Q)$ and $\text{FreeVSE}_c(Q)$ (where the goal is to delete a single tuple). In the theorem we use “c” as the placeholder of “1” since we later refer to this theorem with an arbitrary $c$.

**Theorem 4.6.** [17] Let $Q$ be an sjf-CQ. The following hold for $c = 1$.

1. If $Q$ has head domination, then $\text{MinVSE}_c(Q)$ (hence, $\text{FreeVSE}_c(Q)$) can be solved in polynomial time.

2. If $Q$ has no head domination, then $\text{FreeVSE}_c(Q)$ is NP-complete; therefore, it is NP-hard to approximate $\text{MinVSE}_c(Q)$ by any finite factor.

**Comment 4.7.** Theorem 4.6 is weaker than the main theorem of Kimmelfeld et al. [17]. Their theorem adds the following. In Case 1, the problem is solvable by an extremely simple algorithm (called “trivial” there) that was originally proposed by Buneman et al. [5]. In Case 2, the ability to approximate is limited (or more formally the problem is APX-hard) even when using the maximization notion of approximation that we mentioned in Comment 2.2 (and use in Section 6.1).

In the next section we extend Theorem 4.6 to the deletion of multiple tuples, by studying the complexity of $\text{MinVSE}_c(Q)$ and $\text{FreeVSE}_c(Q)$. We will also investigate the complexity of $\text{MinVSE}_c(Q)$ and $\text{FreeVSE}_c(Q)$ for an arbitrary $c$.

5. COMPLEXITY RESULTS

In this section we give our main complexity results for the problems $\text{MinVSE}_c(Q)$ and $\text{FreeVSE}_c(Q)$. We begin with upper bounds.

5.1 Upper Bounds

To obtain positive complexity results (upper bounds), we will show how $\text{MinVSE}_c(Q)$ reduces to focused hitting set when $Q$ has head domination. Later, we will give complexity results based on this reduction.

Let $Q$ be an sjf-CQ with level-$k$ head domination. We now show how $\text{MinVSE}_c(Q)$ reduces to FksVSC($k$). We begin with a specific sjf-CQ $Q$, namely $Q^*_k$ (specified in (3)), where this reduction is straightforward. Given an instance $I$ over schema$(Q^*_k)$ and a set $A$ of answers to delete, the reduction is as follows. The ground set $X$ is (the set of all facts of) $I$.
The pairwise-disjoint sets $s_1, \ldots, s_k$ are simply the relations (i.e., fact sets) $R_1, \ldots, R_l$, respectively. We then define:

\[
C \overset{df}{=} \{ (R_1(a_1), \ldots, R_k(a_k)) \mid (a_1, \ldots, a_k) \in A \}
\]

\[
D \overset{df}{=} \{ (R_1(b_1), \ldots, R_k(b_k)) \mid (b_1, \ldots, b_k) \in Q_k(J) \setminus A \}
\]

It is then easy to see that from a hitting set $H$ for $C$ we obtain a solution $J$ by deleting from $I$ every fact in $H$; conversely, from a solution $J$ we obtain a hitting set for $C$ by taking $H = I \setminus J$. Most importantly, $H$ and $J$ have the same “quality”: $|H \cap D| = |\text{seff}(J)|$. Hence, we get an approximation-preserving reduction.

The sjf-CQ $Q = Q_k^\phi$ gives an extremely simple case of reducing MinVSE($Q$) to FcsUVC($k$). In the remainder of this section, we show that a reduction with similar guarantees exists for every sjf-CQ with level-$k$ head domination.

We begin with some notation. Let $Q(y)$ be a CQ, let $I$ be an instance of schema($Q$), and let $\mu$ be a match for $Q$ in $I$. If $\phi \in \text{atoms}(Q)$, then $\mu[\phi]$ denotes the restriction of $\mu$ to the head variables of $\phi$. If $\Phi$ is a subset of atoms($Q$), then $\mu[\Phi]$ denotes the set $\{ \mu[\phi] \mid \phi \in \Phi \}$. A fact $f$ over the relation symbol $R_\phi$ is consistent with $\mu[\phi]$ if $f$ can be obtained from $\phi$ by (1) replacing each existential variable with some constant, and (2) replacing each head variable $y$ with $\mu[\phi](y)$.

Let $Q$ be an sjf-CQ with level-$k$ head domination, and suppose that $Q$ is head dominated by $\Phi = \{ \varphi_1, \ldots, \varphi_k \}$. The reduction is depicted in Figure 3 as an algorithm named ToFcsHit($Q$). The ground set $X$ consists of all mappings $\mu[\varphi_i]$ where $\mu \in M(Q,I)$ and $1 \leq i \leq k$. The pairwise-disjoint sets $s_1, \ldots, s_k$ are defined by taking as $s_i$ the set of all $\mu[\varphi_i]$. Note that the $s_i$ are indeed pairwise disjoint, since different $\varphi_i$ have different sets of head variables (otherwise, $Q$ does not have level-$k$ head domination). The collection $C$ consists of all the sets $\mu[\Phi]$ where $\mu \in M(Q,I)$ and $\mu(y) \in A$, and the collection $D$ consists of all the sets $\mu[\Phi]$ where $\mu \in M(Q,I)$ and $\mu(y) \notin A$. A solution (hitting set) $H$ for the instance $(s_1, \ldots, s_k, C, D)$ of FcsUVC($k$) is constructed in line 6. From $H$ the reduction constructs a solution $J$ for MinVSE($Q$), as follows. Starting with $J = I$, for each mapping $\mu[\varphi_i] \in H$ the algorithm deletes from $J$ every fact $f$ over $\varphi_i$ such that $f$ is consistent with $\mu[\varphi_i]$.

Next, we prove the correctness of ToFcsHit($Q$)($I,A$). For that, we need the following lemma, which is proved similarly to the optimality of the “unirelation” algorithm for MinVSE$_1(Q)$ in the case of head domination [17].

