# Maximizing Conjunctive Views in Deletion Propagation

Benny Kimelfeld Jan Vondrák Ryan Williams
IBM Research–Almaden
San Jose, CA 95120, USA
{kimelfeld, jvondrak, ryanwill}@us.ibm.com

## ABSTRACT

In deletion propagation, tuples from the database are deleted in order to reflect the deletion of a tuple from the view. Such an operation may result in the (often necessary) deletion of additional tuples from the view, besides the intentionally deleted one. The complexity of deletion propagation is studied, where the view is defined by a conjunctive query (CQ), and the goal is to maximize the number of tuples that remain in the view. Buneman et al. showed that for some simple CQs, this problem can be solved by a trivial algorithm. This paper identifies additional cases of CQs where the trivial algorithm succeeds, and in contrast, it proves that for some other CQs the problem is NP-hard to approximate better than some constant ratio. In fact, this paper shows that among the CQs without self joins, the hard CQs are *exactly* the ones that the trivial algorithm fails on. In other words, for every CQ without self joins, deletion propagation is either APX-hard or solvable by the trivial algorithm.

The paper then presents approximation algorithms for certain CQs where deletion propagation is APX-hard. Specifically, two constant-ratio (and polynomial-time) approximation algorithms are given for the class of star CQs without self joins. The first algorithm is a greedy algorithm, and the second is based on randomized rounding of a linear program. While the first algorithm is more efficient, the second one has a better approximation ratio. Furthermore, the second algorithm can be extended to a significant generalization of star CQs. Finally, the paper shows that self joins can have a major negative effect on the approximability of the problem.

**Categories and Subject Descriptors:** H.2 [Database Management]: Systems, Database Administration

**General Terms:** Algorithms, Theory

**Keywords:** Deletion propagation, dichotomy, approximation


*PODS'11,* June 13–15, 2011, Athens, Greece.


## 1. INTRODUCTION

Classical in database management is the *view update problem* [2, 8, 9, 11, 15, 20]: translate an update operation on the view to an update of the source database, so that the update on the view is properly reflected. A special case of this problem is that of *deletion propagation* in relational databases: given an *undesired* tuple in the view (defined by some monotonic query), delete some tuples from the base relations (where such tuples are referred to as *facts*), so that the undesired tuple disappears from the view, but the other tuples in the view remain. The database resulting from this deletion of tuples is said to be *side-effect free* [4], where the "side effect" is the set of deleted view tuples that are different from the undesired one. Very often, a solution that is side-effect free does not exist, and hence, the task is relaxed to that of *minimizing* the side effect [4,8]. That is, the problem is to delete tuples from the source relations so that the undesired tuple disappears from the view, and the maximum possible number of other tuples remain in the view.

Though view update is a classical database problem, recent applications provide a renewed motivation for it. Specifically, view update naturally arises when debugging *Information Extraction* (IE) programs, which can be highly complicated [23]. As a concrete example, the MIDAS system [1] extracts basic relations from multiple (publicly available) financial data sources, some of which are semistructured or just text, and integrates them into composite *entities*, *events* and *relationships*. Naturally, this task is error prone, and due to the magnitude and complexity of MIDAS, it is practically impossible to reach complete precision. An erroneous conclusion, though, can be observed by a user viewing the final output of MIDAS, and such a conclusion is likely to be the result of errors or ambiguity in the base relations. When the integration query is taken as the view definition, deletion propagation becomes the task of suggesting tuples to be deleted from the base relations for eliminating the erroneous conclusion, while minimizing the effect on the remaining conclusions. Furthermore, eliminating tuples from the base relations may itself entail deletion propagation, since these tuples are typically extracted by consulting external (possibly unclean) data sources [23, 25].

Another motivation arises in *business reorganization*, and the specific application is that of eliminating undesired links, such as between a specific employee and a specific customer. This problem is nicely illustrated by the following simple example by Cui and Widom [10] (also referenced and used by Buneman et al. [4]). Let GroupUser(group, user) and GroupFile(group, file) be two relations representing mem-

berships of users in groups and access permissions of groups to files. A user $u$ can access the file $f$ if $u$ belongs to a group that can access $f$; that is, there is some $g$, such that GroupUser$(g, u)$ and GroupFile$(g, f)$. Suppose that we want to restrict the access of a specific user to a specific file, by eliminating some user-group or group-file pairings. Furthermore, we would like to do so in a way that a maximum number of user-file access permissions remain. This is exactly the deletion-propagation problem, where the view is defined by the following conjunctive query (CQ):

$$\text{Access}(y_1, y_2) :- \text{GroupUser}(x, y_1), \text{GroupFile}(x, y_2) \quad (1)$$

A formal definition of a CQ is given in Section 2. For some of the above applications of deletion propagation (e.g., IE), the involved queries can be much more complicated than CQs. Nevertheless, we believe that understanding deletion propagation for the basic class of CQs is an essential step towards practical algorithms for these applications. Hence, the focus of this paper is on the complexity of deletion propagation, where the view is defined by a CQ.

Formally, for a CQ $Q$ (the view definition), the input for the deletion-propagation problem consists of a database instance $I$ and a tuple $\mathbf{a} \in Q(I)$. A *solution* is an subinstance $J$ of $I$ (i.e., $J$ is obtained from $I$ by deleting facts) such that $\mathbf{a} \notin Q(J)$, and an *optimal solution* maximizes the number of tuples in $Q(J)$. The *side effect* is $(Q(I) \setminus \{\mathbf{a}\}) \setminus Q(J)$. Buneman et al. [4] identify some classes of CQs (e.g., projection-free CQs) for which a straightforward algorithm produces an optimal solution in polynomial time; we recall this algorithm in Section 3 and call it *the trivial algorithm*. But those tractable classes are highly restricted, as Buneman et al. show that even for a CQ as simple as the above Access$(y_1, y_2)$, testing whether there is a solution that is side-effect free is NP-complete. Therefore, finding an *optimal solution* minimizing the side effect is NP-hard for Access$(y_1, y_2)$. Of course, one can often settle for a solution that is just *approximately* optimal, such as a solution that has a side effect that is at most twice as large as the minimum. Nevertheless, as noted by Buneman et al. [4], approximating the minimum side effect is still hard, since such an approximation can be used to test for the existence of a side-effect free solution (because an approximately optimal solution must be side-effect free when a side-effect-free solution exists).

In spite of the above hardness in deletion propagation, we still want to design algorithms (at least for important classes of CQs) with provable guarantees on the quality of a solution. We achieve that by slightly tweaking the optimization measure: instead of trying to minimize the side effect (i.e., the cardinality of $(Q(I) \setminus \{a\}) \setminus Q(J)$), our goal is to maximize the number of remaining tuples (i.e., the cardinality of $Q(J)$). We denote this problem by MaxDP$\langle Q\rangle$. Of course, for finding an optimal solution, this maximization problem is *the same* as the original minimization problem. But in terms of approximation, the picture drastically changes. For example, we show that for *every* CQ $Q$ without self joins, MaxDP$\langle Q\rangle$ is approximable by a constant factor that depends only on $Q$. More specifically, for every CQ $Q$ without self joins there is a constant $\alpha_Q \in (0, 1)$ (lower bounded by the reciprocal of the arity of $Q$), such that the solution $J$ produced by the trivial algorithm satisfies $|Q(J)| \geq \alpha_Q |Q(J')|$ for all solutions $J'$.

In Section 4, we formulate the *head-domination* property of CQs. We prove that when a CQ $Q$ without self joins has this property, MaxDP$\langle Q\rangle$ is solved optimally (and in polynomial time) by the trivial algorithm. Unfortunately, for many other CQs there is a limit to the extent to which MaxDP$\langle Q\rangle$ can be approximated. Particularly, we consider the self-join-free CQs, and show a remarkable phenomenon (Theorem 4.8): for those CQs that do not have the head domination property, not only is the trivial algorithm suboptimal for MaxDP$\langle Q\rangle$, this problem is hard to solve optimally, and even hard to approximate better than some constant ratio. More precisely, we show the following dichotomy. For every CQ $Q$ without self joins, one of the following holds:

- $Q$ has the head-domination property (thus, the trivial algorithm *optimally* solves MaxDP$\langle Q\rangle$ in polynomial time).
- There is a constant $\alpha_Q \in (0, 1)$, such that MaxDP$\langle Q\rangle$ is NP-hard to $\alpha_Q$-approximate.

The proof of this dichotomy is nontrivial. Regarding the constant $\alpha_Q$, we show that it is necessary to have it dependent on $Q$, since there is no global constant $\alpha \in (0, 1)$ that works for all the CQs $Q$ (without self joins); however, we show that such a global constant exists for a large class of CQs (that contains the *acyclic* CQs [3, 12]). The above dichotomy also holds for the aforementioned problems of Buneman et al. [4]. That is, for the problem of determining whether there is a side-effect-free solution, and for the problem of finding an optimal solution, the following holds when there are no self joins: if the CQ has the head-domination property, then the problems are solvable by the trivial algorithm; otherwise, these problems are NP-hard.

So far, the only algorithm we have considered for deletion propagation is the trivial one. Recall that for every CQ $Q$ without self joins, the trivial algorithm is a constant-factor approximation (with a constant depending on $Q$). In Section 5, we devise approximation algorithms with much better ratios, for the important class of star CQs without self joins[1] (which generalize the queries that are processed by the *Fagin algorithm* [13] and the *threshold algorithm* [14]). We first give an algorithm that provides a $\frac{1}{2}$-approximation for MaxDP$\langle Q\rangle$, where $Q$ is a star CQ without self joins. This algorithm, which we call *the greedy algorithm*, is based on purely combinatorial arguments. This algorithm is very simple and has a highly desirable complexity: its running time is polynomial not just under the standard *data complexity*, but also under *query-and-data* complexity (which means that the worst-case cost of this algorithm is significantly smaller than that of evaluating the CQ over the database instance).

We then present an approximation for MaxDP$\langle Q\rangle$ that is based on randomized rounding of a linear program (LP). This randomized algorithm[2] is more complicated than the greedy algorithm. We formulate the problem as an integer linear program, which is relaxed to an ordinary linear program by means of randomized rounding that translates the resulting LP solution into a probabilistic choice of tuples to delete. The LP-based algorithm terminates in polynomial time, but unlike the greedy algorithm, the polynomial is only under data complexity (as it essentially requires the

[1] See Section 5 for the exact definition of a star CQ.

[2] A discussion on randomness in optimization is given in Section 5.

evaluation of the CQ). However, the LP-based algorithm has important two advantages over the greedy one.

The first advantage of the LP-based algorithm is that its approximation ratio is higher: $1 - \frac{1}{e}$ (roughly, 0.632) rather than $\frac{1}{2}$ (which is shown to be tight for the greedy algorithm). The second advantage is that the LP-based algorithm can be extended to a significant generalization of star CQs. Roughly speaking, in this generalization (which we formally define in Section 5) the star restriction is limited just to the existential variables. As we demonstrate, the greedy algorithm inherently fails on CQs of this generalization. The LP-based approach alone is not enough for this generalization, but interestingly, the trivial algorithm handles the cases where the LP fails. Hence, our generalized algorithm takes the best solution from those returned by the LP-based algorithm and the trivial algorithm.

