On Low Treewidth Approximations of Conjunctive Queries

Pablo Barceló[1], Leonid Libkin[2], and Miguel Romero[1]

[1] Department of Computer Science, University of Chile
[2] School of Informatics, University of Edinburgh

Abstract. We recently initiated the study of approximations of conjunctive queries within classes that admit tractable query evaluation (with respect to combined complexity). Those include classes of acyclic, bounded treewidth, or bounded hypertreewidth queries. Such approximations are always guaranteed to exist. However, while for acyclic and bounded hypertreewidth queries we have shown a number of examples of interesting approximations, for queries of bounded treewidth the study had been restricted to queries over graphs, where such approximations usually trivialize. In this note we show that for relations of arity greater than two, the notion of low treewidth approximations is a rich one, as many queries possess them. In fact we look at approximations of queries of maximum possible treewidth by queries of minimum possible treewidth (i.e., one), and show that even in this case the structure of approximations remain rather rich as long as input relations are not binary.

1 Introduction

The concept of approximating queries by those that are easier to execute is fairly standard in databases; see, for example, [5, 6, 8, 12, 15]. Typically one analyzes the structure of both the database and the query in order to find a good, or at least reasonable approximation of the answer. Approximate techniques are relevant for problems of high complexity, and even for problems whose complexity is viewed as acceptable for regular-size databases, when those queries are asked against very large databases.

We have recently initiated a study of approximations of conjunctive queries from the static analysis point of view [2]. The reason we concentrated on conjunctive, or select-project-join queries is twofold. First, they play a special role in database applications. Second, we have a very good understanding of their complexity. We know which classes of conjunctive queries are easy to evaluate; these include acyclic queries, queries of bounded treewidth, and queries of bounded hypertreewidth, see [4, 9–11, 13, 17]. Studying approximations from the point of view of static analysis means that we find approximations independently of the input database: for a query Q, we want to find another query Q' that will be much faster than Q, and whose output would be close to the output of Q on *all* databases.

We have shown in [2] that approximations always exist for classes mentioned above; furthermore, sizes of approximations are at most polynomial in the size of the original query (and often they do not exceed the size of the original query), and they can be found in at most single exponential time.

To see why these properties are desirable, consider, for instance, the complexity of checking whether a tuple $\bar{a}$ belongs to the output of a conjunctive query Q on a database D. This is of the order $|D|^{O(|Q|)}$, where $|\cdot|$ measures the size (of a database or a query) [1, 16]. In fact, the combined complexity of conjunctive query evaluation is well known to be NP-complete even for Boolean conjunctive queries [3]. That is, the problem of checking, for a given database D and a Boolean conjunctive query Q, whether $D \models Q$ (i.e., Q is true in D), is NP-complete.

The data complexity of conjunctive query evaluation is of course very low (in AC^0), but if we are concerned with evaluating queries over large data sets, having $O(|Q|)$ as the exponent may be too high. With conjunctive queries from the good classes mentioned above, the $O(|Q|)$ exponent is replaced by a fixed one. For instance, for acyclic conjunctive queries, evaluation can be done in time $O(|D|\cdot|Q|)$, see [17]. Thus, assuming that we can find an approximation relatively fast (for instance, in time $2^{O(|Q|\log|Q|)}$) and its size is roughly bounded by the size of Q, we may want to check if the approximation gives us the answer first. The complexity of doing so is

$$2^{O(|Q|\log|Q|)} \;+\; O(|D|\cdot|Q|)$$

which is likely to be much less than $|D|^{O(|Q|)}$ on very large databases D.

These were the motivations of [2], which showed how to construct approximations in the above tractable classes (indeed, with the complexity of finding approximations as indicated above).

These classes, however, are defined using different parameters: either *graphs*, or *hypergraphs* of queries. Acyclicity, for instance, is a hypergraph-based notion, as is the hypertree width. Treewidth, on the other hand, is a graph-based notion. In general, these approaches are incompatible [7, 10, 11, 13]. For instance, for the conjunctive query $Q() \text{:-} R(x, y, z)$, its hypergraph will contain a single hyperedge $\{x, y, z\}$ and will be acyclic; its graph, on the other hand, will be K_3 (complete graph of 3 nodes), and will be of treewidth 2. Considering relations of higher arity, we can find queries of arbitrary high treewidth that will be acyclic.