**Lemma 5.1.** Let $Q$ be an $m$-ary sjf-CQ that is head dominated by a set $\Phi \subseteq \text{atoms}(Q)$. Let $J$ be an instance over schema($Q$) and $a \in \text{Const}^{\phi}$. The following are equivalent.

1. $a \in Q(J)$
2. For all $\varphi \in \Phi$ there is a fact $f_\varphi$ over $R_\varphi$, such that $f_\varphi$ is both $(Q,J)$-useful and $Q$-consistent with $a$. Next, we prove the following theorem.

**Theorem 5.2.** Let $Q$ be an sjf-CQ with level-$k$ head domination. Consider an execution of ToFcsHit($Q$)($I,A$). If FcsUVC($s_1, \ldots, s_k, C, D$) returns an $\alpha$-optimal hitting set, then the returned instance $J$ is an $\alpha$-optimal solution for $I$ and $A$.

**Proof.** Observe that every tuple $a \in Q(I)$ corresponds to a unique set in $C \cup D$; we denote this set by $s_a$. Suppose, w.l.o.g., that every fact in $I$ is $(Q,J)$-useful (since we can delete every fact that is not $(Q,J)$-useful without affecting the problem). With that assumption, the following are equivalent for a fact $f_\varphi$ over a relation symbol $R_\varphi$ (where $\varphi \in \text{atoms}(Q)$) and $a \in Q(I)$.

1. $f_\varphi$ is $Q$-consistent with $a$.
2. $f_\varphi$ is consistent with $\mu[\varphi]$ for some $\mu \in M(Q,I)$ that satisfies $\mu(y) = a$.

Then, Lemma 5.1 implies that for a tuple $a \in Q(I)$ if and only if for all $i = 1, \ldots, k$ there is a match $\mu \in M(Q,J)$ such that $\mu[\varphi_i]$ agrees with $a$. In other words, $a \in Q(J)$ if and only if we can find in $M(Q,J)$ matches to every member of $s_a$. This immediately implies that $J$ is indeed a solution for $I$ and $A$, since by hitting $C$ the algorithm eliminates at least one such match for each $a \in A$. By the same arguments we get that the solution $J$ is such that $|H \cap D| = |\text{seff}(J)|$.

We complete the proof by showing that for every solution $J$ for $I$ and $A$ there is a hitting set $H$ such that $|H \cap D| = |\text{seff}(J)|$. By using Lemma 5.1, we obtain the hitting set $H$ by taking every mapping $\mu[\varphi_i]$ (where $\mu \in M(Q,I)$) that is consistent with none of the tuples in $Q(J)$. Then $H$ hits exactly those sets $s$ that correspond to tuples in $Q(I) \setminus Q(J)$, which implies both that $H$ is a hitting set and that $|H \cap D| = |\text{seff}(J)|$, as claimed.

Next, we draw immediate corollaries from Lemma 3.2, in combination with Theorem 5.2. The first one states that MinVSE($Q$) is $k$-approximable whenever $Q$ has level-$k$ head domination.

**Corollary 5.3.** If $Q$ is an sjf-CQ with level-$k$ head domination, then MinVSE($Q$) is $k$-approximable in polynomial time.

As described above, Corollary 5.3 is obtained by reducing MinVSE($Q$) to FcsUVC($k$). Recall that our solution for FcsHit($k$) (which generalizes FcsUVC($k$)) is through the linear program we described in the proof of Lemma 3.2. In that program, each variable $y_k$ corresponds to a unique tuple.
in \(Q(I) \setminus A\), and vice versa. An important observation is that we could associate preferences to the tuples in \(Q(I) \setminus A\) (stating that survival of some tuples is more important than those of others) by associating with each \(y_e\) a weight \(w_s\), thereby setting the goal of minimizing \(\sum_{s \in C} w_s y_s\). Hence, we get a natural extension of the space of supported quality measures for solutions.

Corollary 5.3 gives, as a special case, a class of sjf-CQs for which MinVSE(\(Q\)) is solvable in polynomial time—as those with level-1 head domination (i.e., at least one atom contains all the head variables). Note that this upper bound could also be obtained through a straightforward argument that does not require Corollary 5.3.

**Corollary 5.4.** If \(Q\) is an sjf-CQ with level-1 head domination, then MinVSE(\(Q\)) is in polynomial time.

We later give a complementary lower bound, showing that the class of sjf-CQs with level-1 head domination is precisely that of sjf-CQs for which MinVSE(\(Q\)) is solvable in polynomial time (assuming P \(\neq\) NP).

An algorithm for MinVSE(\(Q\)), with any upper-bound guarantee on the approximation ratio, can be used for solving FreeVSE(\(Q\)): if there is a side-effect-free solution, the algorithm will necessarily find one. Hence, we get the following corollary, generalizing the corresponding part of Theorem 4.6 from MinVSE(\(Q\)) to MinVSE(\(Q\)).

**Corollary 5.5.** If \(Q\) is an sjf-CQ with head domination, then FreeVSE(\(Q\)) is in polynomial time.

### 5.2 Lower Bounds

Theorem 4.6 implies that if \(Q\) has no head domination, then it is NP-hard to solve MinVSE(\(Q\)) optimally. The following lemma states that this is also the case whenever \(Q\) has level-k head domination with \(k > 1\).

**Lemma 5.6.** If \(Q\) is an sjf-CQ with level-k head domination for \(k > 1\), then MinVSE(\(Q\)) is NP-hard.

Lemma 5.6 is a actually special case of a more general result that accounts for functional dependencies, which we discuss in Section 6.2.

Recall from Corollary 5.3 that for an sjf-CQ \(Q\) with level-k head domination, the problem MinVSE(\(Q\)) can be \(k\)-approximated in polynomial time. Whether this bound is tight for every \(Q\) is an open problem. However, we can show an infinite series of such \(Q\) where the lower bound is tight up to a constant factor (that does not depend on \(k\)). More specifically, based on a recent result by Gurumswami and Saket [14], the following theorem states that for \(k > 4\), the CQ \(Q^\alpha\) is such that it is NP-hard to approximate MinVSE(\(Q^\alpha\)) within some ratio linear in \(k\).

