Towards Tractability of the Diversity of Query Answers: Ultrametrics to the Rescue

MARCELO ARENAS, Pontificia Universidad Católica de Chile, Chile and RelationalAI, USA
TIMO CAMILLO MERKL, TU Wien, Austria
REINHARD PICHLER, TU Wien, Austria
CRISTIAN RIVEROS, Pontificia Universidad Católica de Chile, Chile

The set of answers to a query may be very large, potentially overwhelming users when presented with the entire set. In such cases, presenting only a small subset of the answers to the user may be preferable. A natural requirement for this subset is that it should be as diverse as possible to reflect the variety of the entire population. To achieve this, the diversity of a subset is measured using a metric that determines how different two solutions are and a diversity function that extends this metric from pairs to sets. In the past, several studies have shown that finding a diverse subset from an explicitly given set is intractable even for simple metrics (like Hamming distance) and simple diversity functions (like summing all pairwise distances). This complexity barrier becomes even more challenging when trying to output a diverse subset from a set that is only implicitly given (such as the query answers for a given query and a database). Until now, tractable cases have been found only for restricted problems and particular diversity functions.

To overcome these limitations, we focus in this work on the notion of ultrametrics, which have been widely studied and used in many applications. Starting from any ultrametric d and a diversity function δ extending d, we provide sufficient conditions over δ for having polynomial-time algorithms to construct diverse answers. To the best of our knowledge, these conditions are satisfied by all the diversity functions considered in the literature. Moreover, we complement these results with lower bounds that show specific cases when these conditions are not satisfied and finding diverse subsets becomes intractable. We conclude by applying these results to the evaluation of conjunctive queries, demonstrating efficient algorithms for finding a diverse subset of solutions for acyclic conjunctive queries when the attribute order is used to measure diversity.

CCS Concepts: • **Theory of computation** → **Database theory**.

Additional Key Words and Phrases: Query evaluation, diversity, conjunctive queries.

ACM Reference Format:
Marcelo Arenas, Timo Camillo Merkl, Reinhard Pichler, and Cristian Riveros. 2024. Towards Tractability of the Diversity of Query Answers: Ultrametrics to the Rescue. *Proc. ACM Manag. Data* 2, 5 (PODS), Article 215 (November 2024), 26 pages. https://doi.org/10.1145/3695833

1 Introduction

The set of answers to a query may be very large, potentially overwhelming users when presented with the entire set. In such cases, presenting only a small subset of the answers to the user may be preferable. Ideally, the selected answers should give the user a good overview of the variety

Authors' Contact Information: Marcelo Arenas, Pontificia Universidad Católica de Chile, Santiago, Chile and RelationalAI, Berkeley, USA, marenas@uc.cl; Timo Camillo Merkl, TU Wien, Vienna, Austria, timo.merkl@tuwien.ac.at; Reinhard Pichler, TU Wien, Vienna, Austria, reinhard.pichler@tuwien.ac.at; Cristian Riveros, Pontificia Universidad Católica de Chile, Santiago, Chile, cristian.riveros@uc.cl.

ACM 2836-6573/2024/11-ART215
https://doi.org/10.1145/3695833

present in the complete set of answers. As was argued in [34], determining such a small subset by sampling will, in general, not constitute a satisfactory solution to this problem, since it would most probably miss interesting but rarely occurring answers. Instead, the goal should be to present to the user a *diverse* subset of the answer space to reflect its variety.

This raises the question of how to define diversity among query results. The natural way of defining the diversity of a set of elements (see, e.g., [17] for a whole framework on dealing with diversity) is to first define the diversity of 2 elements by a metric (the "distance" function) and then to appropriately extend it to arbitrary (finite) sets. Both, for the metric and for the generalization to arbitrary sets, many choices exist and, as was mentioned in [36], it ultimately depends on the application context which distance and diversity function is best suited.

