Maximizing Conjunctive Views in Deletion Propagation*

Benny Kimelfeld, IBM Research – Almaden
Jan Vondrák, IBM Research – Almaden
Ryan Williams, Stanford University

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 article studies the complexity of deletion propagation, 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 straightforward algorithm, which is called here the unidimensional algorithm. The article identifies additional cases of CQs where the unidimensional algorithm succeeds, and in contrast, shows that for some other CQs the problem is NP-hard to approximate better than some constant ratio. In fact, it is shown here that among the CQs without self joins, the hard CQs are exactly the ones that the unidimensional algorithm fails on. In other words, the following dichotomy result is proved: for every CQ without self joins, deletion propagation is either APX-hard or solvable (in polynomial time) by the unidimensional algorithm.

The article 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 sunflower CQs (i.e., CQs having a sunflower hypergraph) without self joins. The first algorithm, providing the approximation ratio $1 - 1/e$, is obtained by formulating the problem at hand as that of maximizing a monotone submodular function subject to a matroid constraint, and then using a known algorithm for such maximization. The second algorithm gives a smaller approximation ratio, $1/2$, yet in polynomial time even under combined complexity. Finally, it is shown that self joins can significantly harden approximation in deletion propagation.

Categories and Subject Descriptors: H.2 [**Database Management**]: Systems, Database Administration; F.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Deletion propagation, dichotomy, approximation

1. INTRODUCTION

A classic question in database management is the *view update problem* [Bancilhon and Spyratos 1981; Cosmadakis and Papadimitriou 1984; Dayal and Bernstein 1982; Keller 1985; Cong et al. 2006; Fagin et al. 1983]: 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

*An abridged version of this article has been published in Proceedings of the 30th ACM Symposium on Principles of Database Systems [Kimelfeld et al. 2011].

Author's addresses: B. Kimelfeld, IBM Research – Almaden, San Jose, CA 95120, USA, email: kimelfeld@us.ibm.com; J. Vondrák, IBM Research – Almaden, San Jose, CA 95120, USA, email: jvondrak@us.ibm.com; R. Williams, Computer Science Department, Stanford University, Stanford, CA 94305, USA, email: rrw@cs.stanford.edu.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

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* [Buneman et al. 2002], where the "side effect" is the set of deleted view tuples that are different from the undesired one. A solution that is side-effect free does not necessarily exist, and hence, the task is relaxed to that of *minimizing* the side effect [Buneman et al. 2002; Cong et al. 2006]. That is, the goal is to delete facts from the base relations so that the undesired tuple disappears from the view, and the maximum possible number of other tuples remain in the view.

Recent applications provide a renewed motivation for deletion propagation. Specifically, view update naturally arises when debugging *Information Extraction* (IE) programs, which can be highly complicated [Liu et al. 2010]. As a concrete example, the MIDAS system [Balakrishnan et al. 2010] 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*. 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 facts to delete from the base relations so as to eliminate the erroneous conclusion with a minimum effect on the other conclusions. Moreover, eliminating facts from the base relations may itself entail deletion propagation, as these facts are typically extracted by consulting external (possibly unclean) data sources [Riloff and Jones 1999; Liu et al. 2010].

The problem of deletion propagation is nicely illustrated by the following simple example by Cui and Widom [2001] (also referenced and used by Buneman et al. [2002]). Let $\mathrm{GroupUser}(\mathrm{group}, \mathrm{user})$ and $\mathrm{GroupFile}(\mathrm{group}, \mathrm{file})$ be two relations representing memberships 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 $\mathrm{GroupUser}(g, u)$ and $\mathrm{GroupFile}(g, f)$. Now suppose that we want to restrict a specific user from accessing a specific file, by eliminating some group-user 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):

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

A formal definition of a CQ is given in Section 2. In some applications of deletion propagation (like IE), the involved queries can be much more complicated than CQs. Nevertheless, we believe that understanding deletion propagation within the basic class of CQs is an essential step towards algorithmic solutions for practical applications. Hence, the focus of this article 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 $\mathrm{a} \in Q(I)$. A *solution* is a subinstance J of I (i.e., J is obtained from I by deleting facts) such that $\mathrm{a} \notin Q(J)$, and an *optimal solution* maximizes the number of tuples in $Q(J)$. The *side effect* is $(Q(I) \setminus \{\mathrm{a}\}) \setminus Q(J)$. Buneman et al. [2002] 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 *unidimensional* algorithm. But those classes are highly restricted. Buneman et al. [2002] show that even for a CQ as simple as the above $\mathrm{Access}(y_1, y_2)$, the unidimensional algorithm is sub-

optimal, and in fact, finding an *optimal solution* minimizing the side effect is NP-hard for $\mathrm{Access}(y_1, y_2)$. They show that hardness by proving that testing for the existence of a side-effect-free solution is NP-complete. 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. [2002], 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, as far as finding an optimal solution is concerned, that 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 $0 < \alpha_Q < 1$ (lower bounded by the reciprocal of the arity of Q; see Section 2), such that the solution J produced by the unidimensional 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 unidimensional 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 4.7): for those CQs that do not have the head domination property, not only is the unidimensional algorithm sub-optimal for MAXDP$\langle Q\rangle$, this problem is hard to solve optimally, and even hard to approximate better than some constant ratio. Hence, we show the following dichotomy in the complexity of deletion propagation. For every CQ Q without self joins, one of the following holds:

— The unidimensional 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 α_Q-approximate.

The proof of this dichotomy is nontrivial. Regarding the constant α_Q, we show that it is necessarily dependent on Q, since there is no global constant $\alpha \in (0, 1)$ that works for all CQs Q (without self joins); however, we show that such a global constant exists for a large class of CQs without self joins (that contains the *acyclic* CQs [Fagin 1983; Beeri et al. 1983]). A similar dichotomy holds for the problem of testing for the existence of a solution that is side-effect free: if the CQ has head domination, then the problem is solvable in polynomial time (using the unidimensional algorithm); otherwise, the problem is NP-complete.

So far, the only algorithm we have considered for deletion propagation is the unidimensional one. Recall that for every CQ Q without self joins, the unidimensional algorithm is a constant-factor approximation (with a constant depending on Q). In Section 6, we devise approximation algorithms with much better ratios, for the important class of CQs having a *sunflower hypergraph* and no self joins[1] (which generalize the queries that are handled by the *Fagin algorithm* [Fagin 1999] and the *threshold algorithm* [Fagin et al. 2003]).

[1] See Section 6 for the exact definition of a sunflower CQ.

The first approximation algorithm is obtained by reducing our optimization problem to that of maximizing a monotone submodular function subject to a matroid constraint [Calinescu et al. 2011; Nemhauser et al. 1978]. This reduction is not straightforward, but once done we can immediately apply the positive result of Calinescu et al. [2011] to obtain a randomized polynomial-time algorithm with an approximation ratio of $1 - 1/e$ (roughly, 0.632). We then show how this approach, combined with the unidimensional algorithm, extends to a significant generalization of sunflower CQs where, roughly, the sunflower restriction is limited just to the existential variables.

All the algorithms we mentioned thus far (including the unidimensional algorithm) terminate in time that is polynomial in the size of the database instance, but exponential in that of the query. In conventional formalism [Vardi 1982], the algorithms take polynomial time under *data complexity* (where the schema and query are fixed), yet exponential time under *combined complexity* (or *query-and-data complexity*, meaning that the schema and query are part of the input). The reason for the exponential combined complexity is the need in each of these algorithms to evaluate the CQ Q over intermediate solutions, resulting in sets of answers that can be exponentially larger than Q. The next approximation we devise, the *oblivious algorithm*, avoids any evaluation of Q over intermediate solutions. This algorithm is extremely simple, but some effort is needed for proving that it guarantees a $1/2$ approximation ratio for the class of sunflower CQs without self joins. Although this ratio is smaller than that of the submodular-optimization approach, the oblivious algorithm is our only algorithm with polynomial-time combined complexity. In particular, the oblivious algorithm is significantly faster than the rest.

Finally, in Section 7 we show that self joins in CQs introduce further hardness. Recall that for every CQ Q without self joins, $\textsc{MaxDP}\langle Q\rangle$ can be efficiently α_Q-approximated by the unidimensional algorithm, where α_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 unidimensional algorithm fails to give the desired approximation, and furthermore, we show an infinite set of CQs Q such that the achievable approximation ratio for $\textsc{MaxDP}\langle Q\rangle$ is exponentially small as a function 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 unidimensional algorithm is sub-optimal; furthermore, $\textsc{MaxDP}\langle Q\rangle$ is hard to approximate better than some constant ratio.

The work described in this article consider 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 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* [Buneman et al. 2002; Cong et al. 2006], namely, finding a solution with a minimal number of missing facts. Interestingly, this problem is closely related to that of computing the *responsibility* of a fact to an answer of a CQ [Meliou et al. 2011]. Cong et al. [2006] also studied the complexity of deletion propagation (with the goal of finding an optimal solution) in the presence of *key constraints*. Update operations other than deletion (e.g., insertion and replacement) have also been investigated [Bancilhon and Spyratos 1981], and especially in the presence of functional dependencies [Keller 1985; Cosmadakis and Papadimitriou 1984; Dayal and Bernstein 1982]. The work of Cosmadakis and Papadimitriou [1984] 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 [2003]) that, intuitively, contains the information ignored by the view.

This article is an extended version of a previous publication [Kimelfeld et al. 2011]. The technical differences are mainly the following. We give here for the first time the proof of the main result, namely Theorem 4.7. The original paper [Kimelfeld et al. 2011] has only a short, superficial sketch of this proof. In particular, Section 5 of this article gives the full details of the main part of the proof (hardness). Additional proofs given here for the first time are those of Proposition 3.4 and Theorem 7.1. Finally, the formulation of MAXDP$\langle Q\rangle$ as a problem of maximizing a monotone submodular function subject to a matroid constraint (for the case of sunflower CQs without self joins), given in Section 6.2, is new. In the abridged version, we presented a restriction of the algorithm of Calinescu et al. [2011] to our specific case, rather than making the connection to submodular optimization. Further discussion on that algorithm is in Section 6.2.

2. PRELIMINARIES

2.1. Schemas, Instances, and CQs

We fix an infinite set 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 Const (i.e., R_i^I is a finite subset of 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. For example, we may write $R(\mathbf{c}) \in I$ to say that $R(\mathbf{c})$ is a fact of I. As another example, $J \subseteq I$ means that $R_i^J \subseteq R_i^I$ for all $i = 1, \ldots, m$; in that case, we say that J is a *sub-instance* of I.

We fix an infinite set Var of *variables*, and assume that Const and 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 Var), $\mathbf{c}$ is a tuple of constants (from Const), and $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$ is a conjunction of atomic formulas over $\mathbf{R}$; an atomic formula is also called an *atom*. We may write just $Q(\mathbf{y})$, or even just Q, if $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$ is irrelevant. We denote by $\mathrm{atoms}(Q)$ the set of atoms of Q. We usually write $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{c})$ by simply listing $\mathrm{atoms}(Q)$. 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 $\mathrm{schema}(Q)$. 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 in $\mathrm{schema}(Q)$ appears exactly 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 ϕ 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 ϕ. 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 $\mathrm{arity}(Q)$, 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 $\mathrm{Access}(y_1, y_2)$ defined in (1) (and discussed in the introduction), up to renaming of relation symbols.

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

The atoms of $Q_2^\star$ are $\phi_1 = R_1(x, y_1)$ and $\phi_2 = R_2(x, y_2)$. In the examples of this article, R_i and R_j are assumed to be different symbols when $i \neq j$. In particular, $\mathrm{schema}(Q_2^\star)$ consists of two distinct binary relation symbols: R_1 and R_2. Hence, $Q_2^\star$ 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:

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

Note that $\mathrm{schema}(Q_k^\star)$ has k (distinct) binary relation symbols: $R_1, \ldots, R_k$. □

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 $\mathrm{schema}(Q)$. An *assignment* for Q is a mapping $\mu : \mathsf{Var}(Q) \to \mathsf{Const}$. For an assignment μ 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 $\phi \in \mathrm{atoms}(Q)$, the fact $\mu(\phi)$ is the one obtained from ϕ by replacing every variable z with the constant $\mu(z)$. A *match for* Q *in* I is an assignment μ for Q, such that $\mu(\phi)$ is a fact of I for all $\phi \in \mathrm{atoms}(Q)$. If μ is a match for Q in I, then $\mu(\mathbf{y})$ is called an *answer* (*for* Q *in* I), and we also say that μ *produces* the tuple $\mu(\mathbf{y})$. The *result* of evaluating Q over I, denoted $Q(I)$, is the set of all the answers for Q in I.

Let Q be a CQ, and let I be an instance over $\mathrm{schema}(Q)$. A fact $f \in I$ is said to be Q*-useful* (*in* I) if there is an atom $\phi \in \mathrm{atoms}(Q)$ and a match μ for Q in I, such that $f = \mu(\phi)$.

Example 2.2. Figure 1 depicts an instance I_3 over $\mathrm{schema}(Q_3^\star)$. Note that every fact of I_3 is $Q_3^\star$-useful. However, if we added to I_3 the fact $f = R_2(4, \diamond)$ then f would not be $Q_3^\star$-useful, since no match for $Q_3^\star$ in I_3 would map $R_2(x, y_2)$ to f. □

For a CQ Q, we denote by $\mathcal{H}(Q)$ the hypergraph that is associated with Q; that is, $\mathcal{H}(Q)$ is the hypergraph (V, E) where $V = \mathsf{Var}(Q)$ and $E = \{\mathsf{Var}(\phi) \mid \phi \in \mathrm{atoms}(Q)\}$.

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 $\textsc{MaxDP}\langle Q\rangle$ and is defined as follows. The input consists of an instance I over $\mathrm{schema}(Q)$, 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. Clearly, every solution J satisfies $|Q(J)| \leq |Q(I)| - 1$ (since $\mathbf{a} \in Q(I)$ is assumed and $\mathbf{a} \notin Q(J)$ is required). A solution J that satisfies $|Q(J)| = |Q(I)| - 1$ (that is, J is an optimal solution with $Q(J) = Q(I) \setminus \{\mathbf{a}\}$) is said to be *side-effect free*. Note that a side-effect-free solution does not necessarily exist.

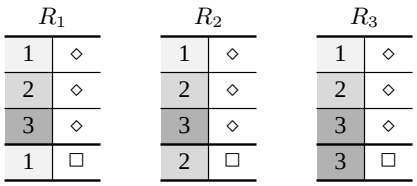

Fig. 1. Instance I_3 over $\mathrm{schema}(Q_3^\star)$

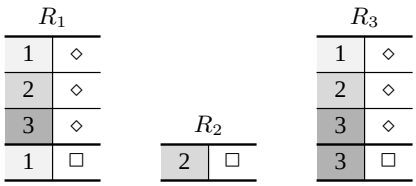

Fig. 2. The instance $I_3 \setminus I_3[\phi_2, \mathbf{a}]$

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

2.3. Complexity

In this article we use *data complexity* [Vardi 1982] for measuring the complexity of deletion propagation and related algorithms. This means that the CQ Q is held fixed, and the input consists of the instance I and the tuple $\mathbf{a}$. An exception is in Section 6.3, where we discuss the *combined complexity* (a.k.a. *query-and-data complexity*), meaning that the query Q is taken as part of the input.

3. THE UNIDIMENSIONAL ALGORITHM

In this section, we recall an extremely simple algorithm for deletion propagation by Buneman et al. [2002]. They gave this algorithm for proving the tractability of MAXDP$\langle Q\rangle$ in the case where there is no projection (i.e., Q has no existential variables). Here, we extend that algorithm to general CQs in a straightforward way. But first, we need some notation.

Consider a CQ $Q(y_1, \ldots, y_k)$. Let ϕ be an atom of Q, let I be an instance over $\mathrm{schema}(Q)$, let f be a fact in I, and let $\mathbf{a}$ be a tuple in Const^k. We say that f is *ϕ-consistent with* $\mathbf{a}$ if $f = \mu(\phi)$ for some assignment μ for Q, such that $\mu(\mathbf{y}) = \mathbf{a}$ (note that μ is not necessarily a match for Q in I). We denote by $I[\phi, \mathbf{a}]$ the set of all the facts that are ϕ-consistent with $\mathbf{a}$. Hence, $I \setminus I[\phi, \mathbf{a}]$ is the sub-instance of I that is obtained from I by deleting every fact that is ϕ-consistent with $\mathbf{a}$.

Example 3.1. Consider again the CQ $Q_3^\star$ (defined in Example 2.1), and the instance I_3 of Figure 1. Let ϕ_2 be the second atom of $Q_3^\star$, namely $R_2(x, y_2)$, and let f be the fact $R_2(3, \diamond)$. Then f is ϕ_2-consistent with $(\square, \diamond, \diamond)$, but it is not ϕ_2-consistent with either $(\diamond, \square, \diamond)$ or $(\square, \square, \square)$. Now, let $\mathbf{a}$ be the tuple $(\diamond, \diamond, \diamond)$. All three top facts of R_2 are ϕ_2-consistent with $\mathbf{a}$; hence, $I[\phi_2, \mathbf{a}]$ is the set of these three facts, and the sub-instance $I \setminus I[\phi_2, \mathbf{a}]$ is the one that is depicted in Figure 2. □

Going back to deletion propagation, the following is an easy proposition.