**Theorem 5.7.** Let \(k > 4\) be a natural number and let \(\epsilon > 0\) be a number. It is NP-hard to \((k/4 - \epsilon)\)-approximate MinVSE(\(Q^\alpha\)).

**Proof.** The proof is by a reduction from MinUVVC(\(k\)) (defined in Section 3). Gurumswami and Saket [14] showed that for all \(\epsilon > 0\), it is NP-hard to obtain a \((k/4 - \epsilon)\)-optimal hitting set. So consider an instance \((s_1, \ldots, s_k, C)\) of this problem. We define an instance \(I\) over schema(\(Q^\alpha\)) as follows. Each relation \(R_i\) contains all the values of \(s_i\) (as unary tuples) and, in addition, a set of \(N\) fresh (distinct) constants. (Note that each fresh constant occurs exactly once.) The value \(N\) will be determined later. The set \(A\) of answers to delete is obtained from \(C\) by straightforwardly translating each \(s \in C\) into a tuple of the Cartesian product \(X_{i=1}^k s_i\) that comprises the elements in \(s\).

Let \(H_{opt}\) be a minimal hitting set for \((s_1, \ldots, s_k, C)\). Let \(J_0\) be the solution that is obtained from \(H_{opt}\) by deleting every element of \(H_{opt}\) (from the proper relation). Let \(J\) be any solution for \(I\) and \(A\) (obtained through some algorithm). From \(J\) we obtain a hitting set \(J\) by selecting all the values of the \(s_i\) that are missing (i.e., have been deleted) from \(J\). We will show that if \(J\) is a good approximation w.r.t. \(J_0\), then \(J\) is a good approximation w.r.t. \(H_{opt}\). More precisely, we will prove the following for the right choice of \(N\) and some \(\delta > 0\) that depends only on \(\epsilon\).

\[
\frac{|\text{seff}(J)|}{|\text{seff}(J_0)|} < k/4 - \delta \quad \Rightarrow \quad \frac{|H_J|}{|H_{opt}|} < k/4 - \epsilon \quad (8)
\]

We first estimate the size of \(\text{seff}(J)\). Define \(M = \prod_{i=1}^k |s_i|\). Then \(\text{seff}(J)\) is the disjoint union of two sets:

- Tuples \(t\) that contain precisely one element in \(\bigcup_{i=1}^k s_i\). There are precisely \(|H_J| \cdot N^{k-1}\) such tuples.
- Tuples \(t\) that contain at least two elements in \(\bigcup_{i=1}^k s_i\). There are at most \(MN^{k-2}\) such tuples (since \(M\) bounds the number of choices of the elements from \(\bigcup_{i=1}^k s_i\)).

We conclude the following:

\[
|H_J| \cdot N^{k-1} \leq |\text{seff}(J)| \leq |H_J| \cdot N^{k-1} + MN^{k-2}
\]

And similarly (since \(J\) is arbitrary):

\[
|H_{opt}| \cdot N^{k-1} \leq |\text{seff}(J_0)| \leq |H_{opt}| \cdot N^{k-1} + MN^{k-2}
\]

So we get the following argument.

\[
\frac{k}{4} - \delta > \frac{|\text{seff}(J)|}{|\text{seff}(J_0)|} \geq \frac{|H_J|}{|H_{opt}| + M/N} = \frac{|H_J|}{|H_{opt}| + 1} \frac{1}{1 + M/N} \geq \frac{|H_J|}{|H_{opt}|} \frac{1}{1 + M/N} \Rightarrow \frac{|H_J|}{|H_{opt}|} < k/4 - \delta + kM/4N < k/4 - \delta + kM/4N
\]

So we obtain (8) by choosing \(\delta = 2\epsilon\) and \(N = 4kM/\epsilon\).  

### 5.3 Trichotomy in Complexity

The main result of this paper is the following theorem, stating a trichotomy in complexity for MinVSE(\(Q\)) over the sjf-CQs \(Q\). The theorem is obtained by combining Theorem 4.6, Corollaries 5.3, 5.4 and 5.5, and Lemma 5.6.

**Theorem 5.8 (Trichotomy).** Let \(Q\) be an sjf-CQ.

- If \(Q\) has level-1 head domination, then MinVSE(\(Q\)) is solvable in polynomial time.
- If \(Q\) has level-k head domination with \(k > 1\), then MinVSE(\(Q\)) is NP-hard but \(k\)-approximable in polynomial time; moreover in that case FreeVSE(\(Q\)) is in polynomial time.
- If \(Q\) has no head domination, then MinVSE(\(Q\)) is NP-complete (and approximating MinVSE(\(Q\)) by any finite factor is NP-hard).
Table 1: The complexity of MinVSE(Q) and FreeVSE(Q), and their bounded-deletion variants

<table>
<thead>
<tr>
<th>Problem</th>
<th>Level-1 h. d.</th>
<th>Level-k h. d. for k &gt; 1</th>
<th>No. h. d.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinVSE(Q) (optimal)</td>
<td>PTime</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>MinVSE(Q) (approx.)</td>
<td>PTime</td>
<td>PTime</td>
<td>NP-hard</td>
</tr>
<tr>
<td>FreeVSE(Q)</td>
<td>PTime</td>
<td>PTime</td>
<td>NP-complete</td>
</tr>
<tr>
<td>MinVSE_c(Q) (optimal)</td>
<td>PTime</td>
<td>PTime</td>
<td>NP-hard</td>
</tr>
<tr>
<td>MinVSE_c(Q) (approx.)</td>
<td>PTime</td>
<td>PTime</td>
<td>NP-hard</td>
</tr>
<tr>
<td>FreeVSE_c(Q)</td>
<td>PTime</td>
<td>PTime</td>
<td>NP-complete</td>
</tr>
</tbody>
</table>