We restrict ourselves here to the relational model. Hence, the answer to a query is a set of tuples and we are interested in outputting a subset with a given size k so as to maximize the diversity. A natural and simple choice for the distance between two tuples is the Hamming distance (i.e., the number of positions in which the two tuples differ), which was used e.g., in the analysis of the diversity of query answers in [22]. A more nuanced point of view was taken in [33, 34], where an ordering of the attributes is assumed and tuples are considered as more distant if they differ on an attribute that comes earlier in the ordering. This idea was exemplified by a car-relation with attributes make-model-color-year-description in this order. Hence, for instance, the query engine would preferably output a subset of cars with different models rather than with different colors. Actually, this distance function is an ultrametric, i.e., a metric $\mathbf{d}$, that satisfies the strong triangle inequality $\mathbf{d}(a,c) \leq \max\{\mathbf{d}(a,b), \mathbf{d}(b,c)\}$ for any three elements.

For the generalization of the distance to a diversity δ of sets, one can aggregate the pairwise distances, for instance, by taking their sum or minimum (see, e.g., [17, 22, 33, 34]). In this work, we also want to look at a more sophisticated diversity measure proposed by Weitzman in [36], which we will refer to as δ_{W}. It is motivated by the goal of capturing the increase of diversity (measured as the minimum distance from the already chosen elements) when yet another element is added. Detailed formal definitions of all concepts mentioned here will be given in Section 2.

Aiming at a diverse subset of the answers to a query raises several computational problems. The most basic problem is to actually evaluate the diversity function δ for a given set S of tuples. Clearly, this problem is easy to solve, if δ is defined by taking one of the standard aggregate functions sum or min over some efficiently computable metric (such as the Hamming distance). However, if we take the more sophisticated diversity measure δ_{W} proposed by Weitzman, this is not clear any more. In fact, only an exponential algorithm was proposed in [36] for this task and it was left open, if a polynomial-time algorithm exists. We will settle this open question by proving NP-hardness.

Our ultimate goal is to select a small subset (say, of size k for given $k > 1$) of the query answers so as to maximize the diversity. When considering data complexity and restricting ourselves to FO-queries, query evaluation is tractable and we may assume the entire set S of query answers as *explicitly* given. Now the goal is to find a subset $S' \subseteq S$ of size k such that $\delta(S')$ is maximal. In [22], it was shown that this task is NP-hard even for the simple setting where δ is defined as the sum or as the minimum over the pairwise Hamming distances of the tuples. Taking the Weitzman diversity clearly makes this task yet more complex. It is here that ultrametrics come to the rescue. Indeed, we show tractability of the following problem: given a set S of elements and integer $k > 1$, find a subset S' of S such that $\delta(S')$ is maximal, where δ is a diversity function extending an ultrametric and δ satisfies a certain monotonicity property we call *weak subset-monotonicity*. Moreover, we show that even slightly relaxing the monotonicity property immediately leads to NP-hardness.

Things get yet more complex if we consider combined complexity. Since the set S of query answers can be exponentially big, we cannot afford to compute it upfront. In other words, S is only given *implicitly* by the database D and query Q. But the goal remains the same: we want to find a

subset S' of S with $|S'| = k$ that maximizes the diversity $\delta(S')$. Since query evaluation is intractable for conjunctive queries even without worrying about diversity, we now restrict the query language to acyclic conjunctive queries. We then manage again to prove tractability for the task of finding a subset S' with maximal diversity, provided that δ is *subset-monotone* – a restriction slightly stronger than weak subset-monotonicity but which is satisfied by δ_{W}, for example. Again we show tightness of this tractability result by proving that without this stronger notion of monotonicity, the problem is NP-hard. Finally, we also identify a kind of middle ground in terms of monotonicity of δ that ensures fixed-parameter tractability when considering k as parameter.

Structure of the paper. In Section 2, we introduce basic notions and formally define the computational problems studied here. The complexity of evaluating δ_{W} is studied in Section 3. Fundamental (and well-known) properties of ultrametrics are recalled in Section 4. In Sections 5 and 6, we study the problem of finding a subset $S' \subseteq S$ maximizing $\delta(S')$ for the cases where S is given explicitly or implicitly, respectively. In particular, in Section 6, we first define a general framework that formalizes the notion of an implicitly given set S equipped with an ultrametric. The general results for the implicit setting are then studied in Section 7 for the concrete case of combined complexity of query answering for acyclic conjunctive queries. We discuss related work in Section 8 and we provide a conclusion and an outlook to future work in Section 9. Due to lack of space, proofs are only sketched in the main body of the text. Full proofs of our membership results are provided in the appendix. Full proofs of all results presented here are given in the full version of this paper [4].