Finally, in Section 6 we show that self joins in CQs introduce inherent hardness. Recall that for every CQ $Q$ without self joins, MaxDP$\langle Q\rangle$ can be efficiently $\alpha_Q$-approximated by the trivial algorithm, where $\alpha_Q$ is bounded by the reciprocal of the arity of $Q$. We show that this result does not extend to CQs with self joins. Specifically, the trivial algorithm fails to give the desired approximation, and furthermore, we show an infinite set of CQs $Q$, such that the achievable approximation ratio for MaxDP$\langle Q\rangle$ is exponentially smaller than the reciprocal of the arity of $Q$. In addition, we show a CQ $Q$ with self joins, such that $Q$ has the head-domination property and yet the trivial algorithm is sub-optimal; furthermore, MaxDP$\langle Q\rangle$ is hard to approximate better than some constant ratio.

Note that the work reported here considers a basic and restricted case of the view update problem: deletion propagation for conjunctive views, with the goal of preserving as many tuples of the view as possible. We found that even in this basic case, approximation is a nontrivial topic. We believe that the insights and techniques drawn from this work will be helpful in the exploration of additional aspects of view update, and deletion propagation in particular, like those studied in the literature. For example, deletion propagation has been explored under the goal of minimizing the *source side effect* [4,8], namely, finding a solution with a minimal number of missing facts. Cong et al. [8] also studied the complexity of deletion propagation (with the goal of finding an optimal solution) in the presence of *key constraints*. Naturally, update operations other than deletion (e.g., insertion and replacement) have also been investigated [2], and especially in the presence of functional dependencies [9, 11, 20]. The work of Cosmadakis and Papadimitriou [9] is distinguished by their requirement for a view to keep intact a *complement view* (which has also been explored more recently by Lechtenbörger and Vossen [22]) that, intuitively, contains the information ignored by the view.

Due to a lack of space, some of the proofs are presented in the extended version of this paper [21].

# 2. PRELIMINARIES

## 2.1 Schemas, Instances, and CQs

We fix an infinite set $\mathsf{Const}$ of *constants*. We usually denote constants by lowercase letters from the beginning of the Latin alphabet (e.g., $a$, $b$ and $c$). A *schema* is a finite sequence $\mathbf{R} = \langle R_1, \ldots, R_m\rangle$ of distinct relation symbols, where each $R_i$ has an arity $r_i > 0$. An *instance* $I$ (*over* $\mathbf{R}$) is a sequence $\langle R_1^I, \ldots, R_m^I\rangle$, such that each $R_i^I$ is a finite relation of arity $r_i$ over $\mathsf{Const}$ (i.e., $R_i^I$ is a finite subset of $\mathsf{Const}^{r_i}$). We may abuse this notation and use $R_i$ to denote both the relation symbol and the relation $R_i^I$ that interprets it. If $\mathbf{c} \in \mathsf{Const}^{r_i}$, then $R_i(\mathbf{c})$ is called a *fact*, and it is a fact *of* the instance $I$ if $\mathbf{c} \in R_i^I$. Notationally, we view an instance as the set of its facts (hence, we may write $R(\mathbf{c}) \in I$ to say that $R(\mathbf{c})$ is a fact of $I$).

If $I$ and $J$ are two instances over $\mathbf{R} = \langle R_1, \ldots, R_m\rangle$, then $J$ is a *sub-instance* of $I$, denoted $J \subseteq I$, if $R_i^J \subseteq R_i^I$ for all $i = 1, \ldots, m$.

We fix an infinite set $\mathsf{Var}$ of *variables*, and assume that $\mathsf{Const}$ and $\mathsf{Var}$ are disjoint sets. We usually 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): a CQ over a schema $\mathbf{R}$ is an expression of the form

$$Q(\mathbf{y}) :- \varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$$

where $\mathbf{x}$ and $\mathbf{y}$ are tuples of variables (from $\mathsf{Var}$), $\mathbf{c}$ is a tuple of constants (from $\mathsf{Const}$), and $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$ is a conjunction of atomic formulas $R_i(\mathbf{x}, \mathbf{y}, \mathbf{c})$ over $\mathbf{R}$; an atomic formula is also called an *atom*. We may write just $Q(\mathbf{y})$ or $Q$ if $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$ is irrelevant. We denote by atoms$(Q)$ the set of atoms of $Q$. We usually write $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$ by simply listing atoms$(Q)$. We make the requirement that each variable occurs at most once in $\mathbf{x}$ and $\mathbf{y}$, and no variable occurs in both $\mathbf{x}$ and $\mathbf{y}$. Furthermore, we require every variable of $\mathbf{y}$ to occur (at least once) in $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$.

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 (with the proper arities) that appear in $Q$. When we want to refer to that schema, we denote it by schema$(Q)$. Multiple occurrences of the same relation symbol in $Q$ form a *self join*; hence, when we say that $Q$ *has no self joins* we mean that every relation symbol appears at most once in $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$.

Let $Q(\mathbf{y}) :- \varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$ be a CQ. A variable of $\mathbf{x}$ is called an *existential* variable of $Q$, and a variable of $\mathbf{y}$ is called a *head* variable of $Q$. We use $\mathsf{Var}_\exists(Q)$ and $\mathsf{Var}_\mathsf{h}(Q)$ to denote the sets of existential variables and head variables of $Q$, respectively. Similarly, if $\phi$ is an atom of $Q$, then $\mathsf{Var}_\exists(\phi)$ and $\mathsf{Var}_\mathsf{h}(\phi)$ denote the set of existential variables and head variables, respectively, that occur in $\phi$. We denote by $\mathsf{Var}(Q)$ and $\mathsf{Var}(\phi)$ the unions $\mathsf{Var}_\exists(Q) \cup \mathsf{Var}_\mathsf{h}(Q)$ and $\mathsf{Var}_\exists(\phi) \cup \mathsf{Var}_\mathsf{h}(\phi)$, respectively. A *join variable* of $Q$ is a variable that occurs in two or more atoms of $Q$ (note that a join variable can be an existential variable or a head variable). Finally, the *arity of* $Q$, denoted arity$(Q)$, is the length of the tuple $\mathbf{y}$.

Example 2.1. An important CQ in this work is the CQ $Q_2^\star$, which is the same as the CQ Access$(y_1, y_2)$ defined in (1) (and discussed in the introduction), up to renaming of relation symbols.

$$Q_2^\star(y_1, y_2) :- R_1(x, y_1), R_2(x, y_2) \tag{2}$$

Here and later, in our examples $R_i$ and $R_j$ are assumed to be different symbols when $i \neq j$. The atoms of $Q_2^\star$ are $\phi_1 = R_1(x, y_1)$ and $\phi_2 = R_2(x, y_2)$. Note that $Q_2^\star$ 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 $Q_2^\star$, namely $x$, and the two head variables are $y_1$ and $y_2$.

Hence $\mathsf{Var}_{\exists}(Q) = \{x\}$ and $\mathsf{Var}_{\mathsf{h}}(Q) = \{y_1, y_2\}$. Furthermore, $\mathsf{Var}_{\exists}(\phi_1) = \{x\}$ and $\mathsf{Var}_{\mathsf{h}}(\phi_1) = \{y_1\}$. Finally, note that $x$ is the single join variable of $Q_2^\star$.

We generalize the notation $Q_2^\star$ to $Q_k^\star$, for all positive integers $k$, where the CQ $Q_k^\star$ is defined as follows.

$$Q_k^\star(y_1, \ldots, y_k) :- R_1(x, y_1), \ldots, R_k(x, y_k) \qquad (3)$$

Note that schema($Q_k^\star$) consists of the $k$ (distinct) binary relations $R_1, \ldots, R_k$. □

Observe that the CQ $Q_k^\star$ from Example 2.1 does not contain any constants. An example of a query with the constant Emma is the following.

$$Q(y_1, y_2) :- R_1(x, y_1), R_2(x, y_2, \mathsf{Emma})$$

Consider the CQ $Q(\mathbf{y})$, and let $I$ be an instance over schema($Q$). An *assignment* for $Q$ is a mapping $\mu : \mathsf{Var}(Q) \to \mathsf{Const}$. For an assignment $\mu$ for $Q$, the tuple $\mu(\mathbf{y})$ is the one obtained from $\mathbf{y}$ by replacing every head variable $y$ with the constant $\mu(y)$. Similarly, for an atom $\phi \in \text{atoms}(Q)$, the fact $\mu(\phi)$ is the one obtained from $\phi$ by replacing every variable $z$ with the constant $\mu(z)$. A *match for $Q$ in $I$* is an assignment $\mu$ for $Q$, such that $\mu(\phi)$ is a fact of $I$ for all atoms $\phi \in \text{atoms}(Q)$. If $\mu$ is a match for $Q$ in $I$, then $\mu(\mathbf{y})$ is called an *answer (for $Q$ in $I$)*. The *result* of evaluating $Q$ over $I$, denoted $Q(I)$, is the set of all the answers for $Q$ in $I$.

## 2.2 Deletion Propagation

Let $Q$ be a CQ. The problem of maximizing the view in deletion propagation, with $Q$ as the view definition, is denoted by MaxDP$\langle Q\rangle$ and is defined as follows. The input consists of an instance $I$ over schema($Q$), and a tuple $\mathbf{a} \in Q(I)$. A *solution (for $I$ and $\mathbf{a}$)* is an instance $J \subseteq I$, such that $\mathbf{a} \notin Q(J)$. The goal is to find an *optimal* solution, which is a solution $J$ that maximizes $|Q(J)|$; that is, $J$ is such that $|Q(J)| \geq |Q(K)|$ for all solutions $K$.

As we discuss later, finding an optimal solution for the problem MaxDP$\langle Q\rangle$ may be intractable. Often, though, we can settle for *approximations*, which we naturally define as follows. For a number $\alpha \in [0, 1]$, a solution $J$ is *$\alpha$-optimal* if $|Q(J)| \geq \alpha \cdot |Q(K)|$ for all solutions $K$. An *$\alpha$-approximation for* MaxDP$\langle Q\rangle$ is an algorithm that, given $I$ and $\mathbf{a}$, always returns an $\alpha$-optimal solution.

# 3. THE TRIVIAL ALGORITHM

In this section, we recall a straightforward algorithm of Buneman et al. [4], which they gave for proving the tractability of finding a solution $J$ with a minimum-size side effect (recall that the *side effect* is the set $(Q(I) \setminus \{\mathbf{a}\}) \setminus Q(J)$), in the case where there is no projection (i.e., $Q$ has no existential variables); here, we trivially extend that algorithm to general CQs.

Consider a CQ $Q(\mathbf{y})$, and suppose that $I$ and $\mathbf{a}$ are input for MaxDP$\langle Q\rangle$. Let $\phi$ be an atom of $Q$. A *$\phi$-fact* of $I$ is a fact $f \in I$ that is equal to $\mu(\phi)$ for some assignment $\mu$ for $Q$ (note that $\mu$ is not necessarily a match for $Q$ in $I$); furthermore, if there is such $\mu$ that satisfies $\mu(\mathbf{y}) = \mathbf{a}$ (in addition to $\mu(\phi) = f$), then we say that $f$ is *consistent with* $\mathbf{a}$. The *trivial algorithm* for generating a solution for $I$ and $\mathbf{a}$ is as follows. For each $\phi \in \text{atoms}(Q)$, we obtain from $I$ the sub-instance $J_\phi$ by removing all of the $\phi$-facts that are consistent with $\mathbf{a}$. Then, we return the $J_\phi$ that maximizes $|Q(J_\phi)|$. Pseudo-code for Trivial$\langle Q\rangle$ is shown in Figure 1.