Example 5.9. Consider the following three sjf-CQs.

\[ Q_1(y_1, y_2, y_3) : = R(y_1, y_2), S(y_2, y_3), T(y_1, y_2, y_3) \]
\[ Q_2(y_1, y_2, y_3) : = R(y_1, y_2), S(x, y_1), T(y_1, x, y_1) \]
\[ Q_3(y_1, y_2, y_3) : = R(y_1, y_2), S(x, y), T(y_1, y_2, x) \]

Observe that \( Q_1 \) has level-1 head domination, \( Q_2 \) has level-2 head domination, and \( Q_3 \) has no head domination. From Theorem 5.8 we then conclude the following. First, the problem MinVSE(Q_1) is solvable in polynomial time. Second, MinVSE(Q_2) is NP-hard but 2-approximable in polynomial time; moreover, FreeVSE(Q_2) is solvable in polynomial time. Third, the problem FreeVSE(Q_3) is NP-complete, and hence, MinVSE(Q_3) cannot be approximated in polynomial time for any finite constant, unless P = NP; moreover, these results hold already for the single-tuple variants: FreeVSE_1(Q_3) and MinVSE_1(Q_3).

5.4 Bounded Deletion

We now consider the complexity of MinVSE_c(Q), and give an immediate corollary of Theorem 5.2: in the case of head domination, MinVSE_c(Q) is solvable in polynomial time, and is even FPT for the parameter \( c \). Recall that FPT stands for fixed-parameter tractability [12], which means that when a parameter \( c \) is involved, the running time is bounded by a function of the form \( f(c) \cdot p(n) \), where \( f \) is a function and \( p \) is a polynomial over the size \( n \) of the input. Hence, this corollary extends the positive result of Theorem 4.6 for MinVSE_c(Q) from \( c = 1 \) to an arbitrary \( c \).

Corollary 5.10. If \( Q \) is an sjf-CQ with head domination and \( c \) is a natural number, then MinVSE_c(Q) is solvable (optimally) in polynomial time, and is even FPT for the parameter \( c \).

Proof. When calling ToFcsHit(Q(I, A) with \( |A| = c \), the generated instance of FcsUVC(k) has a set \( C \) with \( |C| \leq c \), where each member \( s \in C \) is of a constant size (i.e., \( k \) if \( Q \) has level-k head domination). But this instance of FcsUVC(k) has a straightforward polynomial-time solution: consider every possible set \( H' \) obtained by selecting one member of each set in \( C \), compute each \( |H' \cap D| \), and then select as \( H \) the \( H' \) with a minimal \( |H' \cap D| \). Observe that, since \( k \) is fixed, the number of considered sets \( H' \) is bounded by a function of \( c \) regardless of \( I \) and \( A \), and hence, we get that the problem is FPT.

Combining Theorem 4.6 and Corollary 5.10, we get the following.

Theorem 5.11. Theorem 4.6 holds true for every \( c \in \mathbb{N} \) (not just \( c = 1 \)).

Table 1 summarizes the complexity results of Theorems 5.8 and 5.11. Each entry corresponds to a problem (row) and a class of sjf-CQs (column). In the second and fifth rows, “PTime” refers to \( k \)-approximation, while “NP-hard” refers to every finite (constant or function) approximation.

6. Extensions

In this section we discuss two extensions of our complexity results. In the first extension we study the maximization variants of our problems, and in the second we consider the effect of functional dependencies.

6.1 Maximizing the View

In this section, we consider a deletion-propagation problem dual to MinVSE(Q), namely, maximizing the remaining view \( Q(J) \) rather than minimizing the side effect. Of course, in terms of seeking optimal solutions these problems are identical. However, in the case of a single-tuple deletion, the two problems have different complexities in terms of approximation [17].

Formally, we consider the problem MaxView(Q) that has the same input as MinVSE(Q), namely \( I \) and \( A \), except that the goal is to find a solution \( J \) (i.e., \( J \subseteq I \)) satisfying \( Q(J) \subseteq A = \emptyset \) with a maximal \( |Q(J)| \). In particular, given a number \( \alpha \) satisfying \( Q(J) \subseteq A = \emptyset \), a solution \( J \) is \( \alpha \)-optimal if \( |Q(J)| \geq |Q(K)|/\alpha \) for all solutions \( K \). For a natural number \( c \), the problem MaxView_c(Q) is the same as MaxView(Q), except that the input is restricted to \( |A| \leq c \).

Let \( Q \) be a CQ. For a natural number \( h \), we say that \( Q \) has \( h \)-coverage if \( Q \) has \( h \) atoms \( \varphi_1, \ldots, \varphi_h \), covering all the head variables, that is, \( \text{Vars}(Q) = \bigcup_{i=1}^{h} \text{Vars} \varphi_i \). Kimiliau et al. [17] proved the following.

Proposition 6.1. [17] If \( Q \) is an sjf-CQ with \( h \)-coverage, then one can find in polynomial time a solution \( J \) such that \( |Q(J)| \geq (|Q(I)| - 1)/h \). In particular, MaxView_1(Q) is \( h \)-approximable in polynomial time.

The question we explore here is whether Proposition 6.1 extends to MaxView(Q) and to MaxView_c(Q). In the reminder of this section we address the complexity aspect this question and show the following.

- Under a fairly general assumption on the sjf-CQ \( Q \) (in addition to a reasonable complexity assumption), it is intractable to approximate MaxView(Q) within any constant factor (Section 6.1.1).
- If \( Q \) is an sjf-CQ with \( h \)-coverage, then MaxView_c(Q) is (almost) \( h \)-approximable in polynomial time (Section 6.1.2).

But first, a complexity-independent question is whether there always exists a solution that retains at least \( 1/h \) of the non-deleted answers, or even any constant factor thereof. The following example shows a negative answer.