2 Preliminaries

Sets and sequences. We denote by $\mathbb{N}$ the set of natural numbers, by $\mathbb{Q}$ the set of rational numbers, and by $\mathbb{Q}_{\geq 0}$ the set of non-negative rational numbers. Given a set A, we denote by $\mathrm{finite}(A)$ the set of all non-empty finite subsets of A. For $k \in \mathbb{N}$, we say that $B \in \mathrm{finite}(A)$ is a *k-subset* if $|B| = k$. We usually use a, b, or c to denote elements, and $\bar{a}$, $\bar{b}$, or $\bar{c}$ to denote sequences of such elements. For $\bar{a} = a_1, \ldots, a_k$, we write $\bar{a}[i] := a_i$ to denote the i-th element of $\bar{a}$ and $|\bar{a}| := k$ to denote the length of $\bar{a}$. Further, given a function f we write $f(\bar{a}) := f(a_1), \ldots, f(a_k)$ to denote the function applied to each element of $\bar{a}$.

Conjunctive queries. Fix a set $\mathbb{D}$ of data values. A relational schema σ (or just schema) is a pair $(\mathcal{R}, \mathrm{arity})$ where $\mathcal{R}$ is a set of relation names and $\mathrm{arity} : \mathcal{R} \to \mathbb{N}$ assigns each name to a number. An R-tuple of σ (or just a tuple) is a syntactic object $R(a_1, \ldots, a_k)$ such that $R \in \mathcal{R}$, $a_i \in \mathbb{D}$ for every i, and $k = \mathrm{arity}(R)$. We will write $R(\bar{a})$ to denote a tuple with values $\bar{a}$. A *relational database* D over σ is a finite set of tuples over σ.

Fix a schema $\sigma = (\mathcal{R}, \mathrm{arity})$ and a set of variables $\mathcal{X}$ disjoint from $\mathbb{D}$. A *Conjunctive Query* (CQ) over σ is a syntactic structure of the form:

$$Q(\bar{x}) \;\leftarrow\; R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m) \qquad (\dagger)$$

such that Q denotes the answer relation and R_i are relation names in $\mathcal{R}$, $\bar{x}_i$ is a sequence of variables in $\mathcal{X}$, $|\bar{x}| = \mathrm{arity}(Q)$, and $|\bar{x}_i| = \mathrm{arity}(R_i)$ for every $i \leq m$. Further, $\bar{x}$ is a sequence of variables appearing in $\bar{x}_1, \ldots, \bar{x}_m$. We will denote a CQ like (†) by Q, where $Q(\bar{x})$ and $R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)$ are called the *head* and the *body* of Q, respectively. Furthermore, we call each $R_i(\bar{x}_i)$ an *atom* of Q.

Let Q be a CQ like (†), and D be a database over the same schema σ. A *homomorphism* from Q to D is a function $h : \mathcal{X} \to \mathbb{D}$ such that $R_i(h(\bar{x}_i)) \in D$ for every $i \leq m$. We define the *answers* of Q over D as the set of Q-tuples

$$[\![Q]\!](D) \;:=\; \{Q(h(\bar{x})) \mid h \text{ is a homomorphism from } Q \text{ to } D\}.$$

Diversity setting. Let $\mathcal{U}$ be an infinite set. We see $\mathcal{U}$ as a *universe* of possible solutions and $S \in \text{finite}(\mathcal{U})$ as a candidate finite set of solutions that cannot be empty. To determine the diversity of S, we first determine how different the pairs of elements are in S, which is done through a metric. A *metric* $\mathbf{d}$ over $\mathcal{U}$ is a function $\mathbf{d} : \mathcal{U} \times \mathcal{U} \to \mathbb{Q}_{\geq 0}$ such that $\mathbf{d}(a, b) = 0$ iff $a = b$, $\mathbf{d}$ is symmetric (i.e., $\mathbf{d}(a, b) = \mathbf{d}(b, a)$), and $\mathbf{d}$ satisfies the triangle inequality (i.e., $\mathbf{d}(a, c) \leq \mathbf{d}(a, b) + \mathbf{d}(b, c)$). We define the distance of an element $a \in \mathcal{U}$ to a set $S \in \text{finite}(\mathcal{U})$ as $\mathbf{d}(a, S) := \min_{b \in S} \mathbf{d}(a, b)$.