PROPOSITION 3.2. *Let Q be a CQ and let ϕ be an atom of Q. Given the input I and $\mathbf{a}$ for* MAXDP$\langle Q\rangle$*, the instance $I \setminus I[\phi, \mathbf{a}]$ is a solution.*

Proposition 3.2 holds, since every match μ for Q in I, where μ produces $\mathbf{a}$, maps the atom ϕ to a fact that is ϕ-consistent with $\mathbf{a}$; therefore, no matches for Q in $I \setminus I[\phi, \mathbf{a}]$ produces $\mathbf{a}$. The proposition gives us a solution J_ϕ (i.e., $I \setminus I[\phi, \mathbf{a}]$) for each atom ϕ of Q, and the unidimensional algorithm simply selects the best J_ϕ. In other words, the unidimensional algorithm returns $I \setminus I[\phi_{\max}, \mathbf{a}]$, where $\phi_{\max}$ is such that $Q(I \setminus I[\phi_{\max}, \mathbf{a}])$

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

1: **for all** $\phi \in \text{atoms}(Q)$ **do**
2: $\quad \mathcal{A}_\phi \leftarrow Q(I \setminus I[\phi, \mathbf{a}])$
3: $\phi_{\max} \leftarrow \text{argmax}_\phi\{|\mathcal{A}_\phi|\}$
4: **return** $I \setminus I[\phi_{\max}, \mathbf{a}]$

Fig. 3. The unidimensional algorithm

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

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

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

Fig. 4. Solution J

has the maximal cardinality among all the $Q(I \setminus I[\phi, \mathbf{a}])$. We denote this algorithm by Unidim$\langle Q \rangle$, and its pseudo-code is in Figure 3.

Example 3.3. For the CQ $Q_3^\star$, let us show how the unidimensional algorithm operates on I_3 of Figure 1 and the tuple $\mathbf{a} = (\diamond, \diamond, \diamond)$. For $i \in \{1, 2, 3\}$, let ϕ_i be the atom $R_i(x, y_i)$. As shown in Example 3.1 for $i = 2$, the set $I_3[\phi_i, \mathbf{a}]$ consists of all the facts of R_i except for $R_i(i, \square)$; hence, in $I_3 \setminus I_3[\phi_i, \mathbf{a}]$, only $(i, \square)$ remains in R_i. For $i = 1, 2, 3$, let $J_i = I_3 \setminus I_3[\phi_i, \mathbf{a}]$. So, we have $Q_3^\star(J_1) = \{(\square, \diamond, \diamond)\}$, $Q_3^\star(J_2) = \{(\diamond, \square, \diamond)\}$, and $Q_3^\star(J_3) = \{(\diamond, \diamond, \square)\}$. It follows that $|Q_3^\star(J_i)| = 1$ for all $i \in \{1, 2, 3\}$, and therefore, the unidimensional algorithm can return any J_i (depending on the traversal order over the atoms). None of J_1, J_2 and J_3 are optimal solutions; to see that, consider the instance J of Figure 4. The reader can easily verify that J is a solution (i.e., $J \subseteq I$ and $\mathbf{a} \notin Q_3^\star(J)$), and moreover, $Q_3^\star(J)$ contains the answers that survived in *all* three J_i: $(\square, \diamond, \diamond)$, $(\diamond, \square, \diamond)$, and $(\diamond, \diamond, \square)$. In particular, $|Q_3^\star(J)| = 3$. Observe that J is an optimal solution, and even side-effect free; indeed, the reader can easily verify that $Q_3^\star(I_3) = Q_3^\star(J) \cup \{\mathbf{a}\}$. $\square$

3.1. Complexity

Recall that our default notion of complexity is that of data complexity. In particular, the unidimensional algorithm terminates in polynomial time, although it involves the evaluation of the answer sets $Q(I \setminus I[\phi, \mathbf{a}])$ that can be exponentially larger than Q (hence, each such evaluation can take exponential time in the size of Q).

3.2. Approximation

Next, we show that in the case where Q has no self joins,[2] the unidimensional algorithm approximates MaxDP$\langle Q \rangle$ within a constant ratio that depends only on Q.

PROPOSITION 3.4. *If Q is a CQ without self joins, then* Unidim$\langle Q \rangle$ *is a* $1/k$*-approximation for* MaxDP$\langle Q \rangle$*, where* $k = \min\{\text{arity}(Q), |\text{atoms}(Q)|\}$.

PROOF. Let $\{\phi_1, \ldots, \phi_m\}$ be a minimal subset of $\text{atoms}(Q)$ such that $\text{Var}_\mathsf{h}(Q) = \cup_{i=1}^m \text{Var}_\mathsf{h}(\phi_i)$ (that is, $\phi_1, \ldots, \phi_m$ cover all the head variables). Note that $m \leq k$. We will show that Unidim$\langle Q \rangle$ is a $1/m$-approximation. Let I and $\mathbf{a}$ be input for MaxDP$\langle Q \rangle$, and let J_{opt} be an optimal solution. For $i = 1, \ldots, m$, let $J_i = I \setminus I[\phi_i, \mathbf{a}]$. Let $\mathbf{b}$ be a tuple in $Q(J_{\text{opt}})$, and let μ be a match for Q in J_{opt} such that μ produces $\mathbf{b}$. From $\mathbf{b} \neq \mathbf{a}$ (as J is a solution) and the assumption that Q has no self joins, it follows there is some $i \in \{1, \ldots, m\}$ such that $\mu(\phi_i)$ is not ϕ_i-consistent with $\mathbf{a}$. Therefore, the tuples of J_{opt} that are ϕ_i-consistent with $\mathbf{a}$ are not required for producing $\mathbf{b}$, and we have $\mathbf{b} \in Q(J_i)$. We conclude that $Q(J_{\text{opt}}) \subseteq \cup_{i=1}^m Q(J_i)$, and hence, at least one J_j satisfies

[2] In Section 7 we discuss the implications of self joins.

$|Q(J_j)| \geq |Q(J_{\text{opt}})|/m$. But the solution returned by $\mathsf{Unidim}\langle Q\rangle$ is at least as good as each J_i, and hence, a $1/m$-optimal solution. □

Next, we will show that $1/\text{arity}(Q)$ and $1/|\text{atoms}(Q)|$ are tight lower bounds on the approximation ratio of the unidimensional algorithm, for some queries without self joins. More precisely, we will show that for all natural numbers k, the CQ $Q_k^\star$ (defined in Example 2.1), which satisfies $\text{arity}(Q_k^\star) = |\text{atoms}(Q_k^\star)| = k$, is such that the unidimensional algorithm returns no better than a $1/k$-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 $1/k$ ratio. An example of our construction for $i = 3$ is I_3 of Figure 1. For general k, 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 $(i, \square)$. The tuple a is $\diamond^k$ (i.e., the tuple of k diamonds).

To see that I_k and 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. See, for example, the instance J of Figure 4. 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 of only $\diamond$s, except for the ith element, which is $\square$. Note that J is optimal, since J is side-effect free (i.e., $Q_k^\star(I_k) = Q_k^\star(J) \cup \{\mathbf{a}\}$). Nevertheless, the unidimensional 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 3.3. By doing so, the unidimensional algorithm produces a sub-instance J', such that $Q_k^\star(J')$ contains exactly one tuple (namely, the one with $\square$ as the ith element), thus no better than a $1/k$-approximation.

3.3. General Comments

We described the unidimensional algorithm and discussed its correctness, complexity, and approximation guarantee. In the next section, we discuss CQs without self joins, and characterize the ones for which the unidimensional algorithm is optimal. Furthermore, we will show that the unidimensional algorithm is optimal for $\text{MAXDP}\langle Q\rangle$ *precisely* when $\text{MAXDP}\langle Q\rangle$ is tractable; for each of the remaining Q, the problem is 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 unidimensional algorithm (optimally) solves $\text{MAXDP}\langle Q\rangle$. We then state that when there are no self joins, this property *exactly* captures the tractability of $\text{MAXDP}\langle Q\rangle$: in the absence of head domination $\text{MAXDP}\langle Q\rangle$ is intractable, and moreover, it cannot be approximated better than some constant ratio (it is APX-hard). We give the proof of this hardness result in the following section.

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 $\text{atoms}(Q)$ as the set of nodes, and that has an edge $\{\phi_1, \phi_2\}$ whenever ϕ_1 and ϕ_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 ϕ be an atom of Q, and let P be a set of atoms of Q. We say that P is *head-dominated* by ϕ if ϕ 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 $\phi \in \text{atoms}(Q)$, such that P is head-dominated by ϕ.

As an example, for the CQ $Q_k^\star$ defined in (3), the graph $\mathcal{G}_\exists(Q_k^\star)$ is a clique over $\text{atoms}(Q_k^\star)$, and so it has only one connected component. For $k > 1$, none of the atoms of

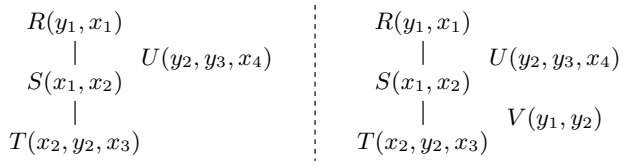

Fig. 5. The graphs $\mathcal{G}_\exists(Q)$ and $\mathcal{G}_\exists(Q')$ for the CQs Q and Q' of Example 4.2

$Q_k^\star$ 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 5 shows the graph $\mathcal{G}_\exists(Q)$, which has two connected components: $\{R(y_1, x_1), S(x_1, x_2), T(x_2, y_2, x_3)\}$ and $\{U(y_2, y_3, x_4)\}$. 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 5. 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. □

The CQ Q' in Example 4.2 illustrates the fact that in the definition of head domination (Definition 4.1), the atom ϕ is not required to be in the component P itself—it can be any atom of the CQ.

4.2. Optimality of the Unidimensional Algorithm

We now prove that if a CQ Q without self joins has the head-domination property, then the unidimensional algorithm is optimal for MAXDP$\langle Q\rangle$. For that, we need the following proposition.

PROPOSITION 4.3. *Let Q be a CQ, and let ϕ be an atom of Q. Let I and $\mathbf{a}$ be input for* MAXDP$\langle Q\rangle$*, and let J be any solution. Suppose that none of the tuples in $J[\phi, \mathbf{a}]$ are Q-useful in J. Then $Q(J) \subseteq Q(I \setminus I[\phi, \mathbf{a}])$, and hence, the solution J' returned by the unidimensional algorithm is such that $|Q(J')| \geq |Q(J)|$.*

PROOF. Since no fact in $J[\phi, \mathbf{a}]$ is Q-useful in J, removing these facts from J does not affect the set of answers: $Q(J) = Q(J \setminus J[\phi, \mathbf{a}])$. Also, $J \subseteq I$, and removing facts outside of J does not change anything, so we obtain

$$Q(J) = Q(J \setminus I[\phi, \mathbf{a}]) \subseteq Q(I \setminus I[\phi, \mathbf{a}])$$

using the monotonicity of Q (as a CQ). One of the solutions considered by the unidimensional algorithm is $I \setminus I[\phi, \mathbf{a}]$, so if J' is eventually returned then $|Q(J')| \geq |Q(J)|$. □

We now prove the theorem.

THEOREM 4.4. *Let Q be a CQ without self joins. If Q has the head-domination property, then* Unidim$\langle Q\rangle$ *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 a for MAXDP$\langle Q\rangle$. Let J be any solution (e.g., an optimal solution). Let $P_1,\ldots,P_m$ be the connected components of $\mathcal{G}_\exists(Q)$. For each P_i, let ϕ_i be an (arbitrary) atom of Q, such that P_i is head-dominated by ϕ_i. We will show that there is some $i\in\{1,\ldots,m\}$, such that none of the tuples in $J[\phi_i,\mathbf{a}]$ are Q-useful in J. This is enough for completing the proof, due to Proposition 4.3.

So, by way of contradiction, suppose otherwise. Then for all $i\in\{1,\ldots,m\}$ we select a fact $f_i\in J[\phi_i,\mathbf{a}]$ and a match μ_i for Q in J, such that μ_i maps some atom ϕ_i' of Q to f_i. Since ϕ_i and ϕ_i' have the same relation symbol (i.e., that of f), and Q has no self joins, we get that ϕ_i' is necessarily ϕ_i. So, for all $i\in\{1,\ldots,m\}$ we have $\mu_i(\phi_i)=f_i$.

Suppose that the tuple $\mathbf{y}$ of head variables of Q is $y_1,\ldots,y_k$, and that the input tuple $\mathbf{a}$ is $a_1,\ldots,a_k$. Recall that each ϕ_i head-dominates P_i, and that each $f_i=\mu_i(\phi_i)$ is ϕ_i-consistent with $\mathbf{a}$. Therefore, every head variable y_j (where $1\leq j\leq k$) that appears in P_i (hence, in ϕ_i) is mapped by μ_i to a_j; that is, $\mu_i(y_j)=a_j$ whenever y_j occurs in P_i.

We conclude that if P_q and P_l are components that share head variables, then μ_q and μ_l agree on those shared head variables. Also, recall that P_q and P_l do not share existential variables unless $q=l$. So now, we define an assignment μ for Q, as follows. For every variable $z\in\mathsf{Var}(Q)$, the constant $\mu(z)$ is equal to $\mu_i(z)$, where P_i is one of the components where z occurs. Observe the following. First, μ is well defined. Second, for each atom ϕ of Q, if ϕ belongs to the component P_i then $\mu(\phi)=\mu_i(\phi)$; in particular, $\mu(\phi)\in J$, since μ_i is a match for Q in J. Finally, μ maps every head variable y_j (where $1\leq j\leq k$) to a_j; hence, $\mu(\mathbf{y})=\mathbf{a}$. But then, it follows that μ is a match for Q in J with $\mu(\mathbf{y})=\mathbf{a}$, and so $\mathbf{a}\in Q(J)$, which contradicts the fact that J is a solution. □

As a simple consequence of Theorem 4.4, if Q is a CQ without self joins, and every join variable of Q is a head variable, then the unidimensional algorithm optimally solves MAXDP$\langle Q\rangle$ (since then $\mathcal{G}_\exists(Q)$ has no edges). Actually, it can easily be shown that if every head variable is a join variable, then this statement is true even without requiring the lack of self joins. However, in Section 7 we show that Theorem 4.4 is not correct for *all* CQs (including those with 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 unidimensional algorithm does not optimally solve MAXDP$\langle Q\rangle$; moreover, 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 states that if a CQ Q without self joins does not have the head-domination property, then in contrast to Theorem 4.4, MAXDP$\langle Q\rangle$ is NP-hard, and even hard to approximate better than some constant ratio. In Section 5, we prove this theorem.

THEOREM 4.5. *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 α_Q-approximated in polynomial time.*

One may wonder whether, in Theorem 4.5, it is necessary to have α_Q depending on Q. In other words, does Theorem 4.5 hold for a global α that is applicable to all CQs without self joins? In Section 5 we show that the answer is positive if we are restricted to the class of *acyclic* CQs [Fagin 1983; Beeri et al. 1983] and to the class of CQs over binary relation symbols; this will be shown in Theorem 5.17. In contrast, the following proposition shows that no such global α exists for the class of all 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*) *$\mathsf{Unidim}\langle Q_k\rangle$ is a $(1-1/k)$-approximation for $\mathrm{MAXDP}\langle Q_k\rangle$.*

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 ϕ_i is given by

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

Note that ϕ_i contains all the variables of Q_k, except for y_i. Clearly, Q_k satisfies Properties 1 and 2 (i.e., Q_k 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 $\mathrm{MAXDP}\langle Q_k\rangle$. Let i and j be such that $1 \leq i < j \leq k$, and let f_i and f_j be facts in $I[\phi_i, \mathbf{a}]$ and $I[\phi_j, \mathbf{a}]$, respectively. The only answer in $Q_k(I)$ that agrees with both f_i and f_j on the head variables is $\mathbf{a}$. Therefore, among the answers in $Q_k(I) \setminus \{\mathbf{a}\}$, those that require f_i are disjoint from those that require f_j. Hence, the best solution among the $I \setminus I[\phi_i, \mathbf{a}]$ misses at most $1/k$ of the tuples in $Q_k(I) \setminus \{\mathbf{a}\}$. In particular, the unidimensional algorithm returns a $(1-1/k)$-approximation, as claimed. □

4.4. Dichotomy

We conclude this section with the following dichotomy that is obtained by combining Theorems 4.4 and 4.5.

THEOREM 4.7 (DICHOTOMY). *For a CQ Q without self joins, one of the following holds.*

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

Moreover, (1) holds if and only if Q has the head-domination property.