Example 6.2. Consider the CQ \( Q_2^c \) defined in (2), and let \( I^c \) be an instance over schema \( Q_2^c \), such that each of
the relations $R_1$ and $R_2$ contains the tuples consisting of the
constants $1, \ldots, n$. Then $Q^*_2(I^*)$ is the set \{1, \ldots, n\} ×
\{1, \ldots, n\}. Now suppose that $A$ is the set of all pairs \((i, j)\) ∈
$Q^*_2(I^*)$ such that $i \neq j$. Then there are $n^2 - n$ answers to
delete and $n$ remaining answers. Now, let $J$ be a solution.
Then $J$ cannot contain any $R_1(i)$ and $R_2(j)$ with $i \neq j$, and
hence, $Q^*_2(J)$ contains at most one answer. In particular,
at most $1/n$ of the non-deleted answers can survive in any solution.

6.1.1 Hardness of Approximation

The next theorem states a general condition (though not a
dichotomy) on an sjf-CQ $Q$, such that MAXVIEW($Q$) (without
any restriction on \(|A|\)) does not have a constant-factor
approximation in polynomial time, under reasonable (yet
somewhat non-standard) complexity assumptions. The condi-
tion is that $Q$ has two head variables that are not atomic
neighbors, where two variables $y_1$ and $y_2$ are atomic neigh-
bor if some atom of $Q$ contains both $y_1$ and $y_2$. The com-
plexity assumptions are due to Ambühl et al. [1] whose result
we use to prove our theorem.

**Theorem 6.3.** Let $Q$ be an sjf-CQ with a pair of head
variables that are not atomic neighbors. Moreover, assume
that for some $\epsilon > 0$, SAT does not have a probabilistic algo-

rithm that decides in $O(2^n)$ time whether a given instance of
size $n$ is satisfiable. Then no (randomized) polynomial-
time algorithm approximates MAXVIEW($Q$) within any con-
stant factor.

Next, we prove Theorem 6.3. The simplest example of
an sjf-CQ with a pair of head variables that are not atomic
neighbors is $Q^*_2$, and we begin by proving the theorem for
that CQ.

**Lemma 6.4.** The following hold.

1. MAXVIEW($Q^*_2$) (hence, MinVSE($Q^*_2$)) is NP-hard.
2. Theorem 6.3 is true for $Q = Q^*_2$.

**Proof.** We will show that MAXVIEW($Q^*_2$) is actually the
maximum edge biclique problem, which in turn is the fol-
lowing problem: given a bipartite graph $G$, find a complete
bipartite subgraph (biclique) of $G$ with a maximal number of
edges. Note that in a biclique the number of edges is simply the product of the sizes of the two sides. We encode
this problem as MAXVIEW($Q^*_2$), as follows. Suppose
that the input consists of a bipartite graph $G = (V_1, V_2, E)$. In
the instance $I$, the relations $R_1$ and $R_2$ contain the nodes of
$V_1$ and $V_2$ (as unary tuples), respectively. The set $A$ contains
every non-edge $(v_1, v_2) \in V_1 \times V_2$. Observe that a solution $J$
ecessarily induces a biclique $B_J$; moreover, $Q^*_2(J)$ is pre-
cisely the set of edges in $B_J$. The other direction is also true:
every biclique $B$ represents a solution $J_B$ such that
$Q^*_2(J_B)$ consists of the edges of $B$.

Peeters [23] proved that maximum edge biclique is NP-
hard; hence, we get Part 1. Ambühl et al. [1] showed in-
approximability by any constant factor for maximum edge
biclique, under the complexity assumption of Theorem 6.3; consequently, we get Part 2.

We can now prove Theorem 6.3.

**Proof.** (Theorem 6.3) We show a fairly straightforward
approximation-preserving reduction from MAXVIEW($Q^*_2$) to
MAXVIEW($Q^*_2$). Let $I^*$ and $A^*$ be input for MAXVIEW($Q^*_2$).
We construct the input $I$ and $A$ for MAXVIEW($Q$), as fol-
lowing. Let $y_1$ and $y_2$ be head variables of $Q$ that are not
atomic neighbors. For each pair $(a_1, a_2) \in Q^*_2(I^*)$, we
define the mapping $\lambda_{a_1, a_2}: \text{Var}(Q) \to \text{Const}$, as follows.

$$
\lambda_{a_1, a_2}(z) =
\begin{cases}
  a_1 & \text{if } z = y_1; \\
  a_2 & \text{if } z = y_2; \\
  c & \text{otherwise.}
\end{cases}
$$

Next, for each $(a_1, a_2) \in Q^*_2(I^*)$ and atom $\varphi$ of $Q$ we add
to $I$ every fact of the form $\lambda_{a_1, a_2}(\varphi)$. We also add to $A$
the tuple that is obtained from $y$ by applying $\lambda_{a_1, a_2}$. This
completes the reduction, and it is straightforward to show that
this reduction is indeed approximation preserving (building
on the fact that $y_1$ and $y_2$ are not atomic neighbors). In
particular, for each solution $J^*$ for $I^*$ and $A^*$ we can con-
struct a solution $J$ for $I$ and $A$ with $|\text{Var}(Q(J))| = |Q^*_2(J^*)|$ and
vice versa.

Theorem 5.8 states that MAXVIEW($Q$) is NP-hard if $Q$
has no level-1 head domination. It is known that a constant
lower bound (in addition to a constant upper bound) exists
on the polynomial-time approximability of MAXVIEW($Q$) in
Case 2.2. But it remains open whether MAXVIEW($Q$) has a constant-factor
approximation for any sjf-CQs $Q$ without level-1 head dom-
ination, such that $Q$ is not among those of Theorem 6.3.

6.1.2 Approximation for Bounded Deletion

In this section we consider the complexity of approximating
MAXVIEW($Q$). In that case, one can delete the unde-
sired answers one by one and, by applying Proposition 6.1,
still get a constant-factor approximation. Hence, we get the
following corollary.