**Algorithm** Trivial$\langle Q\rangle(I, \mathbf{a})$

```
J ← ∅
for all φ ∈ atoms(Q) do
    J_φ ← I
    remove from J_φ all φ-facts consistent with a
    if |Q(J_φ)| > |Q(J)| then
        J ← J_φ
return J
```

**Figure 1: The trivial algorithm**

Example 3.1. Consider the following CQ $Q_3^\star$, which is a special case of (3).

$$Q_3^\star(y_1, y_2, y_3) :- R_1(x, y_1), R_2(x, y_2), R_3(x, y_3)\,.$$

Figure 2 shows an instance $I_3$ over schema($Q_3^\star$), and let $\mathbf{a}$ be the tuple $(\diamond, \diamond, \diamond)$. Let us show how the trivial algorithm operates on $I_3$ and $\mathbf{a}$. For $i \in \{1, 2, 3\}$, let $\phi_i$ be the atom $R_i(x, y_i)$. Then atoms($Q_3^\star$) $= \{\phi_1, \phi_2, \phi_3\}$. For $i \in \{1, 2, 3\}$, the $\phi_i$-facts are those that belong to the relation $R_i$, and the ones that are consistent with $\mathbf{a}$ are those of the form $R_i(j, \diamond)$; the solution $J_{\phi_i}$ is therefore obtained from $I_3$ by removing from $R_i$ all the facts except for $R_i(i, \square)$. Thus, we have $Q_3^\star(J_{\phi_1}) = \{(\square, \diamond, \diamond)\}$, $Q_3^\star(J_{\phi_2}) = \{(\diamond, \square, \diamond)\}$, and $Q_3^\star(J_{\phi_3}) = \{(\diamond, \diamond, \square)\}$. It follows that $|Q_3^\star(J_{\phi_i})| = 1$ for all $i \in \{1, 2, 3\}$, and therefore, the trivial algorithm can return any $J_{\phi_i}$ (depending on the traversal order over the atoms). □

The following (straightforward) proposition states the correctness and efficiency of the trivial algorithm. Unless stated otherwise, in this paper we use *data complexity* [26] for analyzing the complexity of deletion propagation and algorithms thereof; this means that the CQ $Q$ is held fixed, and the input consists of the instance $I$ and the tuple $\mathbf{a}$.

Proposition 3.2. *Trivial$\langle Q\rangle$ returns a solution, and terminates in polynomial time.*

Next, we show that in the case where $Q$ has no self joins, the trivial algorithm approximates MaxDP$\langle Q\rangle$ within a constant ratio that depends only on $Q$. (A discussion on CQs with self joins is in Section 6.)

## 3.1 Approximation

The following proposition shows that, if $Q$ is a CQ without self joins, then the trivial algorithm is a constant-factor approximation for MaxDP$\langle Q\rangle$ (where the constant depends on $Q$). The proof is fairly straightforward.

Proposition 3.3. *If $Q$ is a CQ without self joins, then Trivial$\langle Q\rangle$ is a $\frac{1}{k}$-approximation for* MaxDP$\langle Q\rangle$*, where $k = \min\{\text{arity}(Q), |\text{atoms}(Q)|\}$.*

Next, we will show that $\frac{1}{k}$ is also a tight lower bound on the approximation ratio of the trivial algorithm, for some queries without self joins. More precisely, we will show that for all natural numbers $k$, the CQ $Q_k^\star$ (Example 2.1), which satisfies arity($Q_k^\star$) $= |\text{atoms}(Q)| = k$, is such that the trivial

| $R_1$ | |
|---|---|
| 1 | $\diamond$ |
| 2 | $\diamond$ |
| 3 | $\diamond$ |
| 1 | $\square$ |

| $R_2$ | |
|---|---|
| 1 | $\diamond$ |
| 2 | $\diamond$ |
| 3 | $\diamond$ |
| 2 | $\square$ |

| $R_3$ | |
|---|---|
| 1 | $\diamond$ |
| 2 | $\diamond$ |
| 3 | $\diamond$ |
| 3 | $\square$ |

**Figure 2: Instance $I_3$ over schema$(Q_3^\star)$**

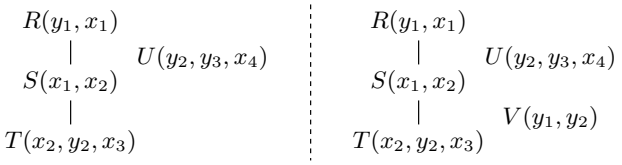


**Figure 3: The graphs $\mathcal{G}_\exists(Q)$ and $\mathcal{G}_\exists(Q')$ for the CQs $Q$ and $Q'$ of Example 4.2**

algorithm returns no better than a $\frac{1}{k}$-approximation. So, let $k$ be a natural number. We will construct an instance $I_k$ and a tuple $\mathbf{a} \in Q_k^\star(I_k)$ that realize the $\frac{1}{k}$ ratio. Each of the $k$ relations $R_i$ contains the $k$ tuples $(1, \diamond), \ldots, (k, \diamond)$; in addition, each relation $R_i$ contains the tuple $(i, \square)$. As an example, Figure 2 shows $I_3$. The tuple $\mathbf{a}$ is $\diamond^k$ (i.e., a tuple that consists of $k$ diamonds).

To see that $I_k$ and $\mathbf{a}$ are as desired, let $J$ be the sub-instance of $I_k$ that is obtained by removing $(i, \diamond)$ from each relation $R_i$. Then $Q_k^\star(J)$ contains $k$ tuples $\mathbf{b}_1, \ldots, \mathbf{b}_k$, where each $\mathbf{b}_i$ is the tuple that comprises only $\diamond$s, except for the $i$th element that is $\square$. Note that $J$ is optimal, since $Q_k^\star(I_k) = Q_k^\star(J) \cup \{\mathbf{a}\}$. Nevertheless, the trivial algorithm will remove from one of the relations, say $R_i$, all the tuples of the form $(x, \diamond)$, as described for $k = 3$ in Example 3.1. By doing so, the trivial algorithm produces a sub-instance $J'$, such that $Q_k^\star(J')$ contains exactly one tuple, thus no better than a $\frac{1}{k}$-approximation.

In the next section we will characterize the CQs, among those without self joins, for which the trivial algorithm is optimal. Furthermore, we will show that (in the absence of self joins) the trivial algorithm is optimal for MaxDP$\langle Q\rangle$ *precisely* for those CQs $Q$ with a tractable MaxDP$\langle Q\rangle$; for the remaining CQs $Q$, MaxDP$\langle Q\rangle$ is hard, and even hard to approximate better than some constant ratio.

# 4. DICHOTOMY

In this section, we define the *head-domination* property of a CQ, and show that if a CQ $Q$ without self joins has this property, then the trivial algorithm (optimally) solves MaxDP$\langle Q\rangle$. Furthermore, we show that this property *exactly* captures the tractability of MaxDP$\langle Q\rangle$, in the sense that in the absence of this property, MaxDP$\langle Q\rangle$ is not only intractable, but actually cannot be approximated better than some constant ratio (it is APX-hard).

## 4.1 Head Domination

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 $\{\phi_1, \phi_2\}$ whenever $\phi_1$ and $\phi_2$ have at least one existential variable in common (that is, $\mathsf{Var}_\exists(\phi_1) \cap \mathsf{Var}_\exists(\phi_2) \neq \emptyset$). Let $\phi$ be an atom of $Q$, and let $P$ be a set of atoms of $Q$. We say that $P$ is *head-dominated* by $\phi$ if $\phi$ contains all the head variables that occur in $P$ (i.e., $\mathsf{Var}_\mathsf{h}(\phi') \subseteq \mathsf{Var}_\mathsf{h}(\phi)$ for all $\phi' \in P$).

Definition 4.1. (**Head Domination**) A CQ $Q$ has the *head-domination* property if for every connected component $P$ of $\mathcal{G}_\exists(Q)$ there is an atom $\phi \in$ atoms$(Q)$, such that $P$ is head-dominated by $\phi$. $\square$

As an example, for the CQ $Q_k^\star$ defined in (3), the graph $\mathcal{G}_\exists(Q_k^\star)$ is a clique over atoms$(Q_k^\star)$, and so it has only one connected component. For $k > 1$, none of the atoms of $Q_k^\star$ contains all the head variables $y_i$, and hence, $Q_k^\star$ does not have the head-domination property. (Note that $Q_1^\star$ has the head-domination property, as does every CQ with only one atom or only one head variable.) Another example follows.

Example 4.2. Consider the CQ $Q$ defined by

$$Q(y_1, y_2, y_3) :- R(y_1, x_1), S(x_1, x_2), T(x_2, y_2, x_3), U(y_2, y_3, x_4)\,.$$

The left side of Figure 3 shows the graph $\mathcal{G}_\exists(Q)$. Note that $\mathcal{G}_\exists(Q)$ has two connected components: the first component is $\{R(y_1, x_1), S(x_1, x_2), T(x_2, y_2, x_3)\}$ and the second is $\{U(y_2, y_3, x_4)\}$. The CQ $Q$ does not have the head-domination property, since the first connected component is not head-dominated by any atom of $Q$ (since no atom contains both $y_1$ and $y_2$). Now, consider the following CQ $Q'$, which is obtained from $Q$ by adding an atom $V(y_1, y_2)$.

$$Q'(y_1, y_2, y_3) :- R(y_1, x_1), S(x_1, x_2), T(x_2, y_2, x_3), U(y_2, y_3, x_4), V(y_1, y_2)$$

The graph $\mathcal{G}_\exists(Q')$ is shown on the right of Figure 3. Note that the atom $V(y_1, y_2)$ head-dominates the connected component $\{R(y_1, x_1), S(x_1, x_2), T(x_2, y_2, x_3)\}$; hence, $Q'$ has the head-domination property. $\square$

## 4.2 Optimality of the Trivial Algorithm

The following theorem states that if a CQ $Q$ without self joins has the head-domination property, then the trivial algorithm is optimal for MaxDP$\langle Q\rangle$.

Theorem 4.3. *Let $Q$ be a CQ without self joins. If $Q$ has the head-domination property, then* Trivial$\langle Q\rangle$ *optimally solves* MaxDP$\langle Q\rangle$.

Proof. Let $Q$ be a CQ without self joins, such that $Q$ has the head-domination property. Consider the input $I$ and $\mathbf{a}$ for MaxDP$\langle Q\rangle$. Let $J$ be any solution (e.g., an optimal solution). Let $f$ be a fact in $I \setminus J$. Note that such a fact $f$ must exist, since $\mathbf{a} \in Q(I)$ and $\mathbf{a} \notin Q(J)$. Assume, w.l.o.g., that $\mathbf{a} \in Q(J \cup \{f\})$; otherwise, $f$ can be added to $J$ without losing any tuple from $Q(J)$, and then we can choose another $f$, and so on. Let $\phi$ be the atom of $Q$, such that $f$ is a $\phi$-fact. Let $P_f$ be the connected component of $\phi$ in $\mathcal{G}_\exists(Q)$, and let $\gamma$ be an atom of $Q$ such that $\gamma$ dominates $P_f$. Such an atom $\gamma$ exists, since $Q$ has the head-domination property.