A proof similar to (actually, simpler than) that of Theorem 4.7 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. [2002]. 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 4.4), and is NP-complete otherwise.

5. PROOF OF HARDNESS

In this section, we prove Theorem 4.5. We begin with Section 5.1 that gives the proof for one special CQ: $Q_2^\star$. Then, in Section 5.2 we define the notion of a *fact-wise reduction*, which is a tool that we use here and thereafter. In Section 5.3 we set some terminology and assumptions for the remainder of the proof. The main body of the proof is in Sections 5.4 and 5.5, which consider two different cases of CQs. Finally, in Section 5.6 we show how the proof is adapted to provide a global (query-independent) inapproximability factor for special classes of CQs.

5.1. Hardness of MAXDP$\langle Q_2^\star \rangle$

Our first step is to show hardness of MAXDP$\langle Q_2^\star \rangle$ (where $Q_2^\star$ is defined in (2)), which is a special case of a CQ that does not have the head-domination property. Specifically, we prove the following.

LEMMA 5.1. *There is a constant $\alpha \in (0,1)$, such that* MAXDP$\langle Q_2^\star \rangle$ *is NP-hard to α-approximate.*

In the remainder of this section, we prove Lemma 5.1. Buneman et al. [2002] showed that deciding whether there is a solution that is side-effect free is NP-complete for $Q_2^\star$. In their proof, they described a reduction from *non-mixed satisfiability*, denoted NMSat, which is the problem of deciding on the satisfiability of a non-mixed CNF formula, where *non-mixed* means that no clause contains both negated and non-negated variables. This problem is known to be NP-complete [Gold 1978]. Guruswami [2003] showed a constant-factor lower bound on the approximability of non-mixed satisfiability. However, later we will show that 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 reduction used by Guruswami [2003] and an inapproximability result by Feige [1998].

We first review the reduction of Buneman et al. [2002]. Consider an input $\varphi = (\wedge_{j=1}^{l} P_j) \wedge (\wedge_{k=1}^{m} N_k)$ for NMSat, over the variables $x_1, \ldots, x_n$, where each P_j is a clause with positive literals (*positive clause*), and each N_k is a clause with negative literals (*negative clause*). The reduction constructs an instance I_φ that contains two fresh constants $\lhd$ and $\rhd$, and the following facts:

— $R_1(x_i, P_j)$ for each positive clause P_j and disjunct x_i of P_j.
— $R_2(x_i, N_k)$ for each negative clause N_k and disjunct $\neg x_i$ of N_k.
— $R_1(x_i, \lhd)$ and $R_2(x_i, \rhd)$ for each variable x_i.

Figure 6 shows an example of the construction of I_φ. The tuple a to be deleted is $(\lhd, \rhd)$. This completes the description of the reduction. Next, we explain its correctness.

Given a solution J, a truth assignment $\tau_J : \{x_1, \ldots, x_n\} \to \{\mathbf{true}, \mathbf{false}\}$ is obtained as follows. Note that for each variable x_i, either $R_1(x_i, \lhd)$ or $R_2(x_i, \rhd)$ should be absent in J, or else $\mathbf{a} = (\lhd, \rhd) \in Q_2^\star(J)$. Without loss of generality, we can assume that J contains exactly one of $R_1(x_i, \lhd)$ or $R_2(x_i, \rhd)$ (for each x_i), and that J contains *all* the facts of I_φ of the form $R_1(x_i, P_j)$ or $R_2(x_i, N_k)$; otherwise, we can satisfy these assumptions by adding tuples to J in such a way that J is still a solution. Now, we set the assignment τ_J as follows.

— If J contains $R_1(x_i, \lhd)$, then we set $\tau_J(x_i) = \mathbf{false}$.
— if J contains $R_2(x_i, \rhd)$, then we set $\tau_J(x_i) = \mathbf{true}$.

Note that a positive clause P_j is satisfied by τ_J if and only if $Q(J)$ contains the answer $(P_j, \rhd)$; similarly, a negative clause N_k is satisfied by τ_J if and only if $Q(J)$ contains the answer $(\lhd, N_k)$. Let $\#\tau_J$ denote the number of clauses that are satisfied by τ_J. Let $M(\varphi)$ denote the number of pairs (P_j, N_k), such that P_j and N_k have one or more common variables. Then, we have the following equation.

$$|Q(J)| = \#\tau_J + M(\varphi) \tag{4}$$

Observe that $|Q(I_\varphi)| = |\varphi| + M(\varphi) + 1$, where $|\varphi|$ is the number of clauses of ϕ. Hence, if J is side-effect free, then $|Q(J)| = |\varphi| + M(\varphi)$; but then, from (4) we have that $\#\tau_J = |\varphi|$, that is, τ_J is a satisfying assignment. So, if there is a side-effect-free solution for I_φ and a then φ is satisfiable. The other direction is shown similarly: for each truth

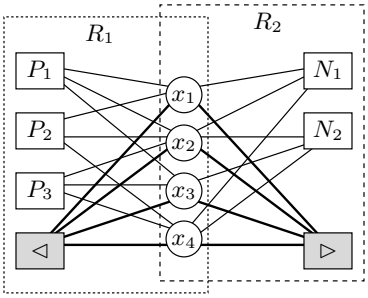

Fig. 6. The instance I_φ for the formula $\varphi = P_1 \wedge P_2 \wedge P_3 \wedge N_1 \wedge N_2$, where $P_1 = x_1 \vee x_2 \vee x_3$, $P_2 = x_1 \vee x_2 \vee x_4$, $P_3 = x_2 \vee x_3 \vee x_4$, $N_1 = \neg x_1 \vee \neg x_2 \vee \neg x_4$, and $N_2 = \neg x_2 \vee \neg x_3 \vee \neg x_4$

assignment τ, we can construct a solution J_τ, such that $|Q(J_\tau)| = \#\tau + M(\varphi)$, and then if τ is a satisfying assignment then J_τ is side-effect free. In conclusion, there is a solution that is side-effect free for I_φ and a if and only if φ is satisfiable.

Our goal is to use Equation (4) in order to obtain a constant lower bound for maximizing $|Q(J)|$. So, we consider the problem MaxNMSat, which is that of finding a truth assignment τ for a non-mixed CNF formula, where τ satisfies as many clauses as possible (i.e., $\#\tau$ is maximal over all truth assignments). Note that Equation (4) shows how to obtain an optimal assignment for φ from an optimal solution for I_φ and vice versa. Guruswami [2003] showed a constant-factor lower bound on the approximability of MaxNMSat. Now, let $0 < \alpha < 1$ be a positive number. What approximation ratio for MaxNMSat does an α-approximation J for MAXDP$\langle Q_2^\star \rangle$ provide? To compute that, let τ_{opt} be an optimal assignment for φ, and let J_{opt} be an optimal solution for I_φ. Then, from (4) we get the following.

$$\#\tau_J + M(\varphi) = |Q(J)| \geq \alpha \cdot |Q(J_{\text{opt}})| = \alpha \cdot \#\tau_{\text{opt}} + \alpha \cdot M(\varphi)$$

Therefore, we get:

$$\frac{\#\tau_J}{\#\tau_{\text{opt}}} \geq \alpha - (1-\alpha)\frac{M(\varphi)}{\#\tau_{\text{opt}}} \tag{5}$$

The problem we face is that $M(\varphi)/\#\tau_{\text{opt}}$ can be large (i.e., not upper bounded by a constant), since $M(\varphi)$ can be quadratic in $|\varphi|$ (while $\#\tau_{\text{opt}} \leq |\varphi|$ always holds). To deal with this problem, we need the following lemma, which gives a constant-factor lower bound on the approximability of Max35NMSat, which is the same as MaxNMSat, except that the (non-mixed) formula φ is such that each clause contains at most 3 variables, and each variable occurs in at most 5 clauses.

LEMMA 5.2. *There is a constant $\beta \in (0,1)$, such that* Max35NMSat *is NP-hard to β-approximate.*

PROOF. We use a simple reduction by Guruswami [2003], which takes a CNF formula γ and produces a new, non-mixed CNF formula φ. Specifically, this reduction replaces each clause of the form $(a \vee b \vee \neg c)$ with the clauses $(a \vee b \vee t)$ and $(\neg c \vee \neg t)$, where t is a fresh new variable that appears just in these two new clauses. Similarly, it replaces each clause of the form $(a \vee \neg b \vee \neg c)$ with $(\neg t \vee \neg b \vee \neg c)$ and $(a \vee t)$. Now, if γ is a 3-CNF formula where each variables appears at most 5 times, then the constructed φ has the following important properties.

(1) φ is non-mixed.
(2) Each clause has at most 3 variables.
(3) Each variable occurs in at most 5 clauses.

(4) If γ is satisfiable, then φ is satisfiable.
(5) If at most $1/2 + \beta/2$ of the clauses of γ are satisfiable, then at most β of the clauses of φ are satisfiable.

So now, we can use a result by Feige [1998] showing that for some β with $1/2 + \beta/2 < 1$ it is NP-hard to distinguish between a Properties (4) and (5) above for γ, even if γ is a 3-CNF formula where each variables appears at most 5 times. □

So, from now on we assume that each clause of φ contains at most 3 variables, and that each variable in φ occurs in at most 5 clauses. In this case, we can bound $M(\varphi)$ as follows. Recall that $M(\varphi)$ is the number of pairs (P_j, N_k), such that the clauses P_j and N_k share at least one common variable. Suppose, w.l.o.g., that at most half of the clauses of φ are negative (as in Figure 6). Now, each N_k contains at most 3 variables, and each of these variables belongs to at most 4 clauses P_j; hence, N_k contributes at most 12 units to $M(\varphi)$. It thus follows that

$$M(\varphi) \leq \frac{|\varphi|}{2} \cdot 12 = 6|\varphi| \,.$$

Another needed observation is that there is always an assignment that satisfies at least half of the clauses of φ. Hence, we have that

$$\#\tau_{\text{opt}} \geq \frac{|\varphi|}{2} \,.$$

From (5) and the above two inequalities we conclude that

$$\frac{\#\tau_J}{\#\tau_{\text{opt}}} \geq \alpha - 12(1-\alpha) \,.$$

Lemma 5.2 states that unless P $\neq$ NP, there is a constant $0 < \beta < 1$, such that no polynomial-time algorithm β-approximates Max35NMSat. But given a $\beta \in (0,1)$, one can easily find $\alpha_\beta \in (0,1)$ such that $\alpha_\beta - 12(1-\alpha_\beta) \geq \beta$; hence, it is intractable to get an α_β-approximation for MAXDP$\langle Q_2^\star \rangle$. This completes the proof of Lemma 5.1.

5.2. Fact-Wise Reductions

We fix two CQs Q and Q' is this section. A *fact-wise reduction* from MAXDP$\langle Q \rangle$ to MAXDP$\langle Q' \rangle$ is a special type of reduction that we use in this article. Using such a reduction, we can obtain an α-optimal solution for MAXDP$\langle Q \rangle$ from an α-optimal solution to MAXDP$\langle Q' \rangle$. In this section, we give the definition of a fact-wise reduction. Formally, a fact-wise reduction from MAXDP$\langle Q \rangle$ to MAXDP$\langle Q' \rangle$ is a pair (Φ, Ψ), where Φ and Ψ are functions that are defined as follows.

The function Φ maps every fact f over $\text{schema}(Q)$ to a finite set $\Phi(f)$ of facts over $\text{schema}(Q')$. For an instance I over $\text{schema}(Q)$ we define:

$$\Phi(I) \stackrel{\text{def}}{=} \bigcup_{f \in I} \Phi(f)$$

That is, $\Phi(I)$ is the instance over $\text{schema}(Q')$ that is the union of all the $\Phi(f)$ over all the facts of I. Note that for two distinct facts f_1 and f_2 of I, the finite sets $\Phi(f_1)$ and $\Phi(f_2)$ are not necessarily disjoint. Also note that for a sub-instance J of I it holds that $\Phi(J) \subseteq \Phi(I)$.

For an instance I over $\text{schema}(Q)$ and a sub-instance J' of $\Phi(I)$, we define:

$$\Phi_I^{-1}(J') = \{f \in I \mid \Phi(f) \subseteq J'\}$$

That is, $\Phi_I^{-1}(J')$ is the sub-instance of I that consists of all the facts $f \in I$, such that J' contains every fact of $\Phi(f)$. Note that for a sub-instance J of I it holds that $J \subseteq \Phi_I^{-1}(\Phi(J))$, and hence, $Q(J) \subseteq Q(\Phi_I^{-1}(\Phi(J)))$.

Let $\mathrm{arity}(Q) = k$ and $\mathrm{arity}(Q') = k'$. The function Ψ is a mapping from Const^k to $\mathsf{Const}^{k'}$. As the special property of the fact-wise reduction, we require Ψ to be a bijection (i.e., a one-to-one and onto function) from $Q(\Phi_I^{-1}(J'))$ to $Q'(J')$ for all instances I over $\mathrm{schema}(Q)$ and sub-instances J' of $\Phi(I)$.

Note that a fact-wise reduction (Φ, Ψ) does not necessarily exist; however, when it exists it gives us a reduction from $\textsc{MaxDP}\langle Q\rangle$ to $\textsc{MaxDP}\langle Q'\rangle$, according to the following lemma.

LEMMA 5.3. *Let Q and Q' be two CQs, let (Φ, Ψ) be a fact-wise reduction from $\textsc{MaxDP}\langle Q\rangle$ to $\textsc{MaxDP}\langle Q'\rangle$, and let I and $\mathbf{a}$ be input for $\textsc{MaxDP}\langle Q\rangle$. Consider the problem $\textsc{MaxDP}\langle Q'\rangle$, and let J' be an α-optimal solution for $\Phi(I)$ and $\Psi(\mathbf{a})$. Then $\Phi_I^{-1}(J')$ is an α-optimal solution for I and $\mathbf{a}$.*

PROOF. Let $J = \Phi_I^{-1}(J')$. We first show that J is a solution. For that, we need to show that $\mathbf{a} \notin Q(J)$. Recall that Ψ maps $Q(J)$ to $Q'(J')$, and hence, if $\mathbf{a} \in Q(J)$ then $\Psi(\mathbf{a}) \in Q'(J')$, in contrast to the fact that J' is a solution for $\Phi(I)$ and $\Psi(\mathbf{a})$.

It is left to prove that J is α-optimal. Let J_{opt} be an optimal solution for I and $\mathbf{a}$, and let J'_{opt} be an optimal solution for $\Phi(I)$ and $\Psi(\mathbf{a})$. Since $\Phi_I^{-1}(\Phi(I)) = I$, we get that Ψ is a bijection from $Q(I)$ to $Q'(\Phi(I))$; similarly, using J_{opt} instead of I we get that Ψ is also a bijection from $Q(J_{\mathrm{opt}})$ to $Q'(\Phi(J_{\mathrm{opt}}))$. Therefore, $\mathbf{a} \notin Q(J_{\mathrm{opt}})$ implies that $\Psi(\mathbf{a}) \notin Q'(\Phi(J_{\mathrm{opt}}))$, and hence, $\Phi(J_{\mathrm{opt}})$ is a solution for $\Phi(I)$ and $\Psi(\mathbf{a})$. So now, we conclude the following.

— Recall that $Q(J_{\mathrm{opt}}) \subseteq Q(\Phi_I^{-1}(\Phi(J_{\mathrm{opt}})))$; hence,

$$|Q(J_{\mathrm{opt}})| \leq |Q(\Phi_I^{-1}(\Phi(J_{\mathrm{opt}})))|\,.$$

— Since Ψ is a bijection from $Q(\Phi_I^{-1}(\Phi(J_{\mathrm{opt}})))$ to $Q'(\Phi(J_{\mathrm{opt}}))$, we have that

$$|Q(\Phi_I^{-1}(\Phi(J_{\mathrm{opt}})))| = |Q'(\Phi(J_{\mathrm{opt}}))|\,.$$

— Since $\Phi(J_{\mathrm{opt}})$ is a solution for $\Phi(I)$ and $\Psi(\mathbf{a})$, we have that

$$|Q'(\Phi(J_{\mathrm{opt}}))| \leq |Q'(J'_{\mathrm{opt}})|\,.$$

— Since J' is an α-optimal solution, we have that

$$|Q'(J'_{\mathrm{opt}})| \leq |Q'(J')|/\alpha.$$

— Finally, since Ψ is a bijection from $Q(J)$ to $Q'(J')$, we have that

$$|Q'(J')| = |Q(J)|\,.$$

We conclude that $Q(J_{\mathrm{opt}}) \leq Q(J)/\alpha$, and hence, J is α-optimal, as claimed. □

Since we use fact-wise reductions for complexity analyses, our interest is in those that can be applied in polynomial time; this property will be obvious in the fact-wise reductions we later present.

5.3. Setting and Assumptions

To prove Theorem 4.5, in the remainder of this section we will fix a CQ Q, such that Q has neither self joins nor the head-domination property. We assume that $\mathrm{arity}(Q) = k$, and that the tuple $\mathbf{y}$ of head variables of Q is $(y_1, \ldots, y_k)$.