For hypergraph-based classes, we have shown that nontrivial acyclic approximations exist, and have given a number of examples, in addition to the general existence theorems in [2].

For graph-based classes, we have only stated general existence results (which do guarantee existence and appropriate complexity bounds), and then studied approximations for queries that themselves operate on graphs. For such queries, interesting approximations of low treewidth can only be found under strong graph-theoretic restrictions, and many queries only posses trivial approximations (this will be explained later in detail).

Our goal now is to show that this phenomenon is restricted only to graph queries, and beyond those we do have a rather rich structure of approximations for graph-based notions. This is what we do in this note.

2 Notations

Conjunctive queries Recall that the term *conjunctive queries* refers to the $\exists, \wedge$-fragment of first-order logic. Such queries can be written in the form

$$Q(\bar{x}) \;=\; \exists \bar{y}\; R_{i_1}(\bar{u}_{i_1}) \wedge \ldots \wedge R_{i_k}(\bar{u}_{i_k})$$

where each R_{i_j} is a relation symbol, and each $\bar{u}_{i_j}$ is a tuple of variables from $\bar{x}, \bar{y}$ of the same arity as R_{i_j}. These are typically written in the rule-based notation:

$$Q(\bar{x}) \text{ :- } R_{i_1}(\bar{u}_{i_1}), \ldots, R_{i_k}(\bar{u}_{i_k}).$$

If we have a Boolean (yes/no) query, i.e., a sentence, or a query without free variables, we indicate this by writing $Q()$ in the rule-based notation.

Given a database D, the answer $Q(D)$ to Q is $\{\bar{a} \mid D \models Q(\bar{a})\}$. If Q is a Boolean query (a sentence), the answer *true* is, as usual, modeled by the set containing the empty tuple, and the answer *false* by the empty set.

We refer to the number of subgoals in the body of a conjunctive query as *the number of atoms*. A query with k atoms requires $k-1$ joins to be evaluated, so this is an important parameter of the complexity of a conjunctive query.

Here we primarily deal with Boolean conjunctive query over a vocabulary consisting of an m-ary relation R, for some $m > 1$. For instance, if $m = 2$, the query

$$Q_{tr}() \text{:-} R(x,y), R(y,z), R(z,x)$$

checks if the graph contains a triangle.

Query containment and tableaux A conjunctive query Q is *contained* in a conjunctive query Q', written as $Q \subseteq Q'$, if $Q(D) \subseteq Q'(D)$ for every database D. If Q and Q' are Boolean conjunctive queries, then $Q \subseteq Q'$ means that the implication $Q \rightarrow Q'$ is valid over all finite databases.

With each Boolean conjunctive query Q over m-ary relation R

$$Q \text{ :- } R(\bar{x}_1), \ldots, R(\bar{x}_m) \tag{1}$$

we associate its *tableau* T_Q, which is a relation R interpreted as the set of tuples $\{\bar{x}_1, \ldots, \bar{x}_m\}$.

Many key properties of conjunctive queries can be stated in terms of homomorphisms of tableaux. For example, $D \models Q$ iff there is a homomorphism from T_Q to D, and $Q \subseteq Q'$ iff there is a homomorphism from $T_{Q'}$ to T_Q [3]. These classical results, of course, extend to queries with free variables, and over arbitrary vocabularies.

Graphs and homomorphisms We shall also need some definitions about graphs and digraphs. Both are defined as pairs $G = \langle V, E\rangle$, where V is a set of nodes (vertices) and E is a set of edges. For graphs, an edge is a set $\{u, v\}$, where $u, v \in V$; for digraphs, an edge is a pair (u, v), i.e., it has an orientation from u to v. If $u = v$, we have a (undirected or directed) loop.

We denote by K_m the complete graph on m vertices: $K_m = \langle\{u_1, \ldots, u_m\}, \{\{u_i, u_j\} \mid i \neq j,\ i, j \leq m\}\rangle$. Furthermore, we let $\mathbf{K}_m$ stand for the finite set of graphs on vertices $u_1, \ldots, u_m$ which have all the edges of K_m and perhaps some loops as well.