Consider the solution $J_\gamma$ that the trivial algorithm constructs (by removing from $I$ all the $\gamma$-facts that are consistent with $\mathbf{a}$). We will prove the theorem by showing that $Q(J) \subseteq Q(J_\gamma)$ (and hence, $|Q(J)| \leq |Q(J_\gamma)|$). If $J \subseteq J_\gamma$, then the claim is obvious (since $Q$ is a monotonic query). Otherwise, let $g$ be a fact in $J \setminus J_\gamma$. Then $g$ is a $\gamma$-fact that is consistent with $\mathbf{a}$, since $g \notin J_\gamma$. We will prove that there

is no match $\mu$ for $Q$ in $J$, such that $\mu(\gamma) = g$. Since $Q$ has no self joins, this would mean that $g$ is actually useless for producing $Q(J)$. As a result, by repeating this argument for all $g \in J \setminus J_\gamma$ we will get that for $J' = J \cap J_\gamma$ it holds that $Q(J) = Q(J')$; and since $Q(J') \subseteq Q(J_\gamma)$ (as $J' \subseteq J_\gamma$ and $Q$ is monotonic), it holds that $Q(J) \subseteq Q(J_\gamma)$.

Suppose, by way of contradiction, that $\mu_g$ is a match for $Q$ in $J$, such that $\mu_g(\gamma) = g$. We will show that $\mathbf{a} \in Q(J)$, and thereby obtain a contradiction to the fact that $J$ is a solution. Let $\mathbf{y}$ be the tuple of head variables of $Q$. Recall that $\mathbf{a} \in Q(J \cup \{f\})$, and let $\mu_f$ be a match for $Q$ in $J \cup \{f\}$ with $\mu_f(\mathbf{y}) = \mathbf{a}$. Note that $\mu_f(\phi) = f$. (Remember that $Q$ has no self joins.)

Observe that $\mu_f$ and $\mu_g$ agree on all the head variables of $P_\phi$, due to the fact that $\gamma$ contains all these variables, and due to the fact that $g$ is consistent with $\mathbf{a}$. Also note that no existential variable occurs both in $P_\phi$ and outside $P_\phi$, since $P_\phi$ is a connected component. It thus follows that $\mu_f$ and $\mu_g$ agree on *all* the variables that occur both inside $P_\phi$ and outside $P_\phi$.

We now construct a match $\mu$ for $Q$ in $J$, as follows: for each $z \in \mathsf{Var}(Q)$ we define $\mu(z) = \mu_g(z)$ if $z$ appears in $P_\phi$, and otherwise $\mu(z) = \mu_f(z)$. Recall that $\phi$ is in $P_\phi$, and hence, $\mu(\phi) = \mu_g(\phi)$; thus, $\mu(\phi)$ belongs to $J$ (and in particular, $\mu(\phi) \neq f$). It follows that $\mu$ is indeed a match for $Q$ in $J$. To complete the proof, we will show that $\mu(\mathbf{y}) = \mathbf{a}$. Suppose that $\mathbf{y} = (y_1, \ldots, y_k)$ and $\mathbf{a} = (a_1, \ldots, a_k)$. Let $i$ be an index in $\{1, \ldots, k\}$. If $y_i$ appears in $P_\phi$, then $\mu(y_i) = a_i$, since $\mu(y_i) = \mu_g(y_i)$, $\mu_g(\gamma) = g$, and $g$ is consistent with $\mathbf{a}$. If $y_i$ does not appear in $P_\phi$, then $\mu(y_i) = a_i$ again, since $\mu(y_i) = \mu_f(y_i)$ and $\mu_f(\mathbf{y}) = \mathbf{a}$. We conclude that $\mu(\mathbf{y}) = \mathbf{a}$, and hence, $\mathbf{a} \in Q(J)$, as claimed. □

As a simple consequence of Theorem 4.3, if $Q$ is a CQ without self joins, and every join variable of $Q$ is a head variable, then the trivial algorithm optimally solves MaxDP$\langle Q \rangle$ (since then $\mathcal{G}_\exists(Q)$ has no edges). Actually, we can show that if every head variable is a join variable, then this statement is true even without requiring lack of self joins. However, in Section 6 we show that Theorem 4.3 is not correct for all the CQs that have self joins. Specifically, we show an example of a CQ $Q$ with self joins, such that $Q$ has the head-domination property, and yet the trivial algorithm does not optimally solve MaxDP$\langle Q \rangle$; even more, for that $Q$ we show that MaxDP$\langle Q \rangle$ is hard to approximate better than some constant ratio.

## 4.3 Hardness

The following theorem[3] states that if a CQ $Q$ without self joins does not have the head-domination property, then in contrast to Theorem 4.3, MaxDP$\langle Q \rangle$ is hard, and even hard to approximate better than some constant ratio. In Section 4.3.1 we discuss the proof.

THEOREM 4.4. *Assume $P \neq NP$, and let $Q$ be a CQ without self joins. If $Q$ does not have the head-domination property, then there is a constant $\alpha_Q \in (0,1)$, such that* MaxDP$\langle Q \rangle$ *cannot be $\alpha_Q$-approximated in polynomial time.*

[3]This result was found after the submission of the reviewed version of this paper, and was added (with the consent of the program committee) after the paper was accepted for publication. In the reviewed version, a similar result was shown for a more restricted class of CQs without self joins.

One may wonder whether, in Theorem 4.4, it is necessary to have $\alpha_Q$ depending on $Q$. In other words, does Theorem 4.4 hold for a global $\alpha$ that is applicable to all the CQs without self joins? The following theorem shows that the answer is positive if we are restricted to the class of *acyclic* CQs [3, 12] and the class of CQs over binary[4] relation symbols. The proof is, essentially, through insights on the reductions used for proving Theorem 4.4 (see Section 4.3.1) in these special cases.

THEOREM 4.5. *There is a constant $\alpha \in (0,1)$, such the following holds for every CQ $Q$ without self joins, provided that $Q$ is acyclic or* schema$(Q)$ *consists of binary relation symbols. If $Q$ does not have the head-domination property, then it is NP-hard to $\alpha$-approximate* MaxDP$\langle Q \rangle$.

However, in contrast to Theorem 4.5, the following proposition shows that no such global $\alpha$ exists for the class of all the CQs without self joins.

PROPOSITION 4.6. *For all natural numbers $k > 1$, there exists a CQ $Q_k$ with the following properties.*

1. *$Q_k$ has no self joins.*
2. *$Q_k$ does not have the head-domination property.*
3. $\mathsf{Trivial}\langle Q_k \rangle$ *is a $(1 - \frac{1}{k})$-approximation for* MaxDP$\langle Q_k \rangle$.

PROOF. We define $Q_k$ as follows. The schema of $Q_k$ has $k$ relation symbols $R_1, \ldots, R_k$, where each $R_i$ is $k$-ary. Define

$$Q_k(y_1, \ldots, y_k) :- \phi_1, \phi_2, \ldots, \phi_k$$

where, for $i = 1, \ldots, k$, the atom $\phi_i$ is given by

$$\phi_i = R_i(x, y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_k)\,.$$

Note that $\phi_i$ contains all the variables of $Q$, except for $y_i$. Clearly, $Q$ satisfies Properties 1 and 2 (i.e., $Q$ has no self joins and it violates the head-domination property). It is left to prove Property 3. Let $I$ and $\mathbf{a}$ be input for MaxDP$\langle Q_k \rangle$. Let $i$ and $j$ be such that $1 \leq i < j \leq k$, and let $f_i$ and $f_j$ be a $\phi_i$-fact and a $\phi_j$-fact, respectively, that are consistent with $\mathbf{a}$ (where consistency is defined in Section 3). The only answer of $Q(I)$ that agrees with both $f_i$ and $f_j$ on the head variables is $\mathbf{a}$. Therefore, among the answers in $Q(I) \setminus \{\mathbf{a}\}$, those that require $f_i$ are disjoint from those that require $f_j$. Hence, if we take the best $J_\phi$ constructed by $\mathsf{Trivial}\langle Q_k \rangle$ (Figure 1), it must be the case that $Q(J_\phi)$ misses at most $\frac{1}{k}$ of the tuples in $Q(I) \setminus \{\mathbf{a}\}$. In particular, the best $J_\phi$ is a $(1 - \frac{1}{k})$-approximation, as claimed. □

Next, we discuss the proof of Theorem 4.4.

### 4.3.1 Proving Theorem 4.4

The full proof of Theorem 4.4 is in the extended version of this paper [21]. Here, we give a brief overview of the proof.

Our first step is to show hardness of $Q_2^\star$ (defined in (2)), which is a special case of a CQ that does not have the head-domination property. Buneman et al. [4] showed that deciding whether there is a solution that is side-effect free is NP-complete for $Q_2^\star$. To show that, they described a reduction from *non-mixed 3-satisfiability*, which is the problem of deciding on the satisfiability of a formula in 3-CNF,

[4]This theorem further extends to the class of CQs where each atom has at most two influential variables, where an *influential variable* is either a join or a head variable

where no clause contains both negated and non-negated variables (that is, each clause is the disjunction of three literals, where the three are either all negative or all positive). Guruswami [17] showed a constant-factor bound on the approximability of non-mixed 3-satisfiability. However, we cannot simply combine the reduction of Buneman et al. and the result of Guruswami, since that reduction does not preserve approximation (or PTAS). Nevertheless, we prove inapproximability by combining that reduction with a more recent result of Guruswami and Khot [18] showing constant-factor inapproximability for non-mixed 3-satisfiability in the case where each variable occurs in at most five clauses. Thus, we get the following.

LEMMA 4.7. *There is a constant $\alpha \in (0,1)$, such that* $\text{MAXDP}\langle Q_2^\star\rangle$ *is NP-hard to $\alpha$-approximate.*

Next, we fix a CQ $Q$, such that $Q$ has neither self joins nor the head-domination property. Our goal is to prove that $\text{MAXDP}\langle Q\rangle$ is hard to approximate within some factor $\alpha_Q$. We fix a component $P$ of $\mathcal{G}_\exists(Q)$, such that $P$ is head-dominated by none of the atoms of $Q$. We call two variables $y$ and $y'$ *atomic neighbors* if $Q$ has an atom that contains both $y$ and $y'$. An important idea in the proof is to distinguish between two cases. The first case is where two of the head variables of $P$ are *not* atomic neighbors. The second case is the complement: every two head variables of $P$ are atomic neighbors.

In the first case, suppose that $y_1'$ and $y_2'$ are head variables of $P$ that are not atomic neighbors. We show a (fairly simple) reduction from $\text{MAXDP}\langle Q_2^\star\rangle$ to $\text{MAXDP}\langle Q\rangle$ where, roughly speaking, the variable $y_i'$ $(i = 1, 2)$ of $Q$ *simulates* the variable $y_i$ of $Q_2^\star$.