We will show a reduction from MAXDP$\langle Q_2^\star\rangle$ to MAXDP$\langle Q\rangle$. Recall the CQ $Q_2^\star$:

$$Q_2^\star(y_1, y_2) :- R_1(x, y_1), R_2(x, y_2)$$

So, in the remainder of this section we also fix an input $I^\star$ and $\mathbf{a}^\star$ for MAXDP$\langle Q_2^\star\rangle$. Without loss of generality, we assume that $\mathbf{a}^\star = (\diamond, \diamond)$, which we also denote by $\diamond^2$. We next construct input for MAXDP$\langle Q\rangle$—an instance I over $\mathrm{schema}(Q)$ and the tuple $\mathbf{a} = \diamond^k$.

A constant d such that $I^\star$ contains the tuples $R_1(d, \diamond)$ and $R_2(d, \diamond)$ is called a *joining* constant. We make the following assumptions about the instance $I^\star$.

(1) Each constant that appears on the left column of either R_1 or R_2 is a joining constant.
(2) Each joining constant appears at most 7 times in $I^\star$.
(3) For each joining constant d there is a constant $c \neq \diamond$, such that $R_i(d, c) \in I$ for some $i \in \{1, 2\}$.
(4) Each constant $c \neq \diamond$ that appears on the right column of either R_1 or R_2 appears at most 3 times in $I^\star$.

Note that we can make these assumptions since they hold in the instance constructed in the reduction from Max35NMSat to MAXDP$\langle Q_2^\star\rangle$ for the proof of Lemma 5.1.

Next, we fix a component P of $\mathcal{G}_\exists(Q)$, such that P is head-dominated by none of the atoms of Q. We denote by $\mathsf{Var}_\mathsf{h}(P)$ and $\mathsf{Var}_\exists(P)$ the sets of head variables and existential variables, respectively, that occur in the atoms of P. By $\mathsf{Var}(P)$ we denote the set of all the variables that appear in P (that is, $\mathsf{Var}(P) = \mathsf{Var}_\mathsf{h}(P) \cup \mathsf{Var}_\exists(P)$). We say that two variables $z, z' \in \mathsf{Var}_\mathsf{h}(P)$ are *atomic neighbors* if Q contains an atom ϕ that contains both z and z' (note that ϕ is not necessarily in P).

Our reduction considers two cases, based on properties of Q. The first case is where some $y, y' \in \mathsf{Var}_\mathsf{h}(P)$ are not atomic neighbors, and in the second we assume that every pair of variables in $\mathsf{Var}_\mathsf{h}(P)$ are atomic neighbors.

5.4. Case 1: Head Variables that are Not Atomic Neighbors

Recall that $(y_1, \ldots, y_k)$ is the tuple of head variables of Q. In the case considered here, we assume that some $y_i, y_j \in \mathsf{Var}_\mathsf{h}(P)$ are not atomic neighbors (hence, $i \neq j$). Assume, without loss of generality, that $i = 1$ and $j = 2$ (that is, $y_1, y_2 \in \mathsf{Var}_\mathsf{h}(P)$ and no atom contains both y_1 and y_2). Next, we construct a polynomial-time fact-wise reduction (Φ, Ψ) from MAXDP$\langle Q_2^\star\rangle$ to MAXDP$\langle Q\rangle$. This is sufficient for proving Theorem 4.5 in the current case, due to Lemmas 5.1 and 5.3.

The function Φ is defined as follows. For a fact $f_1 = R_1(d, c_1)$, we define $\Phi(f_1)$ to be the set of all facts f that are obtained from an atom ϕ of Q that does not include y_2 by:

— replacing every existential variable $x \in \mathsf{Var}_\exists(P)$ with d,
— replacing y_1 with c_1, and
— replacing every other variable with $\diamond$.

Similarly, for a fact $f_2 = R_2(d, c_2)$, we define $\Phi(f_2)$ to be the set of all facts f that are obtained from an atom ϕ of Q that does not include y_1 as above, except that y_2 is replaced with c_2. The function Ψ concatenates the given 2-tuple (c_1, c_2) with $k - 2$ occurrences of $\diamond$ (where k is the arity of Q). To complete the proof, it remains to prove that (Φ, Ψ) is indeed a fact-wise reduction.

LEMMA 5.4. *(Φ, Ψ) is a fact-wise reduction from* MAXDP$\langle Q_2^\star\rangle$ *to* MAXDP$\langle Q\rangle$.

PROOF. Let J be a sub-instance of $\Phi(I^\star)$, and denote $\Phi^{-1}_{I^\star}(J)$ by $J^\star$. It suffices to show that Ψ is a bijection from $Q_2^\star(J^\star)$ to $Q(J)$. The fact that Ψ is injective is obvious from its definition. To prove that Ψ is surjective, suppose that $\mathbf{b} = (c_1, c_2, d_1, \ldots, d_{k-2})$

is a tuple in $Q(J)$ that is realized by the match μ for Q in J. From the construction of Φ it follows that for $z \in \mathsf{Var}(Q)$ we have $\mu(z) = \diamond$ in the following cases:

(1) z is a head variable that is different from y_1 and y_2.
(2) z is an existential variable that is not in P

Due to (1), each d_i ($i = 1, \ldots, k-2$) is $\diamond$. So it remains to show that $(c_1, c_2) \in Q_2^\star(J^\star)$, that is, $J^\star$ contains both $R_1(d, c_1)$ and $R_2(d, c_2)$ for some constant d. Let ϕ_1 and ϕ_2 be atoms in P that contain y_1 and y_2, respectively. Let x_1 and x_2 be existential variables in ϕ_1 and ϕ_2, respectively. Observe that x_1 and x_2 exist, or otherwise ϕ_1 and ϕ_2 would not be in the same connected component (i.e., P). Denote $\mu(x_1)$ and $\mu(x_2)$ by d_1 and d_2, respectively. From the definition of Φ it follows that the fact $\mu(\phi_1)$ and $\mu(\phi_2)$ belong to $\Phi(f_1)$ and $\Phi(f_2)$, respectively, where $f_1 = R_1(d_1, c_1)$ and $f_2 = R_2(d_2, c_2)$ are facts of $I^\star$.

It follows from the definition of Φ that for every fact $\mu(\phi)$, where $\phi \in P$, the attributes that correspond to existential variables of ϕ are equal. Therefore, P being a connected component of $\mathcal{G}_\exists(Q)$ implies that all the existential variables of P are assigned the same constant by μ. In particular, $d_1 = d_2$, and let us denote d_1 by d. So, it remains to show that $J^\star$ contains both f_1 and f_2.

From our conclusion that $\mu(z) = \diamond$ in (1) and (2) above, and our conclusion that μ maps all the existential variables of P to d, we further conclude that the facts in the image of μ include all the facts in $\Phi(f_1)$ and all the facts in $\Phi(f_2)$. Therefore, the definition of $\Phi_{I^\star}^{-1}(\cdot)$ implies that $J^\star$ contains both f_1 and f_2, as required. □

5.5. Case 2: All Pairs are Atomic Neighbors

In this case, which complements the previous case, we assume that every two head variables in $\mathsf{Var}_\mathsf{h}(P)$ are atomic neighbors. As intuition of our proof for this case, consider the following CQ.

$$Q(y_1, y_2, y_3) :- S_1(x, y_1, y_2), S_2(x, y_2, y_3), S_3(x, y_1, y_3), S_4(y_1, y_2)$$

Here, P consists of the first three atoms. As in the previous case, we will construct a reduction from MAXDP$\langle Q_2^\star \rangle$ to MAXDP$\langle Q \rangle$. This reduction is not a fact-wise reduction, and it is actually fairly involved. As a possible scenario, we will use the atom $S_3(x, y_1, y_3)$ to simulate $R_1(x, y_1)$, and the atom $S_2(x, y_2, y_3)$ to simulate $R_2(x, y_2)$. This means that if the tuple $S_3(d, c_1, \diamond)$ is deleted in the generated input for MAXDP$\langle Q \rangle$, then $R_1(d, c_1)$ will be deleted in the original input for MAXDP$\langle Q \rangle$ (and the same goes for $S_2(d, c_2, \diamond)$ and $R_2(d, c_2)$). However, for that to succeed we need to make sure that $S_4(\diamond, \diamond)$ remains in the approximate solution. So, we make $S_4(\diamond, \diamond)$ too valuable to delete, by introducing as query answers many tuples of the form $(\diamond, \diamond, \nabla_i)$ where i is an integer (and of course, $S_4(\diamond, \diamond)$ is needed for each of these solutions). Similarly, we want to make sure that every $S_1(d, \diamond, \diamond)$ is in the approximate solution, and for that we will introduce as query answers many tuples of the form $(\diamond, \diamond, \heartsuit_{d,i})$, where i is an integer, such that $S_1(d, \diamond, \diamond)$ is necessary for obtaining $(\diamond, \diamond, \heartsuit_{d,i})$. In the reduction, $S_1(x, y_1, y_2)$ and $S_4(y_1, y_2)$ are called *encounters* since each of them contains both y_1 and y_2. Moreover, $S_1(x, y_1, y_2)$ is called an *internal* encounter since it is in P, and $S_4(y_1, y_2)$ is called an *external* encounter since it is outside P. As hinted here, we will handle internal and external encounters differently.

For the above intuitive solution to work, we need to carefully choose the atoms ϕ_1 and ϕ_2 of Q that simulate $R_1(x, y_1)$ and $R_2(x, y_2)$, respectively, and we need to use the fact that every two head variables in $\mathsf{Var}_\mathsf{h}(P)$ are atomic neighbors. Moreover, we need to make sure that we are not introducing too many new answers, or else we will lose the constant approximation ratio. Next, we give the precise reduction.

We choose the atoms $\phi_1, \phi_2 \in P$, as follows. As ϕ_1 we choose an atom of P with a maximal number of head variables. Note that at least one atom of P has a head

variable that is not in ϕ_1, or otherwise P is head-dominated by ϕ_1. So, to choose ϕ_2 we consider the atoms of P that include one or more head variables not in ϕ_1, and select such an atom with a maximal number of head variables. In other words, we choose ϕ_1 and ϕ_2 as follows.

$$\phi_1 \stackrel{\text{def}}{=} \mathbf{argmax}_{\phi}\{|\mathsf{Var_h}(\phi)| \mid \phi \in P\} \tag{6}$$

$$\phi_2 \stackrel{\text{def}}{=} \mathbf{argmax}_{\phi}\{|\mathsf{Var_h}(\phi)| \mid \phi \in P \wedge \mathsf{Var_h}(\phi) \not\subseteq \mathsf{Var_h}(\phi_1)\} \tag{7}$$

Note that $\mathsf{Var_h}(\phi_2) \not\subseteq \mathsf{Var_h}(\phi_1)$ and $\mathsf{Var_h}(\phi_1) \not\subseteq \mathsf{Var_h}(\phi_2)$ (since $|\mathsf{Var_h}(\phi_1)|$ is maximal among the atoms in P). W.l.o.g., we assume that y_1 and y_2 are head variables of ϕ_1 and ϕ_2, respectively, such that y_1 is not in ϕ_2 and y_2 is not in ϕ_1; that is:

$$y_1 \in \mathsf{Var_h}(\phi_1) \setminus \mathsf{Var_h}(\phi_2) \qquad y_2 \in \mathsf{Var_h}(\phi_2) \setminus \mathsf{Var_h}(\phi_1)$$

As said above, we call an atom ϕ of Q an *encounter* if ϕ contains both y_1 and y_2. Note that the assumption of the case we consider (Case 2) implies that at least one encounter exists. The encounter ϕ is *internal* if $\phi \in P$ and *external* if $\phi \notin P$.

To construct the instance I, we start with $I = \emptyset$ and proceed with three main steps.

(1) We first consider every pair $R_1(d, c_1)$ and $R_2(d, c_2)$ of facts of $I^\star$ (giving rise to $(c_1, c_2) \in Q_2^\star(I^\star)$), and we add to I every fact $\lambda_{d,c_1,c_2}(\psi)$, such that $\psi \in \text{atoms}(Q)$ and λ_{d,c_1,c_2} is the assignment for Q defined similarly to μ in Case 1; that is:

$$\lambda_{d,c_1,c_2}(z) = \begin{cases} c_i & \text{if } z = y_i \text{ where } i \in \{1,2\} \\ d & \text{if } z \in \mathsf{Var}_\exists(P) \\ \diamond & \text{otherwise.} \end{cases} \tag{8}$$

(2) Define $N^\star$ to be the number of answers for $Q_2^\star$ in $I^\star$; that is,

$$N^\star \stackrel{\text{def}}{=} |Q_2^\star(I^\star)|$$

For each external encounter ϕ and $i \in \{1, \ldots, N^\star\}$, we define a fresh new constant $\nabla_{\phi,i}$ and add to I every fact $\nu_{\phi,i}(\psi)$, such that $\psi \in \text{atoms}(Q)$, and $\nu_{\phi,i}$ is the assignment for Q that is defined as follows:

$$\nu_{\phi,i}(z) = \begin{cases} \nabla_{\phi,i} & \text{if } z \in \mathsf{Var}(P) \setminus \mathsf{Var}(\phi) \\ \diamond & \text{otherwise.} \end{cases} \tag{9}$$

That is, $\nu_{\phi,i}$ maps to $\nabla_{\phi,i}$ all the variables of P, except for those that are also in ϕ; every remaining variable is mapped to $\diamond$. Note that ϕ being an external encounter implies that none of the existential variables of ϕ occurs in P (since P is a component of $\mathcal{G}_\exists(Q)$); hence, a variable of P is mapped to $\diamond$ if and only if this variable is a head variable of ϕ.

(3) Finally, for each internal encounter ϕ, joining constant d (from $I^\star$) and $i \in \{1, \ldots, 6\}$, we define a fresh new constant $\heartsuit_{\phi,d,i}$ and add to I every fact $\xi_{\phi,d,i}(\psi)$, such that $\psi \in \text{atoms}(Q)$ and $\xi_{\phi,d,i}$ is the assignment for Q defined as follows:

$$\xi_{\phi,d,i}(z) = \begin{cases} \heartsuit_{\phi,d,i} & \text{if } z \in \mathsf{Var_h}(P) \setminus \mathsf{Var_h}(\phi) \\ d & \text{if } z \in \mathsf{Var}_\exists(P) \\ \diamond & \text{otherwise.} \end{cases} \tag{10}$$

In other words, $\xi_{\phi,d.i}$ is the same as $\nu_{\phi,i}$, except that $\heartsuit_{\phi,d,i}$ is used instead of $\nabla_{\phi,i}$, and every existential variable of P is mapped to d (rather than to $\nabla_{\phi,i}$).

This completes the construction of I, and the description of our reduction. Next, we show the correctness of this reduction.

Consider a match μ for Q in I. We call μ:

— a λ-*match* if $\mu = \lambda_{d,c_1,c_2}$, as defined in (8), for facts $R_1(d, c_1), R_2(d, c_2) \in I^\star$;
— a ν-*match* if $\mu = \nu_{\phi,i}$, as defined in (9), for some external encounter ϕ and number i;
— a ξ-match if $\mu = \xi_{\phi,d,i}$, as defined in (10), for some internal encounter ϕ, a joining constant d, and a number i.

The following key lemma shows that every match for Q in I is either a λ-match, a ν-match, or a ξ-match; in other words, the construction of I is such that no new matches for Q in I exist beyond those used for defining I.

LEMMA 5.5. *If μ is a match for Q in I, then exactly one of the following holds:*

(*1*) *μ is a λ-match,*
(*2*) *μ is a ν-match.*
(*3*) *μ is a ξ-match.*

PROOF. Consider a match μ for Q in I. The match μ cannot be of more than one type; as evidence, observe that $\mu_1(\mathbf{y}) \neq \mu_2(\mathbf{y})$ whenever μ_1 and μ_2 are of different types, since a ν-match contains at least one $\nabla_{\phi,i}$ (while the others do not), and a ξ-match contains at least one $\heartsuit_{\phi,d,i}$ (while the others do not). So, we need to show that μ is of one of the three types.

If a variable z in $\mathsf{Var}(Q)$ does not appear in P, then the construction of I is such that every attribute that corresponds to z has the constant $\diamond$, and therefore, $\mu(z) = \diamond$ must hold. So we consider just the variables that occur in P. Let $\phi \in P$ be an atom. The construction of I is such that the constants of a fact matching ϕ in positions that correspond to existential variables are equal. And since P is a connected component of $\mathcal{G}_\exists(Q)$, the match μ maps all the variables in $\mathsf{Var}_\exists(P)$ to the same value (which is either a joining constant d or a constant of the form $\nabla_{\phi,i}$); this is true, since every pair of neighbors in $\mathcal{G}_\exists(Q)$ need to agree on their assignments to the common existential variables by μ (and the definition of $\mathcal{G}_\exists(Q)$ implies that at least one such common existential variable exists). Note that P has at least one existential variable (since P is a component of $\mathcal{G}_\exists(Q)$ and P has at least two atoms); let x be such a variable, and let $d = \mu(x)$. Next, we will consider several cases, based on properties of $\mu(\mathbf{y})$, and show that in each case μ is of one of the three types.