Given two graphs (directed or undirected) $G_1 = \langle V_1, E_1\rangle$ and $G_2 = \langle V_2, E_2\rangle$, a *homomorphism* between them is a map $h : V_1 \to V_2$ such that $h(e)$ is in E_2 for every edge $e \in E_1$. Of course by $h(e)$ we mean $\{h(u), h(v)\}$ if $e = \{u, v\}$ and $(h(u), h(v))$ if $e = (u, v)$. The image of h is the (di)graph $\mathrm{Im}(h) = \langle h(V_1), \{h(e) \mid e \in E_1\}\rangle$. If there is a homomorphism h from G_1 to G_2, we write $G_1 \to G_2$ or $G_1 \xrightarrow{h} G_2$.

A graph G is a *core* if there is no homomorphism $G \to G'$ into a proper subgraph G' of G. A subgraph G' of G is a *core of* G if G' is a core and $G \to G'$. It is well known that all cores of a graph are isomorphic and hence we can speak of the core of a graph, denoted by $\mathrm{core}(G)$. We say that two graphs G and G' are *homomorphically equivalent* if both $G \to G'$ and $G' \to G$ hold. Homomorphically equivalent graphs have the same core, i.e., $\mathrm{core}(G)$ and $\mathrm{core}(G')$ are isomorphic. These notions straightforwardly extend to arbitrary structures (and thus can be applied to tableaux of conjunctive queries).

If two tableaux T_Q and $T_{Q'}$ are homomorphically equivalent, then Q and Q' are equivalent queries. Furthermore, $\mathrm{core}(T_Q)$ serves as the tableau of the minimized version of Q: the query equivalent to Q that has the minimum number of atoms. We refer to queries whose tableaux are cores as *minimized*.

3 Classes of conjunctive queries

Tractable classes of conjunctive queries can be defined in terms of both graphs and hypergraphs [4, 7, 10, 11, 13, 17]. Here we look at graph-based classes.

With each conjunctive query Q, we associate its graph $G(Q) = \langle V, E\rangle$ as follows:

- V is the set of variables used in Q;
- there is an edge $\{x, y\}$ if x and y are used in two distinct positions in the same atom of Q.

For instance, for the query Q_{tr} used above, we have $G(Q_{tr}) = K_3$.

For graph-based classes of queries, tractability results rely on the notion of *treewidth*. A *tree decomposition* of a graph $G = \langle V, E\rangle$ is a tree T together with a map $f : T \to 2^V$ that associates a set of vertices in V with each node of T such that

1. each edge from E is contained in one of the sets $f(u)$ for $u \in T$; and

2. for every $v \in V$, the set $\{u \in T \mid v \in f(u)\}$ is a connected subset of T.

The *width* of a decomposition is $\max_{u \in T} |f(u)| - 1$, and the *treewidth* of G is the minimum width of its tree decompositions. If G is a tree (or a forest) to start with, then its treewidth is 1. We refer to the classes of graphs of treewidth at most k as $\mathsf{TW}(k)$.

If $\mathcal{C}$ is a class of graphs, then Q is a *$\mathcal{C}$-query* if $G(Q) \in \mathcal{C}$. We shall be interested in $\mathsf{TW}(k)$-queries for various k.

It was shown in [11] that the notion of treewidth essentially describes tractability for graph-based classes of queries. Modulo some complexity-theoretic assumptions, if the class of $\mathcal{C}$-queries has polynomial combined complexity, then $\mathcal{C} \subseteq \mathsf{TW}(k)$ for some fixed k.

The degree of the polynomial depends on the treewidth, so ideally one wants to find approximations of low treewidth (in particular, treewidth one).

4 The notion of approximation

We now present the notion of approximations from [2]. The idea is as follows: we are given a query Q, and we want to approximate it within the class of $\mathcal{C}$-queries. For that, we define an ordering $\sqsubset_Q$ on queries in $\mathcal{C}$: the meaning of $Q_1 \sqsubset_Q Q_2$ is that "Q_2 approximates Q better than Q_1 does", i.e., Q_2 agrees with Q more often than Q_1. Then we look for maximal elements with respect to $\sqsubset_Q$ as good approximations of Q.