**Corollary 6.5.** If $Q$ is an sjf-CQ with $h$-coverage, then
MAXVIEW($Q$) is $h$-approximable in polynomial time.

In the remainder of this section, we reduce the approxi-
mation factor $h^*$ of Corollary 6.5 to almost $h$ (specifically,
$h + \varepsilon$ for all $\epsilon > 0$) by using further insights into the prob-
lem MAXVIEW($Q$). We begin by describing a particularly
naive algorithm for MAXVIEW($Q$), which we call the unire-
lation algorithm, and which extends the algorithm originally
proposed by Buneman et al. [5].

Consider an sjf-CQ $Q(y)$, and let $I$ and $A$ be input for
MAXVIEW($Q$). For each atom $\varphi$ of $Q$, the solution $J_\varphi$ for $I$
and $A$ is obtained by removing from the relation $R^*_y$ every
fact that is $Q$-consistent with at least one of the tuples in
$A$. Recall that a fact $f$ of $R^*_y$ is $Q$-consistent with a tuple
$a \in A$ if there is some mapping $\mu$ for $\text{Var}(Q)$, where $\mu$
is not necessarily a match for $Q$ in $I$, such that $\mu(\varphi) = f$
and $\mu(y) = a$. The unirelation algorithm simply constructs
the solution $J_\varphi$ for each $\varphi \in \text{atoms}(Q)$, and returns the
best $J_\varphi$, that is, the one with the maximal $|\text{Var}(J_\varphi)|$. Observe
that the unirelation algorithm is indeed highly naive, since it
does not even consider simultaneous deletions of facts from
different relations. Moreover, this algorithm does not guar-
antee any multiplicative-factor approximation; as evidence,
observe that the solution $J$ it returns for the input $I^*$ and
$A$ of MAXVIEW($Q^*_2$) that we defined in Example 6.2 is such that
$Q^*_2(J) = \emptyset$. However, this algorithm does provide an
approximation guarantee that we will use for analyzing the
complexity of approximating $\text{MaxView}^\epsilon(Q)$. This guarantee is stated by the following lemma.

**Lemma 6.6.** Let $Q$ be an sjf-CQ with $h$-coverage. Let $I$ and $A$ be input for $\text{MaxView}^\epsilon(Q)$. The unirule algorithm returns a solution $J$ with

$$|Q(J)| \geq |Q(I)|/h - |A|^h.$$  

**Proof.** Suppose that the atoms $\varphi_1, \ldots, \varphi_h$ cover the head variables of $Q$. For $i = 1, \ldots, h$, let $C_i$ be the set of tuples $b$ in $Q(I)$ such that for some $a \in A$, the projection of $b$ to $\text{Var}_h(\varphi_i)$ is the same as that of $a$. Let $C$ be the set $C_1 \cap \cdots \cap C_h$. An easy observation is that $|C| \leq |A|^h$, since $\varphi_1, \ldots, \varphi_h$ cover $\text{Var}_h(\varphi)$. By definition, each tuple $a \in Q(I) \setminus C$ is excluded from some $C_i$. Now, a simple counting argument implies that there is some $j$ such that $C_j$ excludes at least $|Q(I)/C|/h$ answers. The solution $J_{\varphi_j}$ is such that $Q(J_{\varphi_j})$ contains all those excluded answers, and hence, $|Q(J)| \geq |Q(I)|/h - |A|^h$. So, we get the correctness of the lemma, since $J_{\varphi_j}$ is of the solutions considered by the unirule algorithm.  

The next lemma implies that $\text{MaxView}^\epsilon(Q)$ can be solved optimally, in polynomial time, when $Q(I)$ is of a bounded cardinality.

**Lemma 6.7.** Let $Q$ be an sjf-CQ, and let $d$ be a fixed number. Given the input $I$ and $A$ for $\text{MaxView}^\epsilon(Q)$, if we assume that $|Q(I) \setminus A| \leq d$, then $\text{MaxView}^\epsilon(Q)$ can be solved in polynomial time.

**Proof.** Let $D = Q(I) \setminus A$. If $J$ is any solution, then $Q(J) \subseteq D$. For each $a \in Q(J)$, fix an arbitrary match $\mu_a$ for $Q$ in $J$, such that $\mu_a = (y, a)$. Let $J'$ be the subinstance of $J$ that consists of all the facts $\mu_a(\varphi)$ where $a \in Q(J)$ and $\varphi \in \text{atoms}(Q)$. Note that $|Q(J) = Q(J')|$. Moreover, observe that $|Q(J)| \leq d \cdot |\text{atoms}(Q)|$. It thus suffices to search for an optimal solution among all the subinstances of $I$ of a cardinality bounded by the fixed number $d\cdot |\text{atoms}(Q)|$. And of course, searching all those subinstances can be done in polynomial time.  

We can now prove our upper bound for approximating $\text{MaxView}^\epsilon(Q)$.

**Theorem 6.8.** Let $Q$ be an sjf-CQ with $h$-coverage, and let $\epsilon > 0$. The problem $\text{MaxView}^\epsilon(Q)$ is $(h+\epsilon)$-approximable in polynomial time.

**Proof.** Let $I$ and $A$ be input for $\text{MaxView}^\epsilon(Q)$. We fix a numeric function $f(h, c, \epsilon)$ that we determine later. Note that $f(h, c, \epsilon)$ does not depend on the input $I$ and $A$. In particular, if $|Q(I) \setminus A|$ has at most $f(h, c, \epsilon)$ answers, then we apply Lemma 6.7 to get an optimal solution in polynomial time. So assume that $|Q(I) \setminus A| \geq f(h, c, \epsilon)$. We will define $f(h, c, \epsilon)$ to be such that Lemma 6.6 gives a $(h+\epsilon)$-approximation. Specifically, we need the following to hold:

$$(h + \epsilon)/|Q(I)/h - |A|^h) \geq |Q(I) \setminus A|$$

Hence, it suffices to require the following:

$$|Q(I)| \left( \frac{h + \epsilon}{h} - 1 \right) \geq |A| + (h + \epsilon)|A|^h$$

Then, using the assumption that $|A| \leq c$ we get that it suffices to choose $f(h, c, \epsilon)$ as:

$$f(h, c, \epsilon) = 2h(h + \epsilon)c^h/\epsilon$$

That completes the proof.  