Case 1a: $\mu(\mathbf{y}) = \diamond^k$. We will show that $\mu = \lambda_{d,\diamond,\diamond}$. It suffices to prove that d is a joining constant, or equivalently, that d cannot be any $\nabla_{\phi,i}$. So consider such $\nabla_{\phi,i}$. Let $\psi \in P$ be an atom, such that ψ contains a head variable $y \in \mathsf{Var}_\mathsf{h}(P) \setminus \mathsf{Var}_\mathsf{h}(\phi)$. Note that ψ contains an existential variable, say x', since $\psi \in P$. If $d = \nabla_{\phi,i}$, then $\mu(x') = \nabla_{\phi,i}$ and $\mu(z) = \diamond$. But then, from the construction of I, specifically (9), it follows that no fact such as $\mu(\psi)$ exists in I.

Case 1b: $\mu(\mathbf{y})$ contains a constant $c \neq \diamond$ from I. Define $\mu(y_1) = c_1$ and $\mu(y_2) = c_2$. We will show that $\mu = \lambda_{d,c_1,c_2}$. Note that $c_1 = c$ or $c_2 = c$ (or both). Suppose, w.l.o.g., that $c_1 = c$ (the case where $c_2 = c$ is handled symmetrically). Then $\mu(\phi_1)$ contains both c and d. In particular, d is necessarily a joining constant of $I^\star$, since from (8)–(10) it follows that no $\nabla_{\phi,i}$ co-exists with c in the same fact. It remains to show that $\mu(y) = \lambda_{d,c_1,c_2}(y)$ for each head variable y in P. So let y be such a head variable. If y is either y_1 or y_2, then the claim is obvious. Otherwise, we need to show that $\mu(y) = \diamond$. Since y is an atomic neighbor of y_1, there is an atom ψ of Q that contains both y and y_1. Since $\mu(y_1)$ is a constant of I that is not $\diamond$, we conclude from (8)–(10) that I cannot contain $\mu(\psi)$ if $\mu(y)$ is any constant different from $\diamond$. Hence, $\mu(y) = \diamond$, as required.

Case 2: $\mu(\mathbf{y})$ contains $\nabla_{\phi,i}$ for some external encounter ϕ and number i. We will show that $\mu = \nu_{\phi,i}$. Let $y \in \mathsf{Var}_\mathsf{h}(P)$ be such that $\mu(y) = \nabla_{\phi,i}$. Since at least one variable in $\mathsf{Var}_\exists(P)$, say x', occurs in the same atom as y (and μ maps that atom to a fact of

I), from (9) we get that $\mu(x') = \nabla_{\phi,i}$, and hence, $d = \nabla_{\phi,i}$. It remains to show that for $z \in \mathsf{Var_h}(P)$, the constant $\mu(z)$ is as defined in (9). But this is obvious from the fact that each such z co-exists in the same atom with some existential variable x' of P (and x' is mapped to $\nabla_{\phi,i}$), and the only time where such a fact is added into I is within the match $\nu_{\phi,i}$.

Case 3: $\mu(\mathbf{y})$ contains a constant $\heartsuit_{\phi,d',i}$ for some internal encounter ϕ, a joining constant d' and a number i. We will show that $\mu = \xi_{\phi,d,i}$. Let $y \in \mathsf{Var_h}(P)$ be such that $\mu(y) = \heartsuit_{\phi,d',i}$. Since y occurs with some existential variable in the same atom of P, that variable is mapped by μ to d', and hence, $d = d'$. Furthermore, since y is an atomic neighbor of each head variable z of P, the construction of I implies that $\mu(z)$ must be as defined in (10) for ϕ, d and i. □

The following lemma is a direct corollary of Lemma 5.5.

LEMMA 5.6. *$Q(I)$ consists of the following tuples.*

(*1*) *$\lambda_{d,c_1,c_2}(\mathbf{y})$ for every λ-match λ_{d,c_1,c_2}, or in other words, $(c_1, c_2, \diamond, \ldots, \diamond)$ for every fact $(c_1, c_2) \in Q_2^\star(I^\star)$.*
(*2*) *$\nu_{\phi,i}(\mathbf{y})$ for every ν-match $\nu_{\phi,i}$.*
(*3*) *$\xi_{\phi,d,i}(\mathbf{y})$ for every ξ-match $\xi_{\phi,d,i}$.*

Next, we give some further notation. Recall that $N^\star = |Q_2^\star(I^\star)|$. We also define the following.

— $M^\star_{\text{join}}$ is the number of joining constants.
— ext is the number of external encounters in $\text{atoms}(Q)$.
— int is the number of internal encounters in $\text{atoms}(Q)$.
— enc is the number of encounters in $\text{atoms}(Q)$, that is, the sum $ext + int$.

From Lemma 5.6 we conclude the following.

LEMMA 5.7. $|Q(I)| = N^\star + ext \cdot N^\star + 6int \cdot M^\star_{\text{join}}$.

For the next lemma, recall our choice of the atoms ϕ_1 and ϕ_2 from (6) and (7).

LEMMA 5.8. *If ϕ is an internal encounter, then $\mathsf{Var_h}(\phi_1) \not\subseteq \mathsf{Var_h}(\phi)$ and $\mathsf{Var_h}(\phi_2) \not\subseteq \mathsf{Var_h}(\phi)$.*

PROOF. Recall from (6) and (7) how ϕ_1 and ϕ_2 were chosen. Since ϕ contains both y_1 and y_2, we have $\mathsf{Var_h}(\phi) \not\subseteq \mathsf{Var_h}(\phi_1)$ (since $y_2 \notin \mathsf{Var_h}(\phi_1)$) and $\mathsf{Var_h}(\phi) \not\subseteq \mathsf{Var_h}(\phi_2)$ (since $y_1 \notin \mathsf{Var_h}(\phi_2)$). From the choices of ϕ_1 and ϕ_2 it holds that $|\mathsf{Var_h}(\phi)| \leq |\mathsf{Var_h}(\phi_1)|$ and $|\mathsf{Var_h}(\phi)| \leq |\mathsf{Var_h}(\phi_2)|$. The lemma then follows immediately. □

From Lemma 5.8 we conclude the following.

LEMMA 5.9. *Let λ_{d,c_1,c_2} be a λ-match, and let μ be either a ν-match or a ξ-match. Then $\mu(\phi_1) \neq \lambda_{d,c_1,c_2}(\phi_1)$ and $\mu(\phi_2) \neq \lambda_{d,c_1,c_2}(\phi_2)$.*

PROOF. Suppose first that μ is a ν-match $\nu_{\phi,i}$. Then ν maps the existential variables of P to $\nabla_{\phi,i}$, and since ϕ_1 and ϕ_2 are in P (and hence, contain existential variables), we get the claim of the lemma since neither $\lambda_{d,c_1,c_2}(\phi_1)$ nor $\lambda_{d,c_1,c_2}(\phi_2)$ contains $\nabla_{\phi,i}$.

Now suppose that μ is a ξ-match $\xi_{\phi,d,i}$. Note that ϕ is an internal encounter. Due to (10) and Lemma 5.8, both $\xi_{\phi,d,i}(\phi_1)$ and $\xi_{\phi,d,i}(\phi_2)$ contain $\heartsuit_{\phi,d,i}$. But neither $\lambda_{d,c_1,c_2}(\phi_1)$ nor $\lambda_{d,c_1,c_2}(\phi_2)$ contains $\heartsuit_{\phi,d,i}$. □

The proof of Lemma 5.5 (specifically, Case 1a) also shows the following lemma.

LEMMA 5.10. *Let μ be a match for Q in I. If $\mu(\mathbf{y}) = \diamond^k$, then $\mu = \lambda_{d,\diamond,\diamond}$ for some joining constant d, and μ is neither a ν-match nor a ξ-match.*

From Lemmas 5.9 and 5.10, we conclude the following.

LEMMA 5.11. *There is a solution J' for I and $\diamond^k$, such that J' has the following properties.*

(1) Every ν-match for Q in I is a match for Q in J'.
(2) Every ξ-match for Q in I is a match for Q in J'.
(3) $|Q(J')| \geq N^\star/2 - 1 + ext \cdot N^\star + 6int \cdot M^\star_{\text{join}}$.

Moreover, J' can be obtained in polynomial time.

PROOF. Each tuple in $Q_2^\star(I^\star)$ is of one or more of the following three types:

(1) (c_1, c_2), where $c_1 \neq \diamond$.
(2) (c_1, c_2), where $c_2 \neq \diamond$.
(3) $(\diamond, \diamond)$.

Now, for some i in $\{1, 2\}$, at least $N^\star/2 - 1$ tuples of $Q_2^\star(I^\star)$ are of Type (i). Suppose, w.l.o.g., that this is the case for $i = 1$. Let J' be obtained from I by removing the facts $\lambda_{d,\diamond,\diamond}(\phi_1)$ for all joining constants d. Due to Lemma 5.10, J' is indeed a solution. Recall that ϕ_1 contains y_1; hence, due to Lemma 5.6, $Q(J')$ contains every fact $(c_1, c_2, \diamond, \ldots, \diamond)$ where $(c_1, c_2) \in Q_2^\star(I^\star)$ is of Type (1). Finally, as a result of Lemma 5.9, every ν-match or ξ-match for Q in I is a match for Q in J'. Then, Property (3) in the lemma easily follows. □

LEMMA 5.12. *Let J be a solution for I and $\diamond^k$. One can obtain from J, in polynomial time, a solution J' with the following properties.*

(1) Every ν-match for Q in I is a match for Q in J'.
(2) Every ξ-match for Q in I is a match for Q in J'.
(3) $|Q(J')| \geq |Q(J)|$.

PROOF. W.l.o.g., we can assume that for each fact $f \in I \setminus J$ it is the case that $\diamond^k \in Q(J \cup \{f\})$, or in other words, $J \cup \{f\}$ is not a solution (otherwise, we add f to J). Suppose that some $\nu_{\phi,i}$ (where ϕ is an external encounter) is not a match for Q in J. Let f be a fact of I, such that $f = \nu_{\phi,i}(\psi)$ for some atom ψ of Q, and $f \notin J$. Due to Lemma 5.10, the fact f cannot contain $\nabla_{\phi,i}$; hence, due to (9), f contains only $\diamond$s. But then, this means that for all i, no $\nu_{\phi,i}$ is a match. Then, from Lemma 5.6 we get that $Q(J)$ contains at most $N^\star + 6int \cdot M^\star_{\text{join}}$ tuples. Since $ext > 0$ (otherwise, $\nu_{\phi,i}$ does not exist), we can take as J' the solution from Lemma 5.11.

Now, suppose that some $\xi_{\phi,d,i}$ (where ϕ is an internal encounter) is not a match for Q in J. Let f be a fact of I, such that $f = \xi_{\phi,d,i}(\psi)$ for some atom ψ of Q, and $f \notin J$. Due to Lemma 5.10, the fact f cannot contain $\heartsuit_{\phi,d,i}$; hence, due to (10), f contains only $\diamond$s and d's. Suppose first that f contains only $\diamond$s. Then $\xi_{\phi,d,i}$ is not a match for all joining constants d and all $i = 1, \ldots, 6$. Then, from Lemma 5.6 we get that $Q(J)$ misses at least $6M^\star_{\text{join}}$ tuples of $Q(I)$, which are the $\xi_{\phi,d,i}(\mathbf{y})$. It also misses the fact $\diamond^k$ of $Q(I)$. From our assumptions on $I^\star$ stated in Section 5.3 it follows that $M^\star_{\text{join}} \geq N^\star/12$. Thus, $Q(J)$ misses at least $N^\star/2 + 1$ facts of I. Due to Lemma 5.6, the solution J' of Lemma 5.11 is at least as good as J.

It remains to consider the case where f contains at least one occurrence of d (in a position that corresponds to an existential variable). Due to the previous case, we assume that every $\xi_{\phi,d,i}(\psi)$ that consists of only $\diamond$s is already in J. We would like to add f to J, but for that we need to make sure that we do not give rise to the match $\lambda_{d,\diamond,\diamond}$.

So we remove the fact $\lambda_{d,\diamond,\diamond}(\phi_1)$. Due to Lemma 5.9, by doing so we did not eliminate any ξ-mapping. However, we may have eliminated mappings $\lambda_{d,\diamond,c}$ where $c \neq \diamond$ is such that $R_2(d,c) \in I$. Nevertheless, we have lost at most 5 such mappings, since I can have at most 5 facts $R_2(d,c)$ (since I contains at most 7 occurrences of d, and two of them are with $\diamond$). Thus, by removing $\lambda_{d,\diamond,\diamond}(\phi_1)$ we eliminate at most 5 answers. To compensate for that, we add to J all the facts $\xi_{\phi,d,i}(\psi)$ where $\psi \in \text{atoms}(Q)$ (if these facts are not already in J), for all $i \in \{1,\ldots,6\}$. By doing so, we get the 6 new answers that are obtained by the mappings $\xi_{\phi,d,1},\ldots,\xi_{\phi,d,6}$; these are indeed new answers, since all of them require f (which was missing from J). □

We now choose a solution J for I and $\diamond^k$, and we assume that J satisfies Properties 1 and 2 of Lemma 5.12. Furthermore, we assume that every fact $f \in I \setminus J$ satisfies $\diamond^k \in Q(J \cup \{f\})$.

LEMMA 5.13. *A λ-match λ_{d,c_1,c_2} is a match for Q in J if and only if both $\lambda_{d,c_1,c_2}(\phi_1)$ and $\lambda_{d,c_1,c_2}(\phi_2)$ are Q-useful in J.*

PROOF. The "only if" direction is straightforward. We will prove the "if" direction. Suppose that both $\lambda_{d,c_1,c_2}(\phi_1)$ and $\lambda_{d,c_1,c_2}(\phi_2)$ are Q-useful in J. We need to show that λ_{d,c_1,c_2} is a match for Q in J. From Lemma 5.9 it follows that $\lambda_{d,c_1,c_2}(\phi_1)$ and $\lambda_{d,c_1,c_2}(\phi_2)$ are used in λ-matches of Q in J, say $\lambda_{d,c_1,c_2'}$ and λ_{d,c_1',c_2}, respectively. We will show that $\lambda_{d,c_1,c_2}(\phi) \in J$ or every atom ϕ of Q. So, let ϕ be an atom of Q. If ϕ does not contain y_1, then $\lambda_{d,c_1,c_2}(\phi) = \lambda_{d,c_1',c_2}(\phi) \in J$. Similarly, if ϕ does not contain y_2 then $\lambda_{d,c_1,c_2}(\phi) = \lambda_{d,c_1,c_2'}(\phi) \in J$. If ϕ contains both y_1 and y_2, then ϕ is an encounter, and there are two options: if either $c_1 \neq \diamond$ or $c_2 \neq \diamond$, then $\lambda_{d,c_1,c_2}(\phi) \in J$ since there is no reason to delete it; and if $c_1 = c_2 = \diamond$, then $\lambda_{d,c_1,c_2}(\phi)$ is used in either a ν-match or a ξ-match (and we assume that each such a match is in J). □

We obtain a solution $J^\star$ for $I^\star$ and $\diamond^2$ as follows. For each match λ_{d,c_1,c_2} for Q in J we include in $J^\star$ the facts $R_1(d,c_1)$ and $R_2(d,c_2)$. Due to Lemma 5.13, $J^\star$ is indeed a solution. The following lemma follows straightforwardly from the construction of J and $J^\star$, and the lemmas thus far.

LEMMA 5.14. *The following hold.*

(1) $Q_2^\star(J^\star) = \{(c_1,c_2) \in Q_2^\star(I^\star) \mid (c_1,c_2,\diamond,\ldots,\diamond) \in Q(J)\}$.
(2) $|Q(J)| = |Q_2^\star(J^\star)| + ext \cdot N^\star + 6int \cdot M^\star_{\text{join}}$.

Let $J^\star_{\text{opt}}$ be an optimal solution for $I^\star$ and $\diamond^2$, and let J_{opt} be an optimal solution for I and $\diamond^k$. Using Lemmas 5.9, 5.12 and 5.13, one can easily prove the following.

LEMMA 5.15. $|Q(J_{\text{opt}})| = |Q_2^\star(J^\star_{\text{opt}})| + ext \cdot N^\star + 6int \cdot M^\star_{\text{join}}$.

We can now finalize the proof. Denote $m_Q = ext + 18int$.