Formally, we say that two Boolean conjunctive queries Q and Q' *agree* on a database D if $D \models Q \leftrightarrow Q'$. Then, for conjunctive queries Q, Q_1, and Q_2,

$$Q_1 \sqsubseteq_Q Q_2 \stackrel{\text{def}}{=} \forall D \, \big(Q_1 \text{ and } Q \text{ agree on } D \Rightarrow Q_2 \text{ and } Q \text{ agree on } D\big)$$

That is, Q_2 approximates Q at least as well as Q_1 does. Then Q_2 approximates Q better than Q_1 does if $Q_1 \sqsubset_Q Q_2$, i.e., $Q_1 \sqsubseteq_Q Q_2$ and $Q_2 \not\sqsubseteq_Q Q_1$.

Definition 1. (Approximations) *A $\mathcal{C}$-query Q' is a* $\mathcal{C}$-approximation *of a conjunctive query Q if $Q' \subseteq Q$ and there is no $\mathcal{C}$-query $Q'' \subseteq Q$ such that $Q' \sqsubset_Q Q''$.*

In other words, Q' is an approximation of Q if it is guaranteed to return correct results and no other query approximates Q better than Q'. We require the approximating query to be contained in Q; that is, we only want to return correct answers which is a standard approach in databases [14].

Of course the definitions straightforwardly extend to non-Boolean conjunctive queries: one only has to replaces databases D with pairs $(D, \bar{a})$, where $\bar{a}$ ranges over tuples of elements of the same arity as Q.

Remark Note that there could be different ways of defining approximations; in particular, one can follow either the qualitative approach, which defines approximations as queries that cannot be improved in terms of how close they come to

the query they approximate, or the quantitative approach, which would define a measure of difference between two queries and require the approximating query to be close in that measure to the query it approximates. Here we follow [2] which initiated the study of the qualitative approach, but of course the quantitative approach needs to be studied too.

It was shown in [2] that the use of the ordering $\sqsubseteq_Q$ can be replaced by a containment test: A $\mathcal{C}$-query $Q' \subseteq Q$ is a $\mathcal{C}$-approximation of Q if and only if there is no $\mathcal{C}$-query Q'' such that $Q' \subset Q'' \subseteq Q$.

To state the existence result, we make a technical assumption that a single-element loop, i.e., the graph Loop with a single node x and a loop (x, x) on it, is in $\mathcal{C}$. This is not a restriction for us: the treewidth of Loop is 1, and hence it belongs to $\mathsf{TW}(k)$ for all k.

A key result on the existence of approximations from [2] stated the following.

Fact 2 *Let $\mathcal{C}$ be a class of graphs closed under taking subgraphs. Then every conjunctive query Q has a $\mathcal{C}$-approximation. Furthermore:*

1. *the number of minimized queries which are $\mathcal{C}$-approximations of Q is finite (in fact at most exponential in the size of Q);*
2. *each such minimized $\mathcal{C}$-approximation has at most as many atoms as Q;*
3. *an approximation can be found in time single-exponential in the size of Q.*

Some queries have *trivial* approximations. Let Q^{triv} be the query whose tableau is Loop, the singleton loop graph. That is, $Q() \text{:–} R(x, x, \ldots, x)$. Note that since Loop has treewidth 1, the query Q^{triv} is $\mathsf{TW}(k)$-query for every $k \geq 1$. Moreover, $Q^{\text{triv}} \subseteq Q$ for every conjunctive query Q (via the constant homomorphism). In a way, this is the least interesting possible approximation.

We say that an approximation is *trivial* if it is equivalent to Q^{triv}; otherwise it is nontrivial.

The following was shown in [2].

Fact 3 *A conjunctive query Q over graphs has a nontrivial* $\mathsf{TW}(1)$*-approximation iff $G(Q)$ is bipartite.*

Recall that a graph G is bipartite iff there is a homomorphism $G \to K_2$ (or, equivalently, iff it is 2-colorable) [18].