### 6.2 Functional Dependencies

In this section, we generalize the complexity results of Section 5 to incorporate functional dependencies.

#### 6.2.1 Notation

We begin with some notation. Recall that a schema is a finite sequence $R$ of relation symbols. We extend the definition of a schema to include functional dependencies. A functional dependency (over $R$), abbreviated $fd$, has the form $R_i : A \rightarrow B$, where $R_i \in R$ and $A$ and $B$ are subsets of $\{1, \ldots, \text{arity}(R_i)\}$. So, in this section a schema has the form $S = (R, \Delta)$, where $\Delta$ is a set of functional dependencies. An instance over $S$ satisfies every $fd$ in $\Delta$, which means that for every $fd$ $R_i : A \rightarrow B$ in $\Delta$ and tuples $t$ and $u$ in $R_i^*$, if $t$ and $u$ agree on (i.e., have the same values for) the indices of $A$, then they also agree on the indices of $B$.

Let $S = (R, \Delta)$ be a schema, and let $Q$ be a CQ over $S$. If $\mu$ is an assignment for $Q$ and $Z$ is a subset of $\text{Var}(Q)$, then $\mu[Z]$ denotes the restriction of $\mu$ to the variables in $Z$ (i.e., $Z$ is the domain of $\mu[Z]$, and $\mu[Z](x) = \mu(x)$ for all $x \in Z$). Let $Z_1$ and $Z_2$ be subsets of $\text{Var}(Q)$. We denote by $Q : Z_1 \rightarrow Z_2$ the functional dependency stating that for every instance $I$ over $S$ and matches $\mu, \mu' \in \mathcal{M}(Q, I)$ we have:

$$\mu[Z_1] = \mu'[Z_1] \Rightarrow \mu[Z_2] = \mu'[Z_2]$$

Note that when $\Delta$ is empty, $Q : Z_1 \rightarrow Z_2$ is equivalent to $Z_1 \supseteq Z_2$. We may write $Q : Z \rightarrow z$ instead of $Q : Z \rightarrow \{z\}$. The image of $z$, denoted $\text{img}(z)$, is the set of all the variables $z \in \text{Var}(Q)$ with $Q : Z \rightarrow z$. Note that $Z \subseteq \text{img}(z)$. For $\varphi \in \text{atoms}(Q)$, we use $\text{img}(\varphi)$ as a shorthand notation of $\text{img}(\text{Var}(\varphi))$.

**Example 6.9.** The schema $S$ in our running example for this section has 4 relation symbols, ternary $R$, binary $S$, quaternary $T$ and binary $U$, and the following fds:

$$R : 1 \rightarrow 2 \quad R : 2 \rightarrow 3 \quad T : 1 \rightarrow 2 \quad T : \{2, 3\} \rightarrow 4$$

Let $Q_0$ be the following sjf-CQ:

$$Q_0(y_1, y_2, y_3) = R(x_1, y_1, x_3), S(x_1, x_2), T(x_2, y_2, x_3, x_4), U(x_4, y_3)$$

We have $Q_0 : \{x_1, x_2\} \rightarrow y_1$ due to the fd $R : 1 \rightarrow 2$. As a result, we also have $Q_0 : \{x_1, x_2\} \rightarrow x_3$ due to the fd $R : 2 \rightarrow 3$. Let $\varphi_S$ be the atom $S(x_1, x_2)$. The reader can verify that $\text{img}(\varphi_S) = \{x_1, x_2, y_1, x_3, y_2, x_4\}$.  

**Functional head domination** [16] is a generalization of head domination that incorporates functional dependencies. Here we refine that generalized definition with the corresponding level-k parameterization.

**Definition 6.10. (Functional Head Domination)** A CQ $Q$ over a schema $S$ has functional head-domination if there is a subset $\Phi$ of $\text{atoms}(Q)$ such that for every connected component $P$ of $G_\Phi(Q)$ there is $\varphi \in \Phi$ with $\text{Var}_n(P) \subseteq \text{img}(\varphi)$; in that case, $Q$ has level-$k$ functional head domination if $k$ is the minimal cardinality of such a subset $\Phi$.  

**Example 6.11.** We continue our running example over the schema $S$ and sjf-CQ $Q_0$ defined in Example 6.9. The graph $G_2(Q_0)$ is shown in the top of Figure 4. Observe that $G_2(Q_0)$ has one connected component, with the head
variables $y_1$, $y_2$ and $y_3$. The reader can verify that no atom $\varphi$ of $Q_0$ satisfies $\{y_1, y_2, y_3\} \subseteq \text{img}(\varphi)$. Therefore, $Q_0$ does not have functional head domination.

For a CQ $Q$ over a schema $S$, we denote by $Q^+$ the CQ that is obtained by appending the head to $Q$ (in some order) all the existential variables that are functionally determined by the head variables; so, the set $\text{Var}_0(Q^+)$ of head variables of $Q^+$ is not $\text{Var}_0(Q)$, but rather the set $\text{img}(\text{Var}_0(Q))$.

Example 6.12. For the schema $S$ and sjf-CQ $Q_0$ of Example 6.9, we have:

$$\text{img}(\{y_1, y_2, y_3\}) = \{y_1, y_2, y_3, x_3, x_4\}$$

Therefore, the CQ $Q_0^+$ is the same as $Q_0$, except that the head of $Q_0^+$ is $Q_0^+(y_1, y_2, y_3, x_3, x_4)$. The graph $G_3(Q_0^+)$ is shown in the middle part of Figure 4.