LEMMA 5.16. *For $\alpha^\star \in (0,1)$, if J is α_Q-optimal where $\alpha_Q = (\alpha^\star + 3m_Q)/(1+3m_Q)$, then $J^\star$ is $\alpha^\star$-optimal.*

PROOF. Lemmas 5.14 and 5.15 state the following.

$$|Q(J_{\text{opt}})| = |Q_2^\star(J^\star_{\text{opt}})| + ext \cdot N^\star + 6int \cdot M^\star_{\text{join}}$$
$$|Q(J)| = |Q_2^\star(J^\star)| + ext \cdot N^\star + 6int \cdot M^\star_{\text{join}}$$

Therefore, assuming J is α_Q-optimal, the following hold.

$$\begin{aligned}|Q_2^\star(J^\star)| &= |Q(J)| - ext \cdot N^\star - 6int \cdot M^\star_{\text{join}} \\ &\geq \alpha_Q |Q(J_{\text{opt}})| - ext \cdot N^\star - 6int \cdot M^\star_{\text{join}} \\ &= \alpha_Q |Q_2^\star(J^\star_{\text{opt}})| - (1-\alpha_Q)(ext \cdot N^\star + 6int \cdot M^\star_{\text{join}})\end{aligned}$$

Recall our assumption that, in $I^\star$, each constant $c \neq \diamond$ that appears on the right column of either R_1 or R_2 appears at most 3 times in $I^\star$. In particular, $M^\star_{\text{join}} \leq 3N^\star$. Thus, we have the following.

$$\begin{aligned}|Q_2^\star(J^\star)| &\geq \alpha_Q |Q_2^\star(J^\star_{\text{opt}})| - (1-\alpha_Q)(ext \cdot N^\star + 18int \cdot N^\star) \\ &= \alpha_Q |Q_2^\star(J^\star_{\text{opt}})| - m_Q \cdot N^\star \cdot (1-\alpha_Q)\end{aligned}$$

Recall that $|Q_2^\star(J^\star_{\text{opt}})| \geq N^\star/2 - 1$, which implies that $N^\star \leq 3|Q_2^\star(J^\star_{\text{opt}})|$. Therefore,

$$\begin{aligned}|Q_2^\star(J^\star)| &\geq \alpha_Q |Q_2^\star(J^\star_{\text{opt}})| - 3m_Q |Q_2^\star(J^\star_{\text{opt}})|(1-\alpha_Q) \\ &= |Q_2^\star(J^\star_{\text{opt}})|(\alpha_Q - 3m_Q(1-\alpha_Q))\end{aligned}$$

So we need to show that

$$\alpha_Q - 3m_Q(1-\alpha_Q) \geq \alpha^\star\,,$$

which is equivalent to

$$\alpha_Q \geq \frac{\alpha^\star + 3m_Q}{1+3m_Q}$$

which we get due to the assumption of the lemma. □

This completes the proof of Theorem 4.5. Next, we strengthen the theorem in two families of CQs.

5.6. Global Constant for Special CQ Families

In this section, we consider *acyclic* CQs and *binary* CQs, without self joins, and we show that for these two classes there is a global constant $\alpha \in (0,1)$ that bounds the approximability of $\text{MAXDP}\langle Q\rangle$ in the absence of head domination. By *global* we mean that α does not depend on Q, which is the reason that this result is not subsumed by Theorem 4.5. The formal result is the following.

THEOREM 5.17. *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* $\text{schema}(Q)$ *consists of binary relation symbols: if Q does not have the head-domination property, then it is NP-hard to α-approximate* $\text{MAXDP}\langle Q\rangle$.

We start with acyclic CQs, and our proof in this case is very simple: we show that in the proof of Theorem 4.5, Case 1 (Section 5.4) necessarily holds if Q is acyclic. In other words, our proof is by the following lemma.

LEMMA 5.18. *Let Q be an acyclic CQ, and let $Z \subseteq \mathsf{Var}(Q)$ be such that every two variables of Z are atomic neighbors. Then $Z \subseteq \mathsf{Var}(\phi)$ for some $\phi \in \text{atoms}(Q)$.*

PROOF. Recall that Q is *acyclic* if its hypergraph $\mathcal{H}(Q)$ is acyclic.[3] The standard definition of an acyclic hypergraph $H = (V, E)$ is that one can remove all the edges of

[3]Hypergraph acyclicity is also referred to as α-*acyclicity* [Fagin 1983].

H by repeatedly removing ears; an *ear* of H is an edge e that either intersects with no other edge, or there is a different edge e' where every node in the set $e \setminus e'$ appears only in e (and in no other edges $e'' \neq e$).

Consider an ear-removal process of $\mathcal{H}(Q)$ that removes all the hyperedges. Suppose that $e_1, \ldots, e_n$ are the edges of $\mathcal{H}(Q)$, in the order of their removal (i.e., e_1 is removed first, then e_2, an so on). Of course, $Z \subseteq e_1 \cup \cdots \cup e_n$. Let m be the minimal number, such that $Z \not\subseteq e_{m+1} \cup \cdots \cup e_n$ (note that m can be n). To prove the lemma, we will show that $Z \subseteq e_m$. Suppose otherwise (by way of contradiction). The choice of e_m implies that e_m contains a variable $z \in Z$, such that none of $e_{m+1}, \ldots, e_n$ contain z. Moreover, since $Z \subseteq e_m \cup \cdots \cup e_n$ and $Z \not\subseteq e_m$, there is a variable $z' \in Z$, such that $z' \notin e_m$, but $z' \in e_{m+1} \cup \cdots \cup e_n$. Now, the condition of the lemma says that there is an edge e_i that contains both z and z', and let j be the maximal such i. Since none of $e_m, \ldots, e_n$ contains both z and z', we have that $j < m$. But then, when e_j is removed it is an ear, which in particular implies that z and z' (that belong to $e_{j+1} \cup \cdots \cup e_n$) must co-occur in one of $e_{j+1}, \ldots, e_n$. This contradicts the fact that j is maximal. □

Note that, indeed, the intractable approximation ratio α of Case 1 does not depend on Q. Hence, Theorem 5.17 for acyclic CQs follows immediately from Lemma 5.18.

Next, we consider binary CQs. Let Q be a binary CQ without self joins, and assume that Q does not have the head-domination property. If Case 1 holds, then we are done since we can use the α of that case (as done for acyclic CQs). So, suppose that Case 2 holds (Section 5.5). Consider the nodes y_1 and y_2 that are chosen in that case. Observe that every encounter is necessarily external, since Q is binary (and, hence, an encounter does not contain existential variables).

The construction of the instance I changes as follows. Instead of introducing the assignments $\nu_{\phi,i}$ for *every* external encounter ϕ, we introduce them for just one (arbitrary) external encounter, say ϕ'. It is straightforward to show that Lemmas 5.12 and 5.13 remain correct. Moreover, due to the change in the construction of I, Lemmas 5.14, 5.15 and 5.16 remain correct for $int = 0$ and $ext = m_Q = 1$. Hence, due to Lemma 5.16 we get that $\alpha_Q = (\alpha^\star + 3m_Q)/(1 + 3m_Q) = (\alpha^\star + 3)/(1 + 3)$ does not depend on Q. Thus, we get Theorem 5.17 for binary CQs.

6. APPROXIMATIONS FOR SUNFLOWER CQS

A hypergraph is called a *sunflower hypergraph* if there is a set Z of nodes, such that $Z = e \cap e'$ for all distinct hyperedges e and e'. A CQ Q is a *sunflower CQ* if $\mathcal{H}(Q)$ is a sunflower hypergraph; in other words, Q is a sunflower CQ if every join variable of Q occurs in every atom of Q. Following the conventional terminology of sunflower hypergraphs, we will call the set of join variables of a sunflower CQ Q the *core* of Q. As an example, the CQ $Q_k^\star$ (defined in (3)) is a sunflower CQ for any k; and for $k > 1$, the core is $\{x\}$. As another example, every CQ with two or fewer atoms is a sunflower 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 indeed, they occur in each of the three atoms.

In this section, we present approximation algorithms for MAXDP$\langle Q \rangle$, where Q is a sunflower CQ without self joins. Observe that a sunflower CQ is acyclic. The following corollary of Theorem 5.17 (for the case of acyclic CQs) shows that every MAXDP$\langle Q \rangle$, where Q is a sunflower CQ without self joins, is intractable to approximate within some factor α, except for some very special cases.

COROLLARY 6.1. *There is a number $\alpha \in (0,1)$, such that for all sunflower CQs Q without self joins, α-approximating $\text{MAXDP}\langle Q\rangle$ is NP-hard, unless Q has one (or both) of the following properties:*

(*1*) *Every join variable is a head variable.*
(*2*) *One of the atoms contains all the head variables.*

The proof of Corollary 6.1 is straightforward by observing that in the case of a sunflower CQ Q, head domination is equivalent to the disjunction of the two properties in the corollary.

The constant factor α that we have for the hardness part of Corollary 6.1 is fairly close to 1 (it is larger than 0.99), so this result does not preclude good approximations (though it does preclude PTAS algorithms). Recall from Proposition 3.4 that for every CQ Q without self joins, $\text{MAXDP}\langle Q\rangle$ is $1/k$-approximated using the unidimensional algorithm, where $k = \min\{\text{arity}(Q), |\text{atoms}(Q)|\}$. Later in this section, we will give constant-factor approximation algorithms for $\text{MAXDP}\langle Q\rangle$, where the factor does not depend on Q, assuming that Q is a sunflower CQ without self joins. We will also show how to extend one of the approximations to a significant generalization of sunflower CQs. But first, we give a general reduction that will allow us to describe our algorithms for the simple CQ $Q^\star_k$, rather than for an arbitrary sunflower CQ.

6.1. Reduction for Sunflower CQs

Let Q be a sunflower CQ with k atoms and without self joins. Intuitively, we reduce $\text{MAXDP}\langle Q\rangle$ to $\text{MAXDP}\langle Q^\star_k\rangle$ by contracting the core of Q into the single existential variable x of $Q^\star_k$, and contracting the head variables of each atom of Q into one of the head variables y_i of $Q^\star_k$. Formally, we show a fact-wise reduction (Φ, Ψ) from $\text{MAXDP}\langle Q\rangle$ to $\text{MAXDP}\langle Q^\star_k\rangle$. (Recall that a *fact-wise reduction* is defined in Section 5.2.) We denote Q as $Q(\mathbf{u}) :- \phi_1, \ldots, \phi_k$, where $\mathbf{u} = u_1, \ldots, u_q$.

We first define the function Φ. Let Z be the core of Q, let i be an index in $\{1, \ldots, k\}$, and let f be a fact over the relation symbol of ϕ_i. If no assignment μ for $\mathsf{Var}(\phi_i)$ maps ϕ_i to f, then we set $\Phi(f) = \emptyset$. Otherwise, let μ be such an assignment. We denote by μ^Z the restriction of μ to Z (the core of Q), and by $\mu^{\mathbf{u}}$ the restriction of μ to $\mathsf{Var_h}(\phi_i)$ (the set of head variables u_j of ϕ_i). Note that $\mu^{\mathbf{u}}$ is the empty function if ϕ_i does not have head variables, and so is μ^Z if Q has an empty core. We define:

$$\Phi(f) \stackrel{\text{def}}{=} \{R_i(\mu^Z, \mu^{\mathbf{u}})\}$$

Here, we slightly abuse the notation by assuming that μ^Z and $\mu^{\mathbf{u}}$ are constants in Const. Observe that Φ is not necessarily injective, that is, $\Phi(f) = \Phi(f')$ may hold true even if $f \neq f'$.

We now define the function Ψ. Recall that $\mathbf{u} = u_1, \ldots, u_q$ is the tuple of head variables of Q. For a tuple $\mathbf{e} = (e_1, \ldots, e_q)$ in Const^q and an index $i \in \{1, \ldots, k\}$, let $\nu_i^{\mathbf{e}}$ be the assignment for $\mathsf{Var_h}(\phi_i)$ that maps each $u_j \in \mathsf{Var_h}(\phi_i)$ to e_j. We define:

$$\Psi(\mathbf{e}) \stackrel{\text{def}}{=} (\nu_1^{\mathbf{e}}, \ldots, \nu_k^{\mathbf{e}})$$

To show that (Φ, Ψ) is indeed a fact-wise reduction, we need to prove that for all instances I over $\text{schema}(Q)$ and sub-instances J' of $\Phi(I)$, the function Ψ is a bijection from $Q(\Phi_I^{-1}(J'))$ to $Q^\star_k(J')$. The proof of that is straightforward from the definition of Φ and Ψ, and hence, is omitted. Of course, the computation of Ψ and Φ for specific inputs is in polynomial time, and so, we get the following proposition.

PROPOSITION 6.2. *If Q is a sunflower CQ with k atoms and without self joins, then there is a polynomial-time fact-wise reduction from $\text{MAXDP}\langle Q\rangle$ to $\text{MAXDP}\langle Q^\star_k\rangle$.*

6.2. Approximation via Submodular Maximization

In this section, we show that MAXDP$\langle Q_k^\star\rangle$ has a polynomial-time approximation algorithm, with the constant ratio guarantee of $1 - 1/e$. We do so by formulating MAXDP$\langle Q_k^\star\rangle$ as an instance of the problem of maximizing a *monotone submodular function subject to a matroid constraint*. First, we recall the needed terminology and definitions.

Let E be a set, and let $\mathcal{I}$ be a subset of 2^E (where 2^E is the power set of E). The pair $(E, \mathcal{I})$ is a *matroid* if $\mathcal{I}$ contains the empty set, $\mathcal{I}$ is closed under taking subsets (i.e., $A \in \mathcal{I}$ and $B \subseteq A$ implies $B \in \mathcal{I}$), and $\mathcal{I}$ has the *augmentation property*: if $A, B \in \mathcal{I}$ and $|A| > |B|$, then $B \cup \{\eta\} \in \mathcal{I}$ for some $\eta \in A \setminus B$. The sets in $\mathcal{I}$ are called *independent sets*. A well known special case of a matroid is a *partition matroid*: $(E, \mathcal{I})$ is a partition matroid if there is a partition of E into disjoint sets $E_1, \ldots, E_m$, such that $\mathcal{I}$ is the set of all subsets A of E with at most one element from each E_j (that is, for all $j = 1, \ldots, m$ we have $|A \cap E_j| \leq 1$).

Let E be a set, and let $\Pi : 2^E \to \mathbb{R}_+$ be a nonnegative function defined over subsets of E. We say that Π is *submodular* if $\Pi(A \cup B) + \Pi(A \cap B) \leq \Pi(A) + \Pi(B)$ for all $A, B \subseteq E$. We say that Π is *monotone* if $\Pi(A) \leq \Pi(B)$ whenever $A \subset B$. A well known special case of a monotone submodular function is a *coverage function*: Π is a coverage function if there exists a set system $\{U_\eta : \eta \in E\}$ on some space $\mathcal{U}$ with a measure μ, such that $\Pi(A) = \mu(\cup_{\eta \in A} U_\eta)$ for all $A \subseteq E$. Since we are working with finite sets, a *measure* for our purposes is any function of the form $\mu(U) = \sum_{u \in U} w_u$ for nonnegative weights w_u (e.g., if every w_u is 1 then $\mu(U) = |U|$).

Calinescu et al. [2011] showed that for every matroid $(E, \mathcal{I})$, the problem of finding an independent set $A \in \mathcal{I}$ that maximizes $\Pi(A)$ admits a polynomial-time $(1 - 1/e)$-approximation whenever Π is a monotone submodular function. Their result improves upon the *greedy* algorithm of Nemhauser et al. [1978], which provides a $1/2$-approximation for the problem of maximizing a monotone submodular function subject to a matroid constraint. Feige [1998] proved that $1 - 1/e$ is the best polynomial-time approximation, unless P = NP; in fact, his proof implies that this lower bound holds true even if we assume that Π is a coverage function and that $(E, \mathcal{I})$ is a partition matroid. Next, we formulate MAXDP$\langle Q_k^\star\rangle$ as such a maximization problem, and thereby obtain a $(1 - 1/e)$-approximation. In the case of MAXDP$\langle Q_k^\star\rangle$, the matroid will be a partition matroid and the submodular function will be a coverage function.