In the second case, we again show a reduction from the problem $\text{MAXDP}\langle Q_2^\star\rangle$ to $\text{MAXDP}\langle Q\rangle$, and we again find $y_1'$ and $y_2'$ in $Q$ that can simulate $y_1$ and $y_2$, respectively, in $Q_2^\star$; but here this task is much more subtle. In essence, for the proof to work we need to be able to assume that in a solution, it is not worth to remove any $\phi$-fact if $\phi$ contains both $y_1'$ and $y_2'$. To do that, we choose $y_1'$ and $y_2'$ carefully, and we handle differently those $\phi$ that are *inside* $P$ and those that are *outside* $P$. By *handling* $\phi$ we essentially augment the constructed instance $I$ (over schema$(Q)$) with additional tuples; this is also subtle, since we need to make sure that not too many answers are added to $Q(I)$, or else we can easily lose the (rough) preservation of the approximation ratio in the reduction. Needed for facing the last problem is the fact that every two head variables in $P$ are atomic neighbors.

## 4.4 Dichotomy

We summarize this section with the following dichotomy[3] that is obtained by combining Theorem 4.3 and 4.4.

THEOREM 4.8 (DICHOTOMY). *For a CQ $Q$ without self joins, one of the following holds.*

1. *The trivial algorithm optimally solves* $\text{MAXDP}\langle Q\rangle$ *in polynomial time.*
2. *There is a constant $\alpha_Q \in (0,1)$, such that it is NP-hard to $\alpha_Q$-approximate* $\text{MAXDP}\langle Q\rangle$.

*Moreover, 1 holds if and only if $Q$ has the head-domination property.*

A proof similar to (actually, simpler than) that of Theorem 4.8 gives a similar dichotomy for the problem of testing whether there is a solution that is side-effect free (i.e., a solution $J$ such that $Q(J) = Q(I) \setminus \{\mathbf{a}\}$), which has been studied by Buneman et al. [4]. Specifically, for a CQ $Q$ without self joins, testing whether there is a side-effect-free solution is in polynomial time when $Q$ has the head-domination property (due to Theorem 4.3), and is NP-complete otherwise.

# 5. APPROXIMATIONS FOR STAR CQS

A CQ is a *star CQ* if every join variable occurs in every atom; in other words, a CQ $Q$ is a star CQ if there is a set $Z \subseteq \mathsf{Var}(Q)$, such that $Z = \mathsf{Var}(\phi_1) \cap \mathsf{Var}(\phi_2)$ whenever $\phi_1, \phi_2 \in Q$ and $\phi_1 \neq \phi_2$ (note that $Z$ can be empty). As an example, every $Q_k^\star$ (defined in (3)) is a star CQ (with $Z = \{x\}$). Furthermore, every CQ with two or fewer atoms is a star CQ. An additional example is the following.

$$Q_1(y_1, y_2, y_3) :- R(x_1, y_1), S(y_1, x_1, y_2), T(x_3, y_1, x_1, x_1)$$

Note that in $Q_1$, the join variables are $x_1$ and $y_1$, and they indeed occur in each of the three atoms.

In this section, we present approximation algorithms for the problems $\text{MAXDP}\langle Q\rangle$, where $Q$ is a star CQ without self joins. The following corollary of Theorem 4.5 (for the case of acyclic CQs) shows that star CQs are intractable to approximate within some factor $\alpha$, except for trivial cases.

COROLLARY 5.1. *There is a number $\alpha \in (0,1)$, such that the following holds for all star CQs $Q$ without self joins. If every join variable is a head variable, or one of the atoms contains all the head variables, then* $\mathsf{Trivial}\langle Q\rangle$ *optimally solves* $\text{MAXDP}\langle Q\rangle$*; otherwise, $\alpha$-approximating* $\text{MAXDP}\langle Q\rangle$ *is NP-hard.*

The constant factor $\alpha$ that we found for the hardness part Theorem 5.1 is fairly close to 1 (it is larger than 0.9), so this result does not preclude good approximations (though it does preclude PTAS algorithms). Recall from Proposition 3.3 that for every CQ $Q$ without self joins, $\text{MAXDP}\langle Q\rangle$ is $\frac{1}{k}$-approximated using the trivial algorithm, where $k = \min\{\text{arity}(Q), |\text{atoms}(Q)|\}$. In this section, we give constant-factor approximation algorithms for $\text{MAXDP}\langle Q\rangle$, where the factor does not depend on $Q$, assuming that $Q$ is a star CQ without self joins. Towards the end of this section, we will show how to extend one of the approximations to a significant generalization of star CQs.

## 5.1 A Greedy Algorithm

We now present a $\frac{1}{2}$-approximation for $\text{MAXDP}\langle Q\rangle$, for the case where $Q$ is a star CQ without self joins. We first consider the special case where $Q$ is the CQ $Q_k^\star$ (for some $k > 0$). Later on, we will discuss the extension of the algorithm to the general case.

Fix a natural number $k > 0$. The goal is to approximate $\text{MAXDP}\langle Q_k^\star\rangle$. We call the approximation algorithm we present here the *greedy approximation*, and denote it by $\mathsf{Greedy}\langle k\rangle$. Figure 4 shows pseudo-code for the algorithm. The input includes an instance $I$ and a tuple $\mathbf{a} = (a_1, \ldots, a_k)$, and the algorithm returns a solution $J$.

We use the following notation. For a tuple $\mathbf{d} = (d_1, \ldots, d_k)$, we call a constant $b \in \mathsf{Const}$ a $\mathbf{d}$-*joining constant* if $R_i(b, d_i)$ is a fact of $I$ for all $i = 1, \ldots, k$ (i.e., there is a match $\mu$ for $Q_k^\star$ in $I$, such that $\mu(\mathbf{y}) = \mathbf{d}$ and $\mu(x) = b$).

**Algorithm** Greedy$\langle k\rangle(I, \mathbf{a})$

1: let $\mathbf{a} = (a_1, \ldots, a_k)$
2: $J \leftarrow I$
3: **for all** $\mathbf{a}$-joining constants $b$ **do**
4: **if** exists $j$ where $R_j(b, c) \in I$ for some $c \neq a_j$ **then**
5: $i \leftarrow$ a minimal $j$ with $R_j(b, c) \in I$ for some $c \neq a_j$
6: **else**
7: $i \leftarrow$ an arbitrary number in $\{1, \ldots, k\}$
8: delete $R_i(b, a_i)$ from $J$
9: **return** $J$

**Figure 4: Greedy approximation for MaxDP$\langle Q_k^\star\rangle$**

The algorithm Greedy$\langle k\rangle$ starts with $J = I$. It traverses over all the $\mathbf{a}$-joining constants $b$. For each $b$, the fact $R_i(b, a_i)$ is deleted from $J$, where $i$ is chosen to be the minimal $j$ such that $I$ contains a fact $R_j(b, c)$ for some $c \neq a_j$; if no such $j$ exists, then $i$ is chosen arbitrarily among $\{1, \ldots, k\}$.

The following lemma states that the algorithm returns a solution. The proof is immediate from the fact that for each $\mathbf{a}$-joining constant $b$, the returned instance $J$ misses $R_i(b, a_i)$ for some (actually, for exactly one) $i \in \{1, \ldots, k\}$.

LEMMA 5.2. Greedy$\langle k\rangle(I, \mathbf{a})$ *returns a solution.*

Next, we show that Greedy$\langle k\rangle$ is a 2-approximation for MaxDP$\langle Q_k^\star\rangle$. Fix the input $I$ for the algorithm, and let $J$ be the returned sub-instance of $I$. We will show that $|Q_k^\star(J)| \geq |Q_k^\star(I) \setminus \{\mathbf{a}\}|/2$. This implies that Greedy$\langle k\rangle$ is a $\frac{1}{2}$-approximation regardless of the performance of an optimal algorithm, since $|Q_k^\star(J')| \leq |Q_k^\star(I) \setminus \{\mathbf{a}\}|$ holds for every solution $J'$. Moreover, this implies that there is always a solution that retains at least half of the original (non-$\mathbf{a}$) tuples of $Q_k^\star(I)$.

To show that $|Q_k^\star(J)| \geq |Q_k^\star(I) \setminus \{\mathbf{a}\}|/2$, we use the following counting argument. We map the surviving answers to deleted answers in a way that each deleted answer is the image of some surviving answer. Since no surviving answer has two images, this will imply that the number of surviving answers is at least as large as the number of deleted answers. The intuitive reason for the existence of such a mapping is that we try to delete only facts $R_i(b, a_i)$ where there is an alternative fact $R_i(b, c)$ for the same joining constant $b$; the fact $R_i(b, c)$ will preserve some answers to account for those that have been deleted.

Formally, we define a function $\Psi : Q_k^\star(J) \to \mathsf{Const}^k$, and show that $\Psi$ is onto $(Q_k^\star(I) \setminus \{\mathbf{a}\}) \setminus Q_k^\star(J)$. The function $\Psi$ is defined as follows. For a tuple $\mathbf{c} \in Q_k^\star(J)$, where $\mathbf{c} = (c_1, \ldots, c_k)$, the tuple $\Psi(\mathbf{c})$ is obtained from $\mathbf{c}$ by choosing the minimal $i$ with $c_i \neq a_i$, and replacing that occurrence of $c_i$ with $a_i$. For example, for $k = 4$ and $\mathbf{a} = (\diamond, \diamond, \diamond, \diamond)$, the tuple $\Psi(\diamond, \heartsuit, \square, \diamond)$ is $(\diamond, \diamond, \square, \diamond)$.

LEMMA 5.3. $\Psi$ *is onto* $(Q_k^\star(I) \setminus \{\mathbf{a}\}) \setminus Q_k^\star(J)$.

PROOF. Let $\mathbf{d} = (d_1, \ldots, d_k)$ be a tuple of $Q_k^\star(I)$, such that $\mathbf{d}$ is neither $\mathbf{a}$ nor in $Q_k^\star(J)$. We need to show that there is some $\mathbf{c} \in Q_k^\star(J)$, such that $\Psi(\mathbf{c}) = \mathbf{d}$. Let $b$ be a $\mathbf{d}$-joining constant ($b$ exists since $\mathbf{d} \in Q_k^\star(I)$). The fact that $\mathbf{d} \notin Q_k^\star(J)$ implies that $b$ is an $\mathbf{a}$-joining constant, and that when $b$ was visited we deleted one of the $R_i(b, d_i)$. Consider such an index $i$, and note that $d_i = a_i$ follows from the definition of the algorithm. Now, $\mathbf{d} \neq \mathbf{a}$ implies that, in the iteration of $b$, the condition of Line 4 is true, and hence, the deleted fact $R_i(b, d_i)$ satisfies that $i$ is the minimal $j$ where $R_i(b, c) \in I$ for some $c \neq a_i$. Hence, we must have $d_j = a_j$ for all $j < i$ (and recall that we also have $d_i = a_i$). Let $c \neq a_i$ be a constant such that $R_i(b, c) \in I$, and let $\mathbf{c}$ be obtained from $\mathbf{d}$ by replacing the $i$th element, $d_i$, with $c$. We have that $\mathbf{c} \in Q_k^\star(J)$ and, moreover, $\Psi(\mathbf{c}) = \mathbf{d}$, as required. □

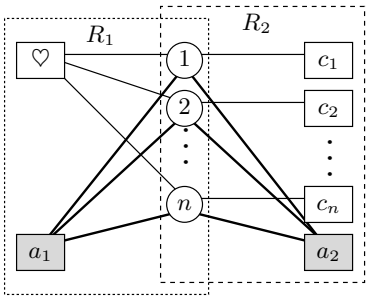


**Figure 5: An instance $I$ where Greedy$\langle k\rangle(I, \mathbf{a})$ provides a $(\frac{1}{2} + \frac{1}{2n})$-approximation**

From Lemmas 5.2 and 5.3 we get the following theorem.