Given a metric $\mathbf{d}$, a *diversity function* δ extending $\mathbf{d}$ is a function $\delta\colon \text{finite}(\mathcal{U}) \to \mathbb{Q}_{\geq 0}$ such that $\delta(S) = 0$ iff $|S| = 1$, and $\delta(\{a, b\}) = \mathbf{d}(a, b)$. Note that we see δ as a function that extends $\mathbf{d}$ from pairs to sets, and that is 0 when the set has a single element (i.e., no diversity). Moreover, we impose the restriction that δ should be closed under isomorphism. That is, if $f : \mathcal{U} \to \mathcal{U}$ is a bijective function such that $\mathbf{d}(a, b) = \mathbf{d}(f(a), f(b))$ for every $a, b \in \mathcal{U}$, then $\delta(S) = \delta(f(S))$ for every $S \in \text{finite}(\mathcal{U})$.

As proposed in [17], one way of defining a diversity function δ for a given metric $\mathbf{d}$ over $\mathcal{U}$ is to define an aggregator f that combines the pairwise distances. That is, we set $\delta(S) := f(\mathbf{d}(a, b)_{a,b \in S})$. Common aggregators are sum and min, which give rise to the following diversity functions extending an arbitrary metric $\mathbf{d}$[1]:

$$\delta_{\text{sum}}(S) := \sum_{a,b \in S} \mathbf{d}(a, b) \quad \text{and} \quad \delta_{\text{min}}(S) := \min_{a,b \in S\,:\, a \neq b} \mathbf{d}(a, b).$$

A more elaborate diversity function is the *Weitzman diversity* function δ_{W} [36], which is recursively defined as follows:

$$\delta_{\text{W}}(S) \;:=\; \max_{a \in S} \big(\delta_{\text{W}}(S \setminus \{a\}) + \mathbf{d}(a, S \setminus \{a\}) \big) \tag{1}$$

where $\delta_{\text{W}}(\{a\}) := 0$ is the base case. In [36], it is shown that δ_{W} satisfies several favorable properties. For instance, in many application contexts, the "monotonicity of species" is desirable. That is, adding an element (referred to as "species" in [36]) to a collection should increase its diversity. Now the question is, by how much the diversity δ should increase. Analogously to the first derivative, it seems plausible to request that

$$\delta(S) \;=\; \delta(S \setminus \{a\}) + \mathbf{d}(a, S \setminus \{a\}) \tag{2}$$

should hold for every $a \in S$. That is, the additional diversity achieved by adding element a corresponds to the distance of a to its closest relative in $S \setminus \{a\}$. However, as is argued in [36], since this property can, in general, not be satisfied for every element a, the diversity function δ_{W} provides a reasonable approximation to Equation (2) by taking the maximum over all $a \in S$.

Diversity problems. When confronted with the task of selecting a *diverse* set of elements from an (explicitly or implicitly) given set, we are mainly concerned with three problems – each of them depending on a concrete *diversity* function δ, which in turn is defined over some universe $\mathcal{U}$ of elements. The most basic problem consists in computing the diversity for a given $S \in \text{finite}(\mathcal{U})$:

Problem:	`DiversityComputation`$[\delta]$
Input:	A finite set $S \subseteq \mathcal{U}$
Output:	$\delta(S)$

An additional source of complexity is introduced if the task is to find a subset of S with a certain diversity.

[1]Note that, for $\delta_{\text{sum}}(S)$, the distance between any two distinct elements a, b is contained twice in this sum, namely as $\mathbf{d}(a, b)$ and $\mathbf{d}(b, a)$. We could avoid this by imposing a condition of the form $a < b$ or by dividing the sum by 2. However, this is irrelevant in the sequel and, for the sake of simplifying the notation, we have omitted such an addition.

Problem:	$\texttt{DiversityExplicit}[\delta]$
Input:	A finite set $S \subseteq \mathcal{U}$ and $k > 1$
Output:	$\arg\max_{S' \subseteq S : \lvert S' \rvert = k} \delta(S')$

In light of the previous problem, it is convenient to introduce the following notation: we call S' with $S' \subseteq S$ a "*k-diverse subset of S*", if $S' = \arg\max_{A \subseteq S : \lvert A \rvert = k} \delta(A)$. In other words, the goal of the $\texttt{DiversityExplicit}[\delta]$ problem is to find a k-diverse subset of S.