Thus, many queries over graphs lack interesting efficient approximations (say, $\mathsf{TW}(1)$-approximations): for instance the query Q_{tr} seen earlier does not have one, since $G(Q_{tr}) = K_3$ is not bipartite.

What we shall see now is that this behavior of queries changes radically when we go beyond queries over graphs.

5 Strong treewidth approximations

What could be the strongest possible approximations in terms of reducing treewidth of (the graph of) a query? Recall that we work with Boolean conjunctive queries over an m-ary relation R. An additional proviso for this section

is that we assume queries to be connected (i.e., $G(Q)$ is connected). This is not a restriction at all: each conjunctive query is a conjunction of connected ones.

For a query Q, let $\mathrm{var}(Q)$ be the number of variables in it. Then the treewidth of $G(Q)$ is at most $\mathrm{var}(Q) - 1$. Hence, the strongest possible approximations in terms of reducing treewidth are those that go from treewidth $\mathrm{var}(Q) - 1$ to the smallest possible treewidth, i.e., 1.

Definition 4. *A conjunctive query Q' is a* strong treewidth approximation *of a conjunctive query Q if Q' is a $\mathsf{TW}(1)$-approximation, and the treewidth of $G(Q)$ is maximal, i.e.,* $\mathrm{var}(Q) - 1$.

Remarks

- Note that not every query has a strong treewidth approximation. In fact, a query Q may have such an approximation only if its treewidth equals $\mathrm{var}(Q) - 1$, i.e., is the maximum possible. We define this notion in order to study the best approximations in terms of treewidth reduction.
- In addition, results on fixed-parameter tractability for graph-based classes of conjunctive queries say that the treewidth of the graph determines the exponent in the complexity of query evaluation. So in a sense, strong approximations show how to approximate the worst-complexity queries by the best-complexity queries.
- The goal of our study here is *not* to come up with a practically relevant class of queries that can be efficiently approximated. Such issues have been discussed in [2], which nonetheless had a gap, as it did not consider approximations within graph-based classes except on queries over graphs. Our goal is to provide evidence that approximations within graph-based classes make perfect sense for queries over relations of higher arities; then complexity and other analyses of [2] apply.

Note that in the case of graphs ($m = 2$), the concept of strong treewidth approximations trivializes. This is due to the fact that if treewidth of $G(Q)$ equals $n - 1$, for $n = \mathrm{var}(Q)$, then $G(Q)$ is in $\mathbf{K}_n$ (i.e., the graph of Q is K_n, perhaps with some loops).

Proposition 1. *Assume that Q is a query of graphs that is not a $\mathsf{TW}(1)$-query. If Q' is its strong treewidth approximation, then Q' is trivial.*

Proof sketch. Indeed, K_n for $n > 2$ are not bipartite and thus can only have trivial $\mathsf{TW}(1)$-approximations. □

So from now, we assume that m, the arity of R, is at least 3.

The first observation is that strong treewidth approximations, if they exist, do not have too many variables.

Proposition 2. *Assume $m > 2$, and let Q' be a strong treewidth approximation of Q. Then $G(Q')$ has at most two nodes.*

Proof sketch. Indeed, if $\mathrm{var}(Q) = n$, then $G(Q) \in \mathbf{K}_n$, and if a homomorphic image of T_Q contains at least 3 nodes, then it will have a triangle, and thus will have treewidth at least 2. □

Thus, $G(Q')$ in the above proposition is either K_2, or a K_2 with one or two loops. While these graphs are quite simple, there are still quite a few queries that generate them.

Potential approximations We call a query Q' a *potential strong treewidth approximation* if $G(Q')$ has at most two nodes. In particular, every strong treewidth approximation is a potential strong treewidth approximation. Some of those potential strong treewidth approximations are equivalent to the trivial query $Q^{\mathrm{triv}}() \text{:–} R(x, x, \ldots, x)$. Of course we are interested in others. Note that there are quite a few of them.

Proposition 3. *There are at least* $2^{2^m-3} - 1$ *nonequivalent, nontrivial, and minimized potential strong treewidth approximations.*

Proof sketch. Assume that the variables are x and y, and consider $2^m - 2$ tuples of length m of those except the two trivial ones (m repetitions of the same variable). We then look at sets of those and see that each such set defines a core as a structure of relation R. Finally we divide the number by two due to the symmetry between x and y. □

So, there are many potential strong treewidth approximations that are nontrivial. But can the potential be realized? The answer is yes.