THEOREM 5.4. Greedy$\langle k\rangle$ *is a* $\frac{1}{2}$*-approximation for the problem* MaxDP$\langle Q_k^\star\rangle$.

Next, we show that in the worst case, Greedy$\langle k\rangle$ indeed gives just $\frac{1}{2}$-approximation. Our example is for $k = 2$, and it is depicted in Figure 5. The relation $R_1$ contains the tuples $(i, a_1)$ and $(i, \heartsuit)$, for all $i \in \{1, \ldots, n\}$. Note that the visual position of nodes on the edges that correspond to $R_1$ is in opposition to the order of the values in $R_1$ (i.e., the edge from $\heartsuit$ to 2 corresponds to the fact $R_1(2, \heartsuit)$). The relation $R_2$ contains the tuples $(i, a_2)$ and $(i, c_i)$ for all $i \in \{1, \ldots, n\}$. The greedy algorithm will remove all the facts $R_1(i, a_1)$, generating a sub-instance $J$ with $|Q_2^\star(J)| = n + 1$. On the other hand, one could remove all the facts $R_2(i, a_2)$, and then get a solution $J'$, such that $|Q_2^\star(J')| = 2n$. Hence, the approximation ratio is at most $\frac{1}{2} + \frac{1}{2n}$.

Finally, we discuss the generalization of the greedy algorithm to general star CQs without self joins. Let $Q$ be a star CQ without self joins. Suppose that $Q$ has $k$ atoms. We reduce MaxDP$\langle Q\rangle$ to MaxDP$\langle Q_k^\star\rangle$ in an approximation-preserving manner. Specifically, given the input $I$ and $\mathbf{a}$ for MaxDP$\langle Q\rangle$, we generate an instance $I'$ over schema$(Q_k^\star)$ and a tuple $\mathbf{a}'$, and apply Greedy$\langle k\rangle(I', \mathbf{a}')$ to get a solution $J'$ for $I'$ and $\mathbf{a}'$. Finally, we transform $J'$ into a solution $J$ for $I$ and $\mathbf{a}$. This reduction is fairly straightforward, and the details are in the extended version of this paper [21]. In conclusion, we get the following theorem.

THEOREM 5.5. *Let $Q$ be a star CQ without self joins. There is a* $\frac{1}{2}$*-approximation for* MaxDP$\langle Q\rangle$*, where the running time is polynomial under query-and-data complexity.*

Recall that *query-and-data complexity* means that the running time is measured as if the query $Q$ is given as part of

the input (in addition to $I$ and $\mathbf{a}$), and is not fixed. The polynomial running time under query-and-data complexity is due to the fact that both $\mathsf{Greedy}\langle k\rangle$ and the reduction from $\textsc{MaxDP}\langle Q\rangle$ to $\textsc{MaxDP}\langle Q_k^\star\rangle$ take polynomial time under query-and-data complexity. In contrast, recall that the trivial algorithm is polynomial time only under data complexity (since it requires the evaluation of $Q$ over the $J_\phi$).

## 5.2 A Randomized-Rounding Algorithm

In this section, we describe a *randomized-rounding* algorithm for approximating $\textsc{MaxDP}\langle Q\rangle$ when $Q$ is a star CQ without self joins. We will show that this algorithm gives an approximation ratio that is higher than $\mathsf{Greedy}\langle k\rangle$, namely, $1-\frac{1}{e}$ (which is, roughly, 0.632) instead of $\frac{1}{2}$; the running time is still polynomial, but it is not as fast as $\mathsf{Greedy}\langle k\rangle$. In particular, the algorithm we describe here will terminate in polynomial time under (the usual) data complexity, but not under query-and-data complexity. At the end of this section, we will show that this algorithm can be used to approximate CQs from a class that significantly generalizes the star CQs without self joins.

More precisely, the algorithm we present is a randomized $(1-\frac{1}{e})$-approximation, where a *randomized $\alpha$-approximation* for $\textsc{MaxDP}\langle Q\rangle$ is a randomized algorithm that, given $I$ and $\mathbf{a}$, returns a solution $J$ such that the expected $|Q(J)|$ is at least $\alpha\cdot|Q(K)|$ for all solutions $K$. Put differently, if $J_{\text{opt}}$ is an optimal solution, then $\mathbb{E}[|Q(J)|]\geq\alpha\cdot|Q(J_{\text{opt}})|$. This is a standard notion of randomized optimization (e.g., [16, 19]). Such a randomized algorithm can be easily transformed into a randomized algorithm that returns an $\alpha$-approximation, where $\alpha$ is arbitrarily close to $(1-\frac{1}{e})$, and the error probability is arbitrarily small. We further note that the algorithm we present here can be derandomized into a deterministic (i.e., ordinary) $(1-\frac{1}{e})$-approximation, using the *pipage-rounding* technique of Călinescu et al. [5]; however, that derandomization is beyond the scope of this paper.

Our algorithm uses ideas from the framework of *submodular maximization subject to a matroid constraint* [5, 6]. In fact, our problem can be formally reduced to the problem of maximizing a monotone submodular function subject to a matroid constraint, for which a $(1-\frac{1}{e})$-approximation has been developed recently [6]. However, it is simpler (and more instructive) to give a self-contained description, which is what we do in the remainder of this section.

$$\text{maximize} \sum_{\mathbf{d}\in Q_k^\star(I)} Y_{\mathbf{d}}$$

subject to

$$\forall \mathbf{d}\in Q_k^\star(I) \quad Y_{\mathbf{d}} \leq \sum_{b\in JC(\mathbf{d})} \sum_{i\in[k]\setminus(\mathbf{d}\sqcap\mathbf{a})} X_i^b$$

$$\forall b\in JC(\mathbf{a}) \quad \sum_{i=1}^{k} X_i^b = 1$$

$$\forall b\in JC(\mathbf{a}), i\in[k] \quad 0\leq X_i^b\leq 1$$

$$\forall \mathbf{d}\in Q_k^\star(I) \quad 0\leq Y_{\mathbf{d}}\leq 1$$

**Figure 6: The program $\text{LP}(I,\mathbf{a})$**

**Algorithm $\mathsf{RRLP}\langle k\rangle(I)$**

1: let $\mathbf{a}=(a_1,\ldots,a_k)$
2: $J\leftarrow I$
3: solve $\text{LP}(I,\mathbf{a})$
4: **for all** $\mathbf{a}$-joining constants $b$ **do**
5: independently pick a random $i\in[k]$ with probability $X_j^b$ for $j$
6: delete $R_i(b,a_i)$ from $J$
7: **return** $J$

**Figure 7: Randomized rounding for $\textsc{MaxDP}\langle Q_k^\star\rangle$**

As we did in the previous section, we will first describe the algorithm for the special CQs $Q_k^\star$ (defined in (3)), and consider general star CQs later. We fix the input $I$ and $\mathbf{a}=(a_1,\ldots,a_k)$. Recall that for a tuple $\mathbf{d}=(d_1,\ldots,d_k)$, a $\mathbf{d}$-joining constant is a constant $b\in\mathsf{Const}$ such that $R_i(b,d_i)\in I$ for all $i=1,\ldots,k$. We make the assumption that for each fact $R_i(b,d)$ of $I$, the constant $b$ is an $\mathbf{a}$-joining constant. There is no loss of generality in making this assumption, since there is no reason to delete a fact $R_i(b',d)$ if $b'$ is not an $\mathbf{a}$-joining constant; hence, the existence of such $R_i(b',d)$ can only increase the approximation ratio that we achieve by the algorithm. For a tuple $\mathbf{d}\in Q_k^\star(I)$, we denote by $JC(\mathbf{d})$ the set of all the $\mathbf{d}$-joining constants. Note that our assumption above implies that $JC(\mathbf{d})\subseteq JC(\mathbf{a})$ for all $\mathbf{d}\in Q_k^\star(I)$.

Let us first formulate $\textsc{MaxDP}\langle Q_k^\star\rangle$ as an integer linear program (LP). For every $\mathbf{a}$-joining constant $b$ and index $i\in\{1,\ldots,k\}$, we have the variable $X_i^b$ that gets values in $\{0,1\}$. We interpret $X_i^b=1$ as saying that the fact $R_i(b,a_i)$ should be removed. So, we will have the following constraints which ensure that the resulting sub-instance is a solution. Note that $[k]$ is a shorthand notation for $\{1,\ldots,k\}$.

$$\forall b\in JC(\mathbf{a}) \quad \sum_{i=1}^{k} X_i^b = 1$$

$$\forall b\in JC(\mathbf{a}), i\in[k] \quad X_i^b\in\{0,1\}$$

Next, we construct the objective function. Suppose that $\mathbf{d}=(d_1,\ldots,d_k)$ is a tuple in $Q_k^\star(I)$. We define the variable $Y_{\mathbf{d}}$ that gets the value 1 if $\mathbf{d}$ *survives* (i.e., belongs to $Q_k^\star(J)$), and 0 otherwise. For that, we denote by $\mathbf{d}\sqcap\mathbf{a}$ the set of indices $i\in[k]$, such that $d_i=a_i$. For example, if $\mathbf{a}=(\diamond,\diamond,\diamond,\diamond)$ and $d=(\diamond,\heartsuit,\square,\diamond)$, then $\mathbf{d}\sqcap\mathbf{a}=\{1,4\}$. So, we have the following constraint:

$$\forall \mathbf{d}\in Q_k^\star(I) \quad Y_{\mathbf{d}} = \min\left(1\ ,\ \sum_{b\in JC(\mathbf{d})}\ \sum_{i\in[k]\setminus(\mathbf{d}\sqcap\mathbf{a})} X_i^b\right)$$

Note that $\sum_{i\in[k]\setminus(\mathbf{d}\sqcap\mathbf{a})} X_i^b\geq 1$ means that the fact $R_i(b,a_i)$ that we delete is such that $a_i\neq d_i$, and hence, none of the $R_j(b,d_j)$ is deleted (and then $\mathbf{d}$ survives).

Finally, the goal is to maximize the sum of the $Y_{\mathbf{d}}$. Figure 6 shows the program $\text{LP}(I,\mathbf{a})$, which is the LP relaxation of the integer LP.

The algorithm for MaxDP$\langle Q_k^\star\rangle$, called RRLP$\langle k\rangle(I, \mathbf{a})$, is described by the pseudo-code of Figure 7. The algorithm first solves LP$(I, \mathbf{a})$, and as a result, gets an optimal (fractional) assignment for each $X_i^b$ and $Y_{\mathbf{d}}$. Like Greedy$\langle k\rangle(I, \mathbf{a})$, this algorithm constructs a sub-instance $J$ of $I$ and returns $J$ in the last line. Still similarly to Greedy$\langle k\rangle(I, \mathbf{a})$, for each $\mathbf{a}$-joining constant $b$, the algorithm selects an index $i \in [k]$ and deletes from $J$ the fact $R_i(i, a_i)$. The difference between the algorithms is in the way $i$ is chosen. Here, we apply the standard action in randomized rounding, namely, $i$ is picked randomly and independently from $[k]$, where the probability of the index $j$ is $X_j^b$. Note that the constraints of LP$(I, \mathbf{a})$ ensure that, for a specific $b$, the $X_i^b$ constitute a probability distribution over $[k]$.