First, some notation is needed. We fix the input I and $\mathbf{a} = (a_1, \ldots, a_k)$ for MAXDP$\langle Q_k^\star\rangle$. Recall that we denote $Q_k^\star$ by $Q_k^\star(y_1, \ldots, y_k) :- R_1(x, y_1), \ldots, R_k(x, y_k)$. 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 $i = 1, \ldots, k$. For $b \in \mathsf{Const}$, we denote by I^b the sub-instance of I that consists of all the facts $R_i(b, c)$ where $i \in [1, k]$ and $c \in \mathsf{Const}$. We assume that for every fact $R_i(b, c)$ of I, the constant b is an $\mathbf{a}$-joining constant, or in other words, $\mathbf{a} \in Q_k^\star(I^b)$. There is no loss of generality in making this assumption, since there is no reason to delete a fact $R_i(b', c)$ if b' is not an $\mathbf{a}$-joining constant; hence, the existence of such $R_i(b', c)$ can only improve the approximation ratio that we achieve by an algorithm. We denote by $b_1, \ldots, b_m$ the $\mathbf{a}$-joining constants. In addition, for $i \in [1, k]$ and $j \in [1, m]$ we define the instance $I_{i,j}$ as follows.

$$I_{i,j} \stackrel{\text{def}}{=} I^{b_j} \setminus \{R_i(b_j, a_i)\}$$

Observe that every $I_{i,j}$ is a solution; moreover, every union $I_{i_1,j_1} \cup \cdots \cup I_{i_n,j_n}$ is a solution, provided that $j_l \neq j_{l'}$ whenever $l \neq l'$. Conversely, every solution J is contained in $I_{i_1,1} \cup \cdots \cup I_{i_m,m}$ for some $i_1, \ldots, i_m \in [1, k]$; this is true because $\mathbf{a} \notin Q_k^\star(J)$ implies that for every b_j, the set $J \cap I^{b_j}$ excludes at least one $R_{i'}(b_j, a_{i'})$, say $R_{i_j}(b_j, a_{i_j})$, and then $J \cup I_{i_j,j}$ is also a solution. Therefore, we can restrict our search space of solutions

to those that have the form $I_{i_1,j_1} \cup \cdots \cup I_{i_n,j_n}$, where $(i_1,j_1),\ldots,(i_n,j_n) \in [1,k] \times [1,m]$ are such that $j_l \neq j_{l'}$ whenever $l \neq l'$. We can further identify a solution with a subset A of $[1,k] \times [1,m]$, such that the second components of the pairs in A are all distinct numbers; we denote by J_A the solution that corresponds to A. Within $\text{MAXDP}\langle Q_k^\star\rangle$, the goal is to find such a set A with a maximal $|Q_k^\star(J_A)|$. Finally, an easy observation is the following.

$$Q_k^\star(J_A) = \bigcup_{(i,j)\in A} Q_k^\star(I_{i,j}) \tag{11}$$

Example 6.3. Consider again the instance I_3 of Figure 1, and let $I = I_3$ and $\mathbf{a} = (\diamond,\diamond,\diamond)$ be input for $\text{MAXDP}\langle Q_3^\star\rangle$. The joining constants are $b_1 = 1$, $b_2 = 2$ and $b_3 = 3$. So here, m and k are both 3. We have the following:

— $I^1 = \{R_1(1,\diamond), R_1(1,\square), R_2(1,\diamond), R_3(1,\diamond)\}$
— $I_{1,1} = \{R_1(1,\square), R_2(1,\diamond), R_3(1,\diamond)\}$
— $I_{2,1} = \{R_1(1,\diamond), R_1(1,\square),), R_3(1,\diamond)\}$
— $I_{3,1} = \{R_1(1,\diamond), R_1(1,\square),), R_2(1,\diamond)\}$

Therefore, we get the following equalities:

$$Q_3^\star(I_{1,1}) = \{(\square,\diamond,\diamond)\} \qquad Q_3^\star(I_{2,1}) = \emptyset \qquad Q_3^\star(I_{3,1}) = \emptyset$$

Similarly, the following hold:

$$Q_3^\star(I_{1,2}) = \emptyset \qquad Q_3^\star(I_{2,2}) = \{(\diamond,\square,\diamond)\} \qquad Q_3^\star(I_{3,2}) = \emptyset$$

$$Q_3^\star(I_{1,3}) = \emptyset \qquad Q_3^\star(I_{2,3}) = \emptyset \qquad Q_3^\star(I_{3,3}) = \{(\diamond,\diamond,\square)\}$$

To solve $\text{MAXDP}\langle Q_3^\star\rangle$ for I and $\mathbf{a}$, we need to select a set $A = \{(i_1,1),(i_2,2),(i_3,3)\}$, where $i_1,i_2,i_3 \in \{1,2,3\}$, so that the set $Q_3^\star(I_{i_1,1}) \cup Q_3^\star(I_{i_2,2}) \cup Q_3^\star(I_{i_3,3})$ is of maximum cardinality; our solution J_A would then be $I_{i_1,1} \cup I_{i_2,2} \cup I_{i_3,3}$. Here the problem is straightforward, since for each $j \in \{1,2,3\}$ there is a exactly one $Q_3^\star(I_{i_j,j})$ that is nonempty, namely $Q_3^\star(I_{j,j})$, and moreover $Q_3^\star(I_{1,1})$, $Q_3^\star(I_{2,2})$ and $Q_3^\star(I_{3,3})$ are pairwise disjoint. Hence, the optimal A is $\{(1,1),(2,2),(3,3)\}$ with $J_A = I_{1,1} \cup I_{2,2} \cup I_{3,3}$, which is the instance J of Figure 4 that is shown to be optimal (and even side-effect free) in Example 3.3. Unfortunately, things are not as straightforward in the general case of an input instance I; in fact, we already know from Corollary 6.1 that finding the best A is NP-hard. We will use the machinery of Calinescu et al. [2011] to obtain an approximation. □

We can now easily define our problem as that of maximizing a coverage function subject to a partition-matroid constraint:

— E is the set $[1,k] \times [1,m]$. For $j = 1,\ldots,m$, we define $E_j = [1,k] \times \{j\}$ (hence, E is the disjoint union $E_1 \cup \cdots \cup E_m$).
— $\mathcal{I} \subseteq 2^E$ is the set of all subsets A of E with at most one element from each E_j. Hence, $(E,\mathcal{I})$ is a partition matroid.
— The space $\mathcal{U}$ is $Q_k^\star(I)$, and the measure μ over $\mathcal{U}$ is the cardinality function (i.e., $\mu(T) = |T|$ for all $T \subseteq \mathcal{U}$).
— For each $\eta = (i,j)$ in E, we define U_η to be the subset $Q_k^\star(I_{i,j})$ of $\mathcal{U}$.
— Π is defined by $\Pi(A) = |Q_k^\star(J_A)|$, or equivalently, due to (11), $\Pi(A) = \mu(\cup_{\eta\in A} U_\eta)$. Hence, Π is a coverage function.

We can now appeal to the general result of Calinescu et al. [2011], and obtain a polynomial-time $(1-1/e)$-approximation algorithm for $\text{MAXDP}\langle Q_k^\star\rangle$. More precisely,

the algorithm we get is a *randomized* $(1-1/e)$-approximation algorithm. A randomized α-approximation algorithm for MAXDP$\langle Q\rangle$ is a randomized algorithm that, given I and $\mathbf{a}$, returns a solution J that is α-optimal in expectation; that is, $\mathbb{E}[|Q(J)|]$ is at least $\alpha\cdot|Q(J')|$ for all solutions J'. This is a standard notion of randomized optimization (e.g., [Goemans and Williamson 1995; Karloff and Zwick 1997]). Note that a randomized α-approximation algorithm can be transformed (via repetitions) into a randomized algorithm that returns a β-approximation, where β is arbitrarily close to α, with an arbitrarily small error probability. Using Proposition 6.2, we get the following.

THEOREM 6.4. *For all sunflower CQs Q without self joins, there is a randomized polynomial-time $(1-1/e)$-approximation algorithm for* MAXDP$\langle Q\rangle$.

The algorithm of Calinescu et al. [2011] is quite involved. In the abridged version of this article [Kimelfeld et al. 2011], we considered the restriction of that algorithm to the special case of MAXDP$\langle Q_k^\star\rangle$. There, we gave a much simpler, self-contained description of the restricted algorithm. That restricted algorithm, deploying linear programming and randomized rounding, is significantly more practical than the application of the original algorithm of Calinescu et al. [2011] to our special case.

6.2.1. Generalization Beyond Sunflower CQs. We now show how the submodular-optimization approach, and consequently Theorem 6.4, can be extended to CQs beyond sunflower CQs. Specifically, we consider the class of CQs we call *existentially-sunflower*, and this class generalizes that of sunflower CQs. Intuitively, in an existentially-sunflower CQ the "sunflower requirement" is restricted to the existential variables, and moreover, extra atoms are allowed as long as they have no existential core variables. Formally, we denote by $\mathsf{Var}_\exists^{\bowtie}(Q)$ the set of existential join variables of the CQ Q, and we say that Q is *existentially-sunflower* if for every atom ϕ of Q, either $\mathsf{Var}_\exists^{\bowtie}(Q)\subseteq\mathsf{Var}(\phi)$ or $\mathsf{Var}_\exists^{\bowtie}(Q)\cap\mathsf{Var}(\phi)=\emptyset$. An atom ϕ with $\mathsf{Var}_\exists^{\bowtie}(Q)\cap\mathsf{Var}(\phi)=\emptyset$ is said to be *existentially disconnected*.

Example 6.5. 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-sunflower, since $\mathsf{Var}_\exists^{\bowtie}(Q)=\{x_1\}$, and every atom either contains x_1 or not. (Actually, by the same argument, every CQ with at most one existential join variable is existentially-sunflower.) Note that the atom $T(y_3,y_1,x_2)$ is existentially disconnected. □

Next, we generalize Theorem 6.4 to existentially-sunflower CQs without self joins. We denote by $Q^\exists$ the CQ that is obtained by removing all the existentially disconnected atoms; that is, $Q^\exists$ is the CQ that is induced by all the atoms ϕ of Q having $\mathsf{Var}_\exists^{\bowtie}(Q)\subseteq\mathsf{Var}(\phi)$. Observe that the arity of $Q^\exists$ can be strictly smaller than that of Q. For example, while the arity of the CQ Q of Example 6.5 is 4, that of $Q^\exists$ is 3:

$$Q^\exists(y_1,y_2,y_3) :- R(x_1,y_1,y_2), S(x_1,y_2,y_3)$$

Consider the input I and $\mathbf{a}$ for MAXDP$\langle Q\rangle$, and let J be a solution. Let $\mathbf{e}'$ be a tuple in $Q(I)$, and let $\mathbf{e}$ be a tuple in $Q^\exists(I)$. We use $\mathbf{e}\sqsubseteq\mathbf{e}'$ to denote that $\mathbf{e}$ agrees with $\mathbf{e}'$ on all the head variables of $Q^\exists$. For instance, under the CQ Q of Example 6.5, $\mathbf{e}\sqsubseteq\mathbf{e}'$ means that $\mathbf{e}$ is a prefix of $\mathbf{e}'$. We denote by $\mathbf{a}^\exists$ the tuple $\mathbf{e}$ that satisfies $\mathbf{e}\sqsubseteq\mathbf{a}$. For a tuple $\mathbf{e}\in Q^\exists(I)$, we denote by $N_I(\mathbf{e})$ the number of tuples $\mathbf{e}'\in Q(I)$ with $\mathbf{e}\sqsubseteq\mathbf{e}'$. Our algorithm is based on the following proposition.

PROPOSITION 6.6. *Let Q be an existentially-sunflower CQ without self joins, let I and $\mathbf{a}$ be input for* MAXDP$\langle Q\rangle$*, and let J be an optimal solution.*

(1) *If there is an existentially disconnected atom ϕ such that none of the tuples in $J[\phi, \mathbf{a}]$ are Q-useful in J, then $Q(J) = Q(I \setminus I[\phi, \mathbf{a}])$ (hence, the solution returned by the unidimensional algorithm is optimal).*
(2) *Otherwise, J is a solution for I and $\mathbf{a}^\exists$ in the problem $\textsc{MaxDP}\langle Q^\exists \rangle$, and moreover,*

$$|Q(J)| = \sum_{\mathbf{d} \in Q^\exists(J)} N_I(\mathbf{d})$$

The proof of Proposition 6.6 is straightforward, given the observation that in Case 2 we can assume that the optimal J contains every fact of I over the existentially disconnected atoms.

Based on Proposition 6.6, the algorithm is as follows. We first accommodate Case 1, and so, obtain a solution J' from the unidimensional algorithm. In the end, we return the best out of J' and the solution J'' we obtain in the next step, which accommodates Case 2 and is as follows.

(1) First, we apply the reduction of Section 6.1 just to $Q^\exists$ (rather than to all of Q). More precisely, we denote $Q^\exists$ as $Q^\exists(\mathbf{u}) :- \phi_1, \ldots, \phi_k$, where $\mathbf{u} = u_1, \ldots, u_q$. We slightly change the reduction: instead of defining Z as the core of $Q^\exists$ (which may not exist here), we define Z as $\mathsf{Var}^{\bowtie}_{\exists}(Q)$. Recall that in the reduction we defined $\Psi(\mathbf{e})$ to be $(\nu^{\mathbf{e}}_1, \ldots, \nu^{\mathbf{e}}_k)$, where $\mathbf{e}$ is a tuple in $Q^\exists(I)$ and $\nu^{\mathbf{e}}_i$ is the assignment for $\mathsf{Var}_{\mathsf{h}}(\phi_i)$ mapping each $u_j \in \mathsf{Var}_{\mathsf{h}}(\phi_i)$ to e_j. For such $\mathbf{d} = (\nu^{\mathbf{e}}_1, \ldots, \nu^{\mathbf{e}}_k)$, we use $N_I(\mathbf{d})$ to denote the number $N_I(\mathbf{e})$.
(2) Recall that in Section 6.2 we defined the measure μ over the space $\mathcal{U} = Q^\star_k(I)$ as the cardinality function. Next, based on Case 2 of Proposition 6.6, we replace μ with the sum of the $N_I(\mathbf{d})$ over the tuples $\mathbf{d}$ in T; that is, $\mu(T) = \sum_{\mathbf{d} \in T} N_I(\mathbf{d})$. Similarly, the goal function Π is now defined as $\Pi(A) = \sum_{\mathbf{d} \in Q^\star_k(J_A)} N_I(\mathbf{d})$.
(3) Finally, once we get a solution J for $\textsc{MaxDP}\langle Q^\star_k \rangle$ from the algorithm of Calinescu et al. [2011], we take the instance $J'' = \Phi_I^{-1}(J)$ and enhance it with all the facts over the existentially disconnected atoms.

Based on Proposition 6.6 and the approximation guarantee of the algorithm by Calinescu et al. [2011], it is easy to show that these three steps result in a $(1-1/e)$-optimal solution (in expectation). So, we get the following result.

THEOREM 6.7. *Let Q be an existentially-sunflower CQ without self joins, and let I and $\mathbf{a}$ be input for $\textsc{MaxDP}\langle Q \rangle$. At least one of the following is true.*

— *The unidimensional algorithm returns an optimal solution.*
— *The adapted application of [Calinescu et al. 2011] (described in this section) returns a solution that is $(1 - 1/e)$-optimal in expectation.*

In particular, there is a polynomial-time randomized $(1-1/e)$-approximation algorithm for $\textsc{MaxDP}\langle Q \rangle$.

6.3. Approximation in Polynomial-Time Combined Complexity

Recall that our usual notion of complexity is *data complexity*, where the query Q is fixed. Under *combined complexity*, the query Q is taken as part of the input. Each algorithm described in this article thus far terminates in polynomial time under data complexity, but in exponential worst-case time under combined complexity, as that time is exponential in the size of Q. Specifically, this is the case for the unidimensional algorithm, and the variants of the adaptation to submodular optimization discussed earlier in this section. The reason for the exponential time is the fact that these algorithms need to compute actual results $Q(J)$ for intermediate solutions (or sub-solution)

Algorithm Oblivious$(I, \mathbf{a})$

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

Fig. 7. The oblivious approximation for MAXDP$\langle Q_k^\star \rangle$

J they consider, and each $|Q(J)|$ can be already exponential in the arity of Q. In the case of the adaptation to submodular optimization, exponential time may already be entailed in the reformulation of the problem (specified in Section 6.2), as each element η of E can be as large as exponential in k. This would also be the case if we used the greedy algorithm [Nemhauser et al. 1978], giving $1/2$-approximation, as we would still need to construct the elements η of E.

In this section, we present a $1/2$-approximation algorithm for MAXDP$\langle Q_k^\star \rangle$ that, in contrast to the above algorithms, terminates in polynomial time under combined complexity. This algorithm extends to the class of sunflower CQs without self joins. Whether $1/2$-approximation in polynomial combined complexity extends to *existentially* sunflower CQs remains an open question. The main feature of the algorithm we present is that it constructs a solution while never actually evaluating $Q_k^\star$; in that sense, the algorithm is essentially *oblivious* to the quality of its intermediate solutions, and so, we call it the *oblivious* algorithm and denote it by Oblivious. Figure 7 gives the pseudo-code for the algorithm. The input consists of an instance I over $\mathrm{schema}(Q_k^\star)$, for some natural number k, and a tuple $\mathbf{a} = (a_1, \ldots, a_k)$; the output is a solution J. Note that the algorithm is not parameterized by the arity k, but rather takes k as input derived from I and $\mathbf{a}$.

The algorithm Oblivious starts with $J = I$. It then traverses over all the a-joining constants b. Recall that a constant $b \in \mathsf{Const}$ is an a-*joining constant* if $R_i(b, a_i)$ is a fact of I for all $i = 1, \ldots, k$. For each such b, the fact $R_i(b, a_i)$ is deleted from J, where i is chosen as follows: i is the minimal index j, such that in addition to $R_j(b, a_j)$, the instance 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 $\{1, \ldots, k\}$.