Theorem 1. *Let Q' be a potential strong treewidth approximation. Assume that Q' is nontrivial. Then, for every $n > m$, there is a conjunctive query Q with $\mathrm{var}(Q) = n$ such that Q' is a strong treewidth approximation of Q.*

Moreover, if the query Q' has k atoms, then Q can be chosen to have at most

$$k + \frac{n(n-1)}{2} - 1$$

atoms.

Proof sketch. Observe that in every atom in Q' one variable occurs at least twice. Assume first that there is an atom in which some variable occurs exactly twice, say, an atom $R(x, \ldots, x, y, y)$. Then in Q we put atoms $R(x_1, \ldots, x_1, x_i, x_j)$ for all $2 \leq i \leq j \leq n$, assuming Q has variables $x_1, \ldots, x_n$. For every other atom $R(x, \ldots, x, y, \ldots, y)$ with r occurrences of y we put in Q the atom $R(x_1, \ldots, x_1, x_2, \ldots, x_{r+1})$ (of course variables can occur in an arbitrary order; we simply replace all the occurrences of x with x_1 and r occurrences of y with $x_2, \ldots, x_{r+1}$, in the same order in which the y's occur in the atom). The construction ensures $G(Q) \in \mathbf{K}_n$, and it is easy to verify that Q has at most $k + n - 2 + \frac{(n-1)(n-2)}{2} = k + \frac{n(n-1)}{2} - 1$ atoms, as only one atom in Q' generates multiple atoms in Q. The mapping sending x_1 to x and every x_i with $i > 1$

to y is a homomorphism showing $Q' \subseteq Q$. If there were $\mathsf{TW}(1)$-approximation Q'' of Q with $Q' \subset Q'' \subset Q$, we would have $G(Q'') \in \mathbf{K}_2$, so a homomorphism of the tableau of Q'' into the tableau of Q' can only be the identity, or swapping the roles of variables x and y. Using this one easily verifies that Q' is an approximation.

If we do not have an atom with exactly two occurrences of a variable, then pick an atom with a minimum number p of repetitions of a variable, say $R(x, \ldots, x, y, \ldots, y)$, where y occurs p times. Then we replace it by putting in Q atoms

$$R(x_1, \ldots, x_1,\ x_2, \ldots, x_{p-1},\ x_i, x_j)$$

with $p \leq i < j \leq n$ (where x_1s correspond to the positions of x). In addition, we put in Q atoms

$$R(x_1, \ldots, x_1,\ x_i, \ldots, x_i,\ x_i, x_i)$$

whenever $2 \leq i \leq n$. The proof then is the same (only the number of atoms in Q gets smaller). □

High arity approximations We saw that strong treewidth approximations are ubiquitous. For graph queries, it is known that $\mathsf{TW}(1)$-approximations strictly decrease the number of atoms [2]. It turns out that going beyond graphs, we can have many strong treewidth approximations that preserve the number of atoms (and thus the number of joins). The result showing this uses relations of arbitrarily high arity.

Theorem 2. *For every $k \geq 3$, one can find a relation symbol R of arity $m > 2$, and two conjunctive queries Q and Q' over R such that:*

- *both Q and Q' are minimized;*
- *Q' is a strong treewidth approximation of Q; and*
- *Q and Q' have k atoms each.*

Proof sketch. We take m, the arity of R, to be equal to k. In Q, the first three atoms are

$$\begin{array}{l} R(\ x_1, x_2,\quad x_3,\quad x_4, \ldots,\ x_k) \\ R(\ x_2, x_1,\quad x_{k+1}, x_4, \ldots,\ x_k) \\ R(\ x_3, x_{k+1}, x_1,\quad x_4, \ldots, x_k). \end{array}$$

The next $k-3$ atoms are of the form

$$R(x_j, x_j, \ldots, x_j, x_1, x_j, \ldots, x_j),$$

where x_1 appears in the jth position; here $4 \leq j \leq k$.