Next, we prove the correctness of RRLP$\langle k\rangle(I, \mathbf{a})$. The following lemma shows that the algorithm returns a solution, and that its running time is polynomial. The proof is straightforward. In particular, as noted before Lemma 5.2 about Greedy$\langle k\rangle$, for each $\mathbf{a}$-joining constant $b$ the returned instance $J$ misses $R_i(b, a_i)$ for some (actually, for exactly one) $i \in [k]$.

Lemma 5.6. RRLP$\langle k\rangle(I, \mathbf{a})$ *returns a solution, and terminates in polynomial time.*

Next, we show that RRLP$\langle k\rangle(I)$ is a randomized $(1 - \frac{1}{e})$-approximation. The proof is based on the following lemma, which states that a tuple $\mathbf{d} \in Q(I)$ survives with a probability of at least $(1 - \frac{1}{e})Y_{\mathbf{d}}$.

Lemma 5.7. *Consider an execution of* RRLP$\langle k\rangle(I, \mathbf{a})$ *that results in a random solution $J$, and let $\mathbf{d} \in Q_k^\star(I)$.*

$$\Pr\left(\mathbf{d} \in Q_k^\star(J)\right) \geq (1 - \frac{1}{e})Y_{\mathbf{d}}\,.$$

Based on Lemma 5.7, we next prove that RRLP$\langle k\rangle(I)$ is indeed a randomized $(1 - \frac{1}{e})$-approximation.

Theorem 5.8. *Consider an execution of* RRLP$\langle k\rangle(I, \mathbf{a})$ *that results in a random solution $J$, and let $J_{\text{opt}}$ be an optimal solution. Then $\mathbb{E}[|Q_k^\star(J)|] \geq (1 - \frac{1}{e})|Q_k^\star(J_{\text{opt}})|$.*

Proof. Consider the execution of LP$(I, \mathbf{a})$ in Line 3 of RRLP$\langle k\rangle(I, \mathbf{a})$. Let $M$ be the sum $\sum_{\mathbf{d}\in Q_k^\star(I)} Y_{\mathbf{d}}$. W.l.o.g., we can assume that in $J_{\text{opt}}$ it holds that for all $\mathbf{a}$-joining constants $b$, exactly one $R_i(b, a_i)$ is missing. Then $J_{\text{opt}}$ defines a solution for LP$(I, \mathbf{a})$, and therefore, $M \geq |Q_k^\star(J_{\text{opt}})|$. On the other hand, from Lemma 5.7 and the linearity of expectation we have that

$$\begin{aligned}\mathbb{E}[|Q_k^\star(J)|] &= \sum_{\mathbf{d}\in Q_k^\star(I)} \Pr(\mathbf{d} \in Q_k^\star(J)) \\ &\geq (1 - \frac{1}{e}) \sum_{\mathbf{d}\in Q_k^\star(I)} Y_{\mathbf{d}} = (1 - \frac{1}{e})M\end{aligned}$$

Therefore, $\mathbb{E}[|Q_k^\star(J)|] \geq (1 - \frac{1}{e})|Q_k^\star(J_{\text{opt}})|$, as claimed. □

We now consider general star CQs without self joins. As we noted before Theorem 5.5, there is a simple approximation-preserving (and polynomial-time) reduction from the problem MaxDP$\langle Q\rangle$, where $Q$ is a star CQ without self joins, to MaxDP$\langle Q_k^\star\rangle$, where $k = |\text{atoms}(Q)|$. Hence, Theorem 5.8 immediately implies the following result.

Theorem 5.9. *Let $Q$ be a star CQ without self joins. There is a randomized $(1-\frac{1}{e})$-approximation for* MaxDP$\langle Q\rangle$*, with a polynomial running time.*

## 5.3 Beyond Star CQs

In this section, we extend Theorem 5.9 to the class of *existentially star* CQs, which generalizes the class of star CQs without self joins. Intuitively, in an existentially star CQ the "star requirement" is restricted to the existential variables of the query.

More formally, let $Q$ be a CQ. We denote by $\mathsf{Var}_\exists^{\bowtie}(Q)$ the set of existential join variables of $Q$. We say that $Q$ is *existentially star* if for every atom $\phi$ of $Q$, either every variable of $\mathsf{Var}_\exists^{\bowtie}(Q)$ occurs in $Q$, or none of them does.

Example 5.10. Consider the following CQ:

$$Q(y_1, y_2, y_3, y_4) :- R(x_1, y_1, y_2), S(x_1, y_2, y_3), T(y_3, y_1, x_2)$$

The CQ $Q$ is existentially star, since $\mathsf{Var}_\exists^{\bowtie}(Q) = \{x_1\}$, and every atom either contains $x_1$ or not. Actually, by the same argument, every CQ that has at most one existential join variable is existentially star. □

Next, we give a short overview of how we handle existentially star CQs without self joins. We denote by $Q_{|\exists}$ the CQ that comprises all the atoms $\phi$ of $Q$ having $\mathsf{Var}_\exists^{\bowtie}(Q) \subseteq \mathsf{Var}(\phi)$. Note that if $Q$ is existentially star, then $Q_{|\exists}$ is also existentially star. Also observe that the arity of $Q$ can be strictly larger than that of $Q_{|\exists}$. As an example, for the CQ $Q$ of Example 5.10, the CQ $Q_{|\exists}$ is:

$$Q_{|\exists}(y_1, y_2, y_3) :- R(x_1, y_1, y_2), S(x_1, y_2, y_3)$$

Note that the arity of $Q$ is 4, while that of $Q_{|\exists}$ is 3.

Let $Q$ be an existentially-star CQ $Q$ without self joins. Our general approach to approximating MaxDP$\langle Q\rangle$ is as follows. Instead of approximating MaxDP$\langle Q\rangle$, we approximate MaxDP$\langle Q_{|\exists}\rangle$; furthermore, instead of using the input tuple $\mathbf{a}$ of MaxDP$\langle Q\rangle$, for MaxDP$\langle Q_{|\exists}\rangle$ we use the restriction of $\mathbf{a}$ to the head variables of $Q_{|\exists}$. Finally, we view every occurrence of a head join variable as a distinct head variable, and thus assume that every join variable is existential. Hence, we treat $Q_{|\exists}$ as if it is a star CQ, and then we apply Theorem 5.9, to get a solution $J$.

There are two main problems with the above approach. First, by restricting to $Q_{|\exists}$ and ignoring the fact that head variables can be join variables, when solving MaxDP$\langle Q_{|\exists}\rangle$ we may end up saving tuples of $Q_{|\exists}(J)$ that do not give rise to *any* tuple of $Q(J)$, while eliminating tuples of $Q_{|\exists}(I)$ that give rise to *multiple* tuples of $Q(I)$. To handle that, we consider again the program LP$(I, \mathbf{a})$ of Figure 6, and observe that we can assign a *weight* $w(\mathbf{d})$ to each variable $Y_{\mathbf{d}}$. That is, the objective function can be as follows.

$$\text{maximize} \sum_{\mathbf{d}\in Q_k^\star(I)} w(\mathbf{d}) \cdot Y_{\mathbf{d}}$$

Indeed, we are able to handle the first problem by using proper weights $w(\mathbf{d})$. Let us call the final algorithm that reduces MaxDP$\langle Q\rangle$ to MaxDP$\langle Q_k^\star\rangle$ and uses weights as described above the *extended* RRLP$\langle k\rangle$.

Unfortunately, the extended RRLP$\langle k\rangle$ does not guarantee a proper approximation, due to a second problem, which is the following. By restricting to $Q_{|\exists}$ instead of $Q$, we ignore the $\phi$-facts of $I$ for $\phi \in \text{atoms}(Q) \setminus \text{atoms}(Q_{|\exists})$. But it may still be the case that deleting facts among those $\phi$-facts is necessary to obtain a proper approximate solution. As an example, consider the following existentially-star CQ.

$$Q(y_1, y_2, y_3) :- R_1(x, y_1), R_2(x, y_2), R_3(y_2, y_3)$$

Let $\mathbf{a}$ be the tuple $(\diamond, \diamond, \diamond)$, and let $I$ be the instance that consists of the facts $R_1(1, \diamond)$, $R_2(1, \diamond)$, $R_3(\diamond, \diamond)$, and $R_3(\diamond, \heartsuit)$. Note that $Q_{|\exists}$ is $Q_2^\star$. By removing the fact $R_3(\diamond, \diamond)$, the answer $(\diamond, \diamond, \heartsuit)$ survives. However, when trying to solve MaxDP$\langle Q_{|\exists}\rangle$ first, as described above, we necessarily lose every tuple of $Q(I)$ (and hence, do not get any constant-factor approximation).

The above problem can be solved by running the trivial algorithm in addition to the extended RRLP$\langle k\rangle$, and taking the maximum. More precisely, at least one of the following must hold for given input $I$ and $\mathbf{a}$.

1. The trivial algorithm returns an optimal solution for $I$ and $\mathbf{a}$.
2. The generalized RRLP$\langle k\rangle$ returns a $(1-\frac{1}{e})$-optimal solution, in expectation, for $I$ and $\mathbf{a}$.

Thus, we get the following result.

THEOREM 5.11. *Let $Q$ be an existentially-star CQ without self joins. There is a randomized $(1-\frac{1}{e})$-approximation for* MaxDP$\langle Q\rangle$*, with a polynomial running time.*

# 6. ABOUT SELF JOINS

In this section, we give a brief discussion on CQs with self joins. In particular, we show that results from Section 3 and 4 do not extend to (all the) CQs with self joins.

We first show that in Theorem 4.3, the requirement for lack of self joins is necessary. For that, we will show a CQ $Q$ that has the head-domination property, and input $I$ and $\mathbf{a}$ for MaxDP$\langle Q\rangle$, such that the trivial algorithm returns a solution that is not optimal. The CQ $Q$ is the following.

$$Q(y) :- R(x, y, x_1), R(x, x_2, y) \tag{4}$$

Let $I$ be the instance that consists of the facts $R(\diamond, 0, 1)$, $R(\diamond, 6, 0)$, $R(\diamond, 1, 8)$, $R(\square, 0, 7)$, $R(\square, 2, 0)$, and $R(\square, 9, 2)$. Let $\mathbf{a}$ be the tuple $(0)$. The trivial algorithm will take the best out of two: the solution $J_1$ that is obtained by deleting $R(\diamond, 0, 1)$ and $R(\square, 0, 7)$, and the solution $J_2$ that is obtained by deleting and $R(\diamond, 6, 0)$ and $R(\square, 2, 0)$. It is easy to show that $Q(J_1) = \{(2)\}$ and $Q(J_2) = \{(1)\}$. Hence, the solution $J$ returned by the trivial algorithm is such that $|Q(J)| = 1$. Now, the solution $J'$ that is obtained by deleting $R(\diamond, 6, 0)$ and $R(\square, 0, 7)$ satisfies $Q(J') = \{(1), (2)\}$, and thus $|Q(J')| > |Q(J)|$. Hence, the trivial algorithm is not optimal for MaxDP$\langle Q\rangle$, as claimed. In fact, the following theorem states that for this CQ, MaxDP$\langle Q\rangle$ is hard to approximate better than some constant ratio $\alpha$. The proof is by a reduction from maximum 3-satisfiability.