Example 6.8. Consider again the instance I_3 of Figure 1, and let I_3 and $\mathbf{a} = (\diamond, \diamond, \diamond)$ be input for MAXDP$\langle Q_3^\star \rangle$. The joining constants are 1, 2 and 3. When 1 is considered in Oblivious$(I_3, \mathbf{a})$, the fact $R_1(1, \diamond)$ is deleted due to the existence of $R_1(1, \square)$. Similarly, $R_2(2, \diamond)$ is deleted due to $R_2(2, \square)$ and $R_3(3, \diamond)$ is deleted due to $R_3(3, \square)$. The resulting solution is the instance J of Figure 4. As explained in Example 3.3, J is an optimal solution; but as expected, in general the oblivious algorithm is not optimal, as we further discuss later in this section. □

The following lemma states that the algorithm Oblivious is correct in the sense that it always returns a solution. The proof is immediate from the fact that for all a-joining

constants b of I, the returned sub-instance J misses $R_i(b, a_i)$ for some (actually, for exactly one) $i \in \{1, \ldots, k\}$.

LEMMA 6.9. Oblivious$(I, \mathbf{a})$ *returns a solution.*

Next, we prove that the algorithm Oblivious is indeed a $1/2$-approximation algorithm for MAXDP$\langle Q_k^\star \rangle$. Fix the input I and $\mathbf{a}$ 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 Oblivious gives a $1/2$-approximation regardless of the size of an optimal solution, since $|Q_k^\star(J')| \leq |Q_k^\star(I) \setminus \{\mathbf{a}\}|$ holds for every solution J'. As a consequence, we conclude 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. Then, since no surviving answer has two images, we will get 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; that fact $R_i(b, c)$ will preserve some answers to account for those that have been deleted.

More precisely, we define a function $\Psi : Q_k^\star(J) \to \mathsf{Const}^k$, and show that Ψ is onto $(Q_k^\star(I) \setminus \{\mathbf{a}\}) \setminus Q_k^\star(J)$. 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 satisfying $c_i \neq a_i$, and replacing that occurrence of c_i with a_i. As an example, for $k = 4$ and $\mathbf{a} = (\diamond, \diamond, \diamond, \diamond)$, the tuple $\Psi(\diamond, \heartsuit, \square, \diamond)$ is $(\diamond, \diamond, \square, \diamond)$.

LEMMA 6.10. Ψ *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 $R_j(b, d_j)$ for some index j. Let i be that index. From the algorithm's definition (Figure 8) we conclude the following.

— Since we deleted $R_i(b, d_i)$, we must have $d_i = a_i$.
— i is the minimal j for which I contains some fact $R_j(b, c)$ with $c \neq a_j$.

It thus follows that $d_j = a_j$ for all $j = 1, \ldots, i$. 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 ith element, d_i, with c. Obviously, $\mathbf{c} \in Q_k^\star(J)$, since d_i is the only constant d for which algorithm deletes $R_j(b, d)$; moreover, $\Psi(\mathbf{c}) = \mathbf{d}$. Hence, $\mathbf{c}$ is as required for the proof. □

From Lemmas 6.9 and 6.10 we get the following theorem.

THEOREM 6.11. Oblivious *is a* $1/2$*-approximation algorithm for* MAXDP$\langle Q_k^\star \rangle$.

Next, we show that (in the worst case) Oblivious does not guarantee any constant approximation better than $1/2$. Our example, which is depicted in Figure 8, is for $k = 2$ (and can be easily adapted to every $k > 2$). 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 oblivious 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' with $|Q_2^\star(J')| = 2n$. Hence, the approximation ratio is at most $1/2 + 1/(2n)$.

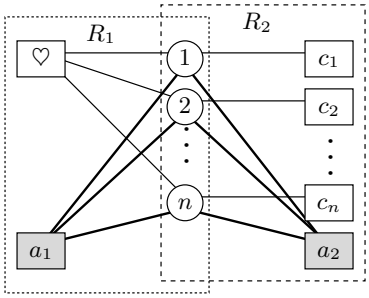

Fig. 8. An instance I where Oblivious$(I, \mathbf{a})$ provides no better than a $(1/2 + 1/(2n))$-approximation

Finally, we discuss the running time of Oblivious$(I, \mathbf{a})$. The reader can easily verify that this running time is indeed polynomial under combined complexity (i.e., even though k is part of its input). This complexity result extends to general sunflower CQs without self joins; the reason is that the polynomial time of the reduction of Section 6.1 (stated in Proposition 6.2) is also polynomial under combined complexity. We conclude the following theorem.

THEOREM 6.12. *Let Q be a sunflower CQ without self joins. There is a $1/2$-approximation algorithm for* MAXDP$\langle Q\rangle$*, where the running time is polynomial under combined complexity.*

7. ON DELETION PROPAGATION FOR CQS WITH SELF JOINS

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

We first show that in Theorem 4.4, 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 a for MAXDP$\langle Q\rangle$, such that the unidimensional algorithm returns a solution that is not optimal. The CQ Q is the following.

$$Q(y) :- R(x, y, x_1), R(x, x_2, y) \tag{12}$$

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 a be the tuple (0). The unidimensional 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 unidimensional 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 unidimensional 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 α.

THEOREM 7.1. *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 α-approximates* MAXDP$\langle Q\rangle$*, unless* P $=$ NP.

PROOF. The proof is by an approximation-preserving reduction from maximum 3-satisfiability, denoted Max3Sat. Consider an input $\varphi = \wedge_{j=1}^{m} C_j$ for Max3Sat over the variables $v_1, \ldots, v_n$, where each C_j is a clause of three literals (where each literal is positive or negative).

The reduction constructs an instance I that uses as constants the variables $v_1, \ldots, v_n$ and the clauses $C_1, \ldots, C_m$. In addition, I has 3 fresh constants $\heartsuit$, $\diamond$ and $\triangle$. The tuple

$\mathbf{a}$ is $(\heartsuit)$. The instance I contains the following two facts for each clause C_j and variable v_i that appears in C_j.

— $R(v_i, \heartsuit, C_j)$, $R(v_i, C_j, \diamond)$ if v_i appears positively in C_j.
— $R(v_i, C_j, \heartsuit)$, $R(v_i, \triangle, C_j)$ if v_i appears negatively in C_j.

Let τ be an assignment for φ, and let $\#\tau$ be the number of clauses that are satisfied by τ. Let J be a solution for I and $\mathbf{a}$. To prove the correctness of the reduction, we will show that τ can be translated into a solution J_τ, such that $\#\tau \leq |Q(J_\tau)|$, and similarly, that J can be (efficiently) translated into an assignment τ_J with $|Q(J)| \leq \#\tau_J$. This is enough, since it shows that an α-optimal solution for I and $\mathbf{a}$ can be translated into an α-optimal assignment for τ.

First, we show how J_τ is obtained from τ. We begin with $J_\tau = I$. Then, for each v_i, if $\tau(v_i) = \mathbf{true}$ then we remove every $R(v_i, C_j, \heartsuit)$ from J_τ; similarly, if $\tau(v_i) = \mathbf{false}$ then we remove every $R(v_i, \heartsuit, C_j)$ from J_τ. It is easy to see that J_τ is indeed a solution. Consider a clause C_j that is satisfied by τ. We will show that $(C_j) \in Q(J_\tau)$. If C_j is satisfied by a variable v_i that appears positively in C_j, then $\tau(v_i) = \mathbf{true}$ and both $R(v_i, \heartsuit, C_j)$ and $R(v_i, C_j, \diamond)$ remain in J_τ, which proves the claim. If C_j is satisfied by a variable v_i that appears negatively in C_j, then $\tau(v_i) = \mathbf{false}$ and both $R(v_i, C_j, \heartsuit)$, $R(v_i, \triangle, C_j)$ remain in J_τ, which again proves the claim.

Next, we show how τ_J is obtained from J. Consider a variable v_i. Since J is a solution, there are no j and k such that J contains both $R(v_i, \heartsuit, C_j)$ and $R(v_i, C_k, \heartsuit)$. If J contains some $R(v_i, \heartsuit, C_j)$, then we define $\tau_J(v_i) = \mathbf{true}$; otherwise, $\tau_J(v_i) = \mathbf{false}$. Observe that every answer $\mathbf{d} \in Q(J)$ is of the form (C_j) for some C_j. To prove that $|Q(J)| \leq \#\tau_J$, we will show that if $(C_j) \in Q(J)$ then τ_J satisfies C_j. So, suppose that $(C_j) \in Q(J)$. There are two cases.

(1) J contains $R(v_i, \heartsuit, C_j)$ and $R(v_i, C_j, \diamond)$ for some v_i.
(2) J contains $R(v_i, C_j, \heartsuit)$ and $R(v_i, \triangle, C_j)$ for some v_i.

In the first case, v_i appears positively in C_j and we define $\tau_J(v_i) = \mathbf{true}$; hence, C_j is satisfied by τ_J. In the second case, v_i appears negatively in C_j, and there cannot be any $R(v_i, \heartsuit, C_k)$ in J (or else J is not a solution); hence, $\tau_J(v_i) = \mathbf{false}$, and so C_j is again satisfied by τ_J. □

Recall from Proposition 3.4 that for every CQ without self joins, there is a $1/k$-approximation for $\text{MaxDP}\langle Q\rangle$, 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 7.2. *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* α^k*-approximates* $\text{MaxDP}\langle Q_k\rangle$.

PROOF. 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 (12), the CQ Q^2 is the following.

$$Q^2(y, y') :- R(x, y, x_1), R(x, x_2, y)$$
$$R(x', y', x'_1), R(x', x'_2, y')$$

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 $\mathsf{Const}^{\mathrm{arity}(Q)}$. 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. the CQ Q) for I and $\mathbf{a}$ if and only if J is a solution (w.r.t. the CQ Q^k) for I and $\mathbf{a}^k$.
(3) $|Q^k(I)| = |Q(I)|^k$.

We can now prove the theorem. Fix a natural number k. We choose as Q_k the CQ Q^k, where Q is the CQ of (12) and α is the one of Theorem 7.1. Clearly, Q_k satisfies $\mathrm{arity}(Q_k) = k$ and $|\mathrm{atoms}(Q_k)| = 2k$. Now, suppose that an algorithm A is an α^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)). Let J_{opt} be an optimal solution for I and $\mathbf{a}$. From Properties (1)–(3) it easily follows that J_{opt} is also optimal for I and $\mathbf{a}^k$. We obtain the following.

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

Thus, using A we get an α-approximation for MAXDP$\langle Q \rangle$, and then our theorem follows immediately from Theorem 7.1. □

8. CONCLUDING REMARKS

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 α-approximable (in polynomial time) for some constant α that is inversely proportional to the reciprocal of the size of Q, and that MAXDP$\langle Q \rangle$ is solved optimally by the unidimensional algorithm if Q has the head-domination property. We also showed a dichotomy by proving some hardness of approximation when head domination does not hold. We gave approximation algorithms for sunflower CQs. Specifically, we presented a reduction to the problem of maximizing a monotone submodular function subject to a matroid constraint, an approach that extends to the existentially sunflower CQs; moreover, we presented the oblivious algorithm that takes polynomial time under combined complexity. Finally, we showed that self joins lead to further computational hardness: first, head domination is not sufficient for tractability, and second, the realizable approximation ratio for MAXDP$\langle Q \rangle$ deteriorates exponentially with the size of Q. Various problems remain open. Among them are the following.

(1) Does a dichotomy like Theorem 4.7 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 α-approximable in polynomial time for *every* CQ Q without self joins?[4]
(3) For the class of (existentially) sunflower CQs, can we do better than Theorem 6.7? That is, is there a polynomial-time α-approximation for some $\alpha > 1 - 1/e$?

Beyond those open problems, there are many practically motivated future directions, as the work described here is limited in various dimensions. For one, we had restrictions on neither the database instances nor the allowed deletions. As previously noted [Cong et al. 2006], this does not hold true in the presence of database constraints. For example, in the case of *inclusion dependencies* [Casanova et al. 1982]

[4]Recall that the approximation ratio of Proposition 3.4 depends on Q, and that Theorem 7.2 implies that no such α exists if self joins are allowed (unless $\mathrm{P} = \mathrm{NP}$).

(e.g., foreign keys), deletion of tuples may require the deletion of other tuples, and then our algorithms may lose their approximation guarantees. On the other hand, *functional dependencies* restrict the database instances in consideration, and then our lower bounds may become invalid. A closely related restriction is to forbid the deletion of certain facts. For example, in information extraction it makes sense to forbid deletion from some clean relations like the English or country-name dictionary.[5] Finally, instead of one answer, it is conceivable that in practical scenarios one may desire to delete *multiple* answers from the view.

REFERENCES

BALAKRISHNAN, S., CHU, V., HERNÁNDEZ, M. A., HO, H., KRISHNAMURTHY, R., LIU, S., PIEPER, J., PIERCE, J. S., POPA, L., ROBSON, C., SHI, L., STANOI, I. R., TING, E. L., VAITHYANATHAN, S., AND YANG, H. 2010. Midas: integrating public financial data. In *SIGMOD Conference*. ACM, 1187–1190.

BANCILHON, F. AND SPYRATOS, N. 1981. Update semantics of relational views. *ACM Trans. Database Syst. 6,* 4, 557–575.

BEERI, C., FAGIN, R., MAIER, D., AND YANNAKAKIS, M. 1983. On the desirability of acyclic database schemes. *J. ACM 30,* 3, 479–513.

BUNEMAN, P., KHANNA, S., AND TAN, W.-C. 2002. On propagation of deletions and annotations through views. In *PODS*. ACM, 150–158.

CALINESCU, G., CHEKURI, C., PÁL, M., AND VONDRÁK, J. 2011. Maximizing a submodular set function subject to a matroid constraint. *SIAM Journal on Computing 40:6, special section on ACM STOC 2008*, 1740–1766.

CASANOVA, M. A., FAGIN, R., AND PAPADIMITRIOU, C. H. 1982. Inclusion dependencies and their interaction with functional dependencies. In *PODS*. ACM, 171–176.

CONG, G., FAN, W., AND GEERTS, F. 2006. Annotation propagation revisited for key preserving views. In *CIKM*. ACM, 632–641.

COSMADAKIS, S. S. AND PAPADIMITRIOU, C. H. 1984. Updates of relational views. *J. ACM 31,* 4, 742–760.

CUI, Y. AND WIDOM, J. 2001. Run-time translation of view tuple deletions using data lineage. Tech. rep., Stanford University. `http://dbpubs.stanford.edu:8090/pub/2001-24`.

DAYAL, U. AND BERNSTEIN, P. A. 1982. On the correct translation of update operations on relational views. *ACM Trans. Database Syst. 7,* 3, 381–416.

FAGIN, R. 1983. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM 30,* 3, 514–550.

FAGIN, R. 1999. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci. 58,* 1, 83–99.

FAGIN, R., LOTEM, A., AND NAOR, M. 2003. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci. 66,* 4, 614–656.

FAGIN, R., ULLMAN, J. D., AND VARDI, M. Y. 1983. On the semantics of updates in databases. In *PODS*. ACM, 352–365.

FEIGE, U. 1998. A threshold of ln n for approximating set cover. *J. ACM 45,* 4, 634–652.

GOEMANS, M. X. AND WILLIAMSON, D. P. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM 42,* 6, 1115–1145.

GOLD, E. M. 1978. Complexity of automaton identification from given data. *Information and Control 37,* 3, 302–320.

GURUSWAMI, V. 2003. Inapproximability results for set splitting and satisfiability problems with no mixed clauses. *Algorithmica 38,* 3, 451–469.

KARLOFF, H. J. AND ZWICK, U. 1997. A 7/8-approximation algorithm for max 3sat? In *FOCS*. 406–415.

KELLER, A. M. 1985. Algorithms for translating view updates to database updates for views involving selections, projections, and joins. In *PODS*. ACM, 154–163.

KIMELFELD, B., VONDRÁK, J., AND WILLIAMS, R. 2011. Maximizing conjunctive views in deletion propagation. In *PODS*. ACM, 187–198.

LECHTENBÖRGER, J. AND VOSSEN, G. 2003. On the computation of relational view complements. *ACM Trans. Database Syst. 28,* 2, 175–208.

[5]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 [Meliou et al. 2011].

LIU, B., CHITICARIU, L., CHU, V., JAGADISH, H. V., AND REISS, F. 2010. Automatic rule refinement for information extraction. *PVLDB 3,* 1, 588–597.

MELIOU, A., GATTERBAUER, W., MOORE, K. F., AND SUCIU, D. 2011. The complexity of causality and responsibility for query answers and non-answers. *PVLDB 4,* 1, 34–45.

NEMHAUSER, G. L., WOLSEY, L. A., AND FISHER, M. L. 1978. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming 14*, 265–294.

RILOFF, E. AND JONES, R. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*. 474–479.

VARDI, M. Y. 1982. The complexity of relational query languages (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*. ACM, 137–146.