Example 6.15. We continue our running example with Example 6.12 that $Q_0^+$ has functional head domination. Therefore, Theorem 6.13 states that $\text{MinVSE}_c(S, Q_0)$ is solvable (optimally) in polynomial time.

Note that Theorem 6.13 generalizes Theorem 4.6 by replacing “head domination” with “functional head domination” and “$Q^+$” with “$Q^+$.” The next theorem similarly generalizes Theorems 5.11 and 5.8, showing that Table 1 generalizes to account for functional dependencies (up to the needed replacement of terms).

Theorem 6.16. Let $S$ be a schema, and let $Q$ be an sjf-CQ over $S$, and let $c$ be a fixed natural number.

1. If $Q^+$ has level-1 functional head domination, then $\text{MinVSE}_c(S, Q)$ is solvable in polynomial time.

2. If $Q^+$ has level-k functional head domination for $k > 1$, then $\text{MinVSE}_c(S, Q)$ is NP-hard but $k$-approximable in polynomial time; moreover, $\text{FreeVSE}_c(S, Q)$ and $\text{MinVSE}_c(S, Q)$ are then solvable in polynomial time.

3. If $Q^+$ does not have functional head domination, then $\text{FreeVSE}_c(S, Q)$ is NP-complete; hence, it is NP-hard to approximate $\text{MinVSE}_c(S, Q)$ by any finite factor.

Example 6.17. To complete our running example, recall from Example 6.15 that $\text{MinVSE}_c(S, Q_0)$ is solvable in polynomial time. Recall from Example 6.12 that $Q_0^+$ has level-2 functional head domination. Then Part 2 of Theorem 6.16 states that $\text{MinVSE}_c(S, Q)$ is NP-hard, but 2-approximable in polynomial time. It also says that $\text{FreeVSE}_c(S, Q_0)$ and $\text{MinVSE}_c(S, Q_0)$ are both solvable in polynomial time (since $Q_0^+$ has functional head domination).

The proofs of parts 1 and 3 of Theorem 6.16 are similar to those we described for their correspondents in Theorems 5.11 and 5.8. The more involved proof is for part 2, and specifically the generalization of Lemma 5.6 to incorporate fds. That proof, which we do not provide here for lack of space, will appear in the full version of this paper.

7. CONCLUSIONS AND DISCUSSION

We studied the computational complexity of deletion propagation when the view is defined by an sjf-CQ, and multiple tuples are to be deleted. In particular, we investigated the problem $\text{MinVSE}(Q)$ and showed a trichotomy in complexity, classifying the sjf-CQs into those where the problem is in polynomial time, NP-hard but approximable, and inapproximable. We extended these results to $\text{MinVSE}_c(Q)$, where at most $c$ tuples are deleted, and generalized them to accommodate functional dependencies. We also studied the problems $\text{MaxView}(Q)$ and $\text{MaxView}_c(Q)$, where the goal is to approximate an optimal solution by aiming to maximize the remaining view (rather than to minimize the side effect). There, we showed that known results on the approximability of $\text{MaxView}_1(Q)$ extend to $\text{MaxView}_c(Q)$, but not to $\text{MaxView}(Q)$; the latter problem is inapproximable by any constant factor (under the complexity assumption...
Many questions regarding the complexity of deletion propagation remain unanswered. Does any of the known dichotomy or trichotomy results, for bounded or unbounded number of tuples, extend to CQs with self joins? It is known that head domination is not a sufficient condition for the tractability of finding an optimal solution if self joins are allowed [17]. Can MaxView($Q$) be approximated better than shown here? Can our established inapproximability for MaxView($Q$) be generalized to a classification over the entire class of (sjf-)CQs $Q$? How do our results extend to handle multiple views (i.e., deleted tuples come from multiple views and/or the goal is to simultaneously minimize multiple views)? we note that our reduction to focused hitting set extends to multiple views, but it not at all clear how and whether the trichotomy so extends. Another question relates to privacy: if the source relations contain sensitive information that is hidden from the users of the view, how do we avoid leaking that information through the side effect? The work reported in this paper relates to the concept of causality in databases that has lately gained interest in the database research community [20,21,24]. In fact, the debugging motivation from the beginning of the introduction can be viewed as a special case thereof. In the work on causality, the goal is to detect source facts that hold high responsibility for the membership of given answers in the view. The facts deleted in an (approximately) optimal solution of our setting could serve as candidates of such owners of responsibility. Validity is according to the semantics of Meliou et al. [20]—eliminating the facts causes the elimination of the answers; but here, the extent of responsibility is based on the focus on the answers. As an example, suppose that “Alin lives in USA” is the result of joining two facts: “Alin lives in CA” and “CA is in USA.” The former fact would be assigned a higher responsibility, since eliminating the latter would incur a larger side effect. This notion of causality is different from the one of Meliou et al. [20] that, interestingly, corresponds to deletion propagation with a minimal source side effect (and, in particular, assigns the same responsibility to both facts in the above example).

Our complexity results in this paper add to the dichotomy results established by the theoretical database research. Besides the ones on deletion propagation we referenced in the body of this paper, various dichotomies in the complexity of operations involving CQs (and sjf-CQs) have been proved. Dalvi and Suciu [10,11] studied query evaluation on probabilistic databases with independent tuples, and classified the CQs into those that can be evaluated in polynomial time and those that are #P-hard. Meliou et al. [22] showed a dichotomy (polynomial time vs. NP-completeness) in the complexity of computing the degree of responsibility of source facts to the tuples of an sjf-CQ. Kolaitis and Pema [18] proved a dichotomy (polynomial time vs. coNP-hardness) in the complexity of computing the consistent answers of a Boolean sjf-CQ with exactly two atoms. Finally, Maslowski and Wijsen [19] showed a dichotomy (polynomial time vs. #P-hardness) in the complexity of counting the database repairs that satisfy a Boolean sjf-CQ.

8. REFERENCES