THEOREM 6.1. *For $Q(y) :- R(x, y, x_1), R(x, x_2, y)$, there is a constant $\alpha \in (0, 1)$, such that no polynomial-time algorithm $\alpha$-approximates* MaxDP$\langle Q\rangle$*, unless* P = NP*.*

Recall from Proposition 3.3 that for every CQ without self joins, there is a $\frac{1}{k}$-approximation for MaxDP$\langle Q\rangle$, where $k = \min\{\text{arity}(Q), |\text{atoms}(Q)|\}$. Next, we show that this result does not extend to all CQs with self joins, as for some of these CQs an exponential bound (in $k$) applies. Formally, we show the following.

THEOREM 6.2. *Assume* P $\neq$ NP*. For some constant $\alpha \in (0, 1)$, the following holds. For all $k > 0$ there exists a CQ $Q_k$, such that* $\text{arity}(Q_k) = k$*,* $|\text{atoms}(Q_k)| = 2k$*, and no polynomial-time algorithm $\alpha^k$-approximates* MaxDP$\langle Q_k\rangle$*.*

Next, we give a simple proof for Theorem 6.2. We first set some basic notation. Let $\mathbf{a}$ be a tuple, let $Q$ be a CQ, and let $k$ be a natural number. By $\mathbf{a}^k$ we denote the tuple that is obtained from $\mathbf{a}$ by concatenating it $k$ times. By $Q^k$ we denote the CQ that is obtained by concatenating $k$ copies of $Q$, where in each copy the variables are renamed into a set of distinct fresh variables. For example, for the CQ $Q$ of (4), the CQ $Q^2$ is the following.

$$\begin{aligned} Q^2(y, y') :- & R(x, y, x_1), R(x, x_2, y) \\ & R(x', y', x'_1), R(x', x'_2, y') \end{aligned}$$

More formally, for $Q(\mathbf{y}) :- \varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$, the CQ $Q^k$ is

$$Q(\mathbf{y}_1, \ldots, \mathbf{y}_k) :- \varphi(\mathbf{x}_1, \mathbf{y}_1, \mathbf{c}), \ldots, \varphi(\mathbf{x}_k, \mathbf{y}_k, \mathbf{c}),$$

where each $\mathbf{x}_i$ is of the arity of $\mathbf{x}$, each $\mathbf{y}_i$ is of the arity of $\mathbf{y}$, and $\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{y}_1, \ldots, \mathbf{y}_k$ are pairwise-disjoint vectors of variables.

Let $k$ be a natural number, let $Q$ be a CQ, and let $\mathbf{a}$ be a tuple in $\mathsf{Const}^{\text{arity}(Q)}$. Moreover, let $I$ be an instance, and let $J$ be a sub-instance of $I$. The proof is based on the following (straightforward) observations.

1. $\mathbf{a} \in Q(I)$ if and only if $\mathbf{a}^k \in Q^k(I)$.
2. $J$ is a solution (w.r.t. $Q$) for $I$ and $\mathbf{a}$ if and only if $J$ is a solution (w.r.t, $Q^k$) for $I$ and $\mathbf{a}^k$.
3. $|Q^k(I)| = |Q(I)|^k$.

We now prove Theorem 6.2. Fix a natural number $k$. We choose as $Q_k$ the CQ $Q^k$, where $Q$ is the CQ of (4) and $\alpha$ is the one of Theorem 6.1. Clearly, $Q_k$ satisfies $\text{arity}(Q_k) = k$ and $|\text{atoms}(Q_k)| = 2k$. Now, suppose that an algorithm $A$ is an $\alpha^k$-approximation for MaxDP$\langle Q_k\rangle$. So, given the input $I$ and $\mathbf{a}$ for MaxDP$\langle Q\rangle$, we feed $A$ with the input $I$ and $\mathbf{a}^k$. Let $J$ be the output of $A$. Then $J$ is a solution for $I$ and $\mathbf{a}$ (by Property 2 above). Let $J_{\text{opt}}$ be an optimal solution for $I$ and $\mathbf{a}$. From Properties 1–3 above it easily follows that $J_{\text{opt}}$ is also optimal for $I$ and $\mathbf{a}^k$. We obtain the following.

$$|Q(J)| = |Q_k(J)|^{\frac{1}{k}} \geq (\alpha^k |Q_k(J_{\text{opt}})|)^{\frac{1}{k}} = \alpha |Q(J_{\text{opt}})|$$

Thus, using $A$ we get an $\alpha$-approximation for MaxDP$\langle Q\rangle$, and then Theorem 6.2 follows immediately from Theorem 6.1. This concludes the proof.

# 7. CONCLUSIONS

We studied the complexity of MaxDP$\langle Q\rangle$ for CQs $Q$, and established several results for CQs without self joins. Among these results, we showed that for every $Q$, the problem MaxDP$\langle Q\rangle$ is $\alpha$-approximable (in polynomial time) for some constant $\alpha$ that is inversely proportional to the size of $Q$, and that MaxDP$\langle Q\rangle$ is solved optimally by the trivial algorithm if $Q$ has the head-domination property. We also showed a strong dichotomy: inapproximability when head domination does not hold. We gave approximation algorithms for star CQs, and for the more general existentially star CQs. Finally, we considered CQs with self joins, and showed an example of such a CQ $Q$, such that $Q$ has the head-domination property, and yet, MaxDP$\langle Q\rangle$ is inapproximable beyond some constant ratio; we also showed a family of CQs with self joins, where the approximability of MaxDP$\langle Q\rangle$ deteriorates exponentially with the size of $Q$.

Many related open problems and practically important challenges remain, and are left for future work. Among the open problems, perhaps most basic are the following.

1. Does a dichotomy in the spirit of Theorem 4.8 hold for the class of all CQs with self joins?
2. Is there a fixed ratio $\alpha \in (0, 1)$, independent of $Q$, such that MaxDP$\langle Q \rangle$ is $\alpha$-approximable in polynomial time for *every* CQ $Q$ without self joins?[5]
3. For the class of (existentially) star CQs, can we do better than Theorem 5.11? That is, is there a polynomial-time $\alpha$-approximation for some $\alpha > 1 - \frac{1}{e}$?

Practically important remaining challenges include complexity analyses, and algorithmic solutions in particular, for deletion propagation within the following generalizations.

- **Deleting multiple tuples.** Instead of one answer $\mathbf{a}$, in this generalization we are required to delete from the view a *set* of answers (and the goal is, as usual, to maximize $|Q(J)|$).
- **Incorporating database constraints.** In this generalization, database constraints are enforced. In the presence of constraints, deletion of tuples may incur following the deletion of others (e.g., in the case of *inclusion dependencies* [7] like foreign keys); thus, constraints are significant for deletion propagation [8].
- **Maximizing a different view.** In this generalization, when deleting the answer $\mathbf{a} \in Q(I)$, the goal is to maximize $|Q'(J)|$ for a CQ $Q'$ that is different from $Q$, or even for multiple CQs $Q'$ simultaneously.
- **Forbidden source deletions.** In our problem setting, every fact of the instance $I$ is a possible candidate for deletion. But in some cases, we may want to forbid the deletion of certain facts. For example, in IE it makes sense to forbid deletion from some clean relations like the English or country-name dictionary.[6]

As part of future work, we intend to explore whether and how the techniques we presented here for approximating MaxDP$\langle Q \rangle$ can help in developing algorithmic solutions within more general settings like the ones above.

## Acknowledgments

We are grateful to Laura Chiticariu, Ronald Fagin, Rajasekar Krishnamurthy and Wang Chiew Tan for fruitful discussions. We are also grateful to members of the PODS 2011 Program Committee for providing valuable comments and suggestions.

## 8. REFERENCES

[1] S. Balakrishnan, V. Chu, M. A. Hernández, H. Ho, R. Krishnamurthy, S. Liu, J. Pieper, J. S. Pierce, L. Popa, C. Robson, L. Shi, I. R. Stanoi, E. L. Ting, S. Vaithyanathan, and H. Yang. Midas: integrating public financial data. In *SIGMOD Conference*, pages 1187–1190. ACM, 2010.

[5]Recall that the approximation ratio of Proposition 3.3 depends on $Q$, and that Theorem 6.2 implies that no such $\alpha$ exists if self joins are allowed (unless P = NP).

[6]This distinction between forbidden facts and other facts is in the spirit of the distinction between *exogenous* and *endogenous* tuples in "causality" for query answers [24].

[2] F. Bancilhon and N. Spyratos. Update semantics of relational views. *ACM Trans. Database Syst.*, 6(4):557–575, 1981.
[3] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983.
[4] P. Buneman, S. Khanna, and W.-C. Tan. On propagation of deletions and annotations through views. In *PODS*, pages 150–158. ACM, 2002.
[5] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *IPCO*, volume 4513 of *Lecture Notes in Computer Science*, pages 182–196. Springer, 2007.
[6] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SIAM Journal on Computing*, special issue on ACM STOC 2008, to appear.
[7] M. A. Casanova, R. Fagin, and C. H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. In *PODS*, pages 171–176. ACM, 1982.
[8] G. Cong, W. Fan, and F. Geerts. Annotation propagation revisited for key preserving views. In *CIKM*, pages 632–641. ACM, 2006.
[9] S. S. Cosmadakis and C. H. Papadimitriou. Updates of relational views. *J. ACM*, 31(4):742–760, 1984.
[10] Y. Cui and J. Widom. Run-time translation of view tuple deletions using data lineage. Technical report, Stanford University, 2001. `http://dbpubs.stanford.edu:8090/pub/2001-24`.
[11] U. Dayal and P. A. Bernstein. On the correct translation of update operations on relational views. *ACM Trans. Database Syst.*, 7(3):381–416, 1982.
[12] R. Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM*, 30(3):514–550, 1983.
[13] R. Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1):83–99, 1999.
[14] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.
[15] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In *PODS*, pages 352–365. ACM, 1983.
[16] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.
[17] V. Guruswami. Inapproximability results for set splitting and satisfiability problems with no mixed clauses. *Algorithmica*, 38(3):451–469, 2003.
[18] V. Guruswami and S. Khot. Hardness of max 3SAT with no mixed clauses. In *IEEE Conference on Computational Complexity*, pages 154–162. IEEE Computer Society, 2005.
[19] H. J. Karloff and U. Zwick. A 7/8-approximation algorithm for max 3sat? In *FOCS*, pages 406–415, 1997.
[20] A. M. Keller. Algorithms for translating view updates to database updates for views involving selections, projections, and joins. In *PODS*, pages 154–163. ACM, 1985.
[21] B. Kimelfeld, J. Vondrák, and R. Williams. Maximizing conjunctive views in deletion propagation (extended version). Accessible at the first author's home page, 2011.
[22] J. Lechtenbörger and G. Vossen. On the computation of relational view complements. *ACM Trans. Database Syst.*, 28(2):175–208, 2003.
[23] B. Liu, L. Chiticariu, V. Chu, H. V. Jagadish, and F. Reiss. Automatic rule refinement for information extraction. *PVLDB*, 3(1):588–597, 2010.
[24] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *PVLDB*, 4(1):34–45, 2011.
[25] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pages 474–479, 1999.
[26] M. Y. Vardi. The complexity of relational query languages (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 137–146. ACM, 1982.