In Q', we have k atoms of the form

$$R(x, y, \ldots, y),\ R(y, x, y, \ldots, y),\ \ldots,\ R(y, \ldots, y, x),$$

i.e., x appears once, every time in a different position. It is straightforward to verify all three conditions of the theorem. □

Low arity approximations A slight drawback of the previous result is that it requires relations of high arity. But we can show that already for *ternary* relations, the behavior of strong treewidth approximations is drastically different from the graph case.

Recall the query $Q_{tr}() \text{:–} R(x,y), R(y,z), R(z,x)$ used earlier; it states that a graph contains a triangle. Its graph is not bipartite, since $G(Q) = K_3$, and hence it only has trivial $\mathsf{TW}(1)$-approximation.

We now look at ternary relations R. We call an instance R of a ternary relation an *almost-triangle* if there is an element that belongs to every triple of R, and when it is removed from every triple, the resulting pairs form a triangle. For instance, $(4,1,2),(4,2,3),(4,3,1)$ is an almost-triangle: when we remove 4 for each of the triples, we end up with the pairs $(1,2),(2,3),(3,1)$ which form a triangle.

Theorem 3. *Let m, the arity of R, be 3. There is a minimized conjunctive query Q with 4 variables, of maximum treewidth 3, such that:*

- *the tableau T_Q of Q is an almost-triangle; and*
- *Q has a strong treewidth approximation Q' with the same number of atoms as Q.*

Thus, indeed, the behavior of treewidth approximations is already drastically different for the case of ternary relations, compared to graphs.

Proof sketch. We define Q over variables x_1, x_2, x_3, x_4 as follows:

$$Q() \text{ :– } R(x_1,x_2,x_3),\ R(x_2,x_1,x_4),\ R(x_4,x_3,x_1).$$

Its tableau is an almost triangle (just remove x_1). Then $G(Q) = K_4$ and thus it has treewidth 3. It is also minimized.

We then look at

$$Q'() \text{ :– } R(x,y,y),\ R(y,x,y),\ R(y,y,x).$$

It is routine to verify that Q' is a strong treewidth approximation of Q satisfying conditions of the theorem. □

Acknowledgments Partial support provided by Fondecyt grant 1110171, EPSRC grants G049165 and F028288, and FET-Open Project FoX, grant agreement 233599. Part of this work was done when the second author visited Santiago.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. P. Barceló, L. Libkin, M. Romero. Efficient approximations of conjunctive queries. In *PODS'12*, to appear.

3. A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *ACM Symp. on Theory of Computing*, 1977, pages 77–90.
4. C. Chekuri, A. Rajaraman. Conjunctive query containment revisited. *Theor. Comput. Sci.* 239(2): 211-229 (2000).
5. W. Fan, J. Li, S. Ma, N. Tang, Y. Wu. Graph pattern matching: from intractable to polynomial time. *PVLDB* 3(1): 264-275 (2010).
6. R. Fink, D. Olteanu. On the optimal approximation of queries using tractable propositional languages. In *ICDT 2011*, pages 174–185.
7. J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *J. ACM*, 49 (2002), 716–752.
8. M. Garofalakis and P. Gibbons. Approximate query processing: taming the terabytes. In *VLDB'01*.
9. G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. *J. ACM*, 48 (2001), 431–498.
10. G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *JCSS*, 64 (2002), 579–627.
11. M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *ACM Symp. on Theory of Computing*, 2001, pages 657–666.
12. Y. Ioannidis. Approximations in database systems. In *ICDT'03*, pages 16–30.
13. Ph. Kolaitis and M. Vardi. Conjunctive-query containment and constraint satisfaction. *JCSS* 61(2):302-332 (2000).
14. M. Lenzerini. Data integration: a theoretical perspective. In *PODS'02*, pages 233–246.
15. Q. Liu. Approximate query processing. *Encyclopedia of Database Systems*, 2009, pages 113–119.
16. M. Vardi. On the complexity of bounded-variable queries. In *PODS'95*, pages 266–276.
17. M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. Conf. on Very Large Databases*, 1981, pages 82–94.
18. D. West. *Introduction to Graph Theory*. Prentice Hall, 2001.