Things may get yet more complex, if the set $S \subseteq \mathcal{U}$ from which we want to select a subset with maximal diversity is only "implicitly" given. For us, the most important example of such a setting is when S is the set of answer tuples to a given query Q (in particular, an acyclic CQ) over database D and we consider combined complexity. Further settings will be introduced in Section 6. All these settings have in common that S might be exponentially big and one cannot afford to turn the implicit representation into an explicit one upfront.

Problem:	$\texttt{DiversityImplicit}[\delta]$
Input:	An implicit representation of a finite set $S \subseteq \mathcal{U}$ and $k > 1$
Output:	$\arg\max_{S' \subseteq S : \lvert S' \rvert = k} \delta(S')$

By slight abuse of notation, we will formulate intractability results on the three functional problems introduced above in the form of "NP-hardness" results. Strictly speaking, we thus mean the decision variants of the diversity problems, i.e., deciding if the diversity $\delta(S)$ is above a given threshold th or if a set $S' \subseteq S$ with $\delta(S') \geq th$ exists.

Complexity analysis of algorithms. For the implementation of our algorithms, we assume the computational model of Random Access Machines (RAM) with uniform cost measure and addition and subtraction as basic operations [1]. Further, in all the scenarios considered in this paper, a metric $\mathbf{d}$ is defined over a countably infinite set $\mathcal{U}$, and the value $\mathbf{d}(a, b)$ is a non-negative rational number for every $a, b \in \mathcal{U}$. Thus, we assume that the codomain of every metric $\mathbf{d}$ is the set $\mathbb{Q}_{\geq 0}$, which in particular implies that we have a finite representation for each possible value of a metric that can be stored in a fixed number of RAM registers. Moreover, although $\mathbf{d}$ is defined over an infinite set, we will only need its values for a finite set, and we assume that $\mathbf{d}(a, b)$ can be computed in constant time for any pair a, b of elements in this set. Alternatively, one could multiply the complexity of our algorithms by a parameter p that encapsulates the cost of computing $\mathbf{d}(a, b)$ or consider the metric as given by a look-up table at the expense of a quadratic blow-up of the input. Neither of these alternatives would provide any additional insights while complicating the notation or blurring the setting. We have therefore refrained from adopting one of them.

3 Computing diversity is hard

The most basic computational problem considered here is $\texttt{DiversityComputation}[\delta]$. Clearly, for δ_{sum} and δ_{min}, this problem is efficiently solvable. Here, we study the complexity of computing the diversity $\delta(S)$ of a subset $S \subseteq \mathcal{U}$ for the more elaborate Weitzman diversity measure δ_{W}. In [36], it was shown that δ_{W} can be computed efficiently, if the distance function $\mathbf{d}$ it extends is an *ultrametric*. However, for arbitrary distance functions, only an exponential algorithm was presented and, implicitly, NP-membership of (the decision variant of) $\texttt{DiversityComputation}[\delta_W]$ was proven. We show that this upper bound is tight by proving also NP-hardness of this problem.

Theorem 3.1. *The* $\texttt{DiversityComputation}[\delta_{\text{W}}]$ *problem of the Weitzman diversity function* δ_{W} *is* NP*-hard.*

Proof Sketch. NP-hardness of (the decision variant of) the $\texttt{DiversityComputation}[\delta_{\text{W}}]$ problem is shown by reduction from the Independent Set problem. Let an arbitrary instance of

Independent Set be given by a graph $G = (V, E)$ and integer k. Let $|V| = n$. Then we set $S = V$ and $th = n - k + 2(k-1)$, and we define the distance function $\mathbf{d}$ on S as follows:

$$\mathbf{d}(u,v) = \begin{cases} 0 & \text{if } u = v \\ 1 & \text{if } u \text{ and } v \text{ are adjacent in } G \\ 2 & \text{otherwise} \end{cases}$$

It is straightforward to verify that $\mathbf{d}$ is a metric and that G has an independent set of size k, if and only if $\delta_{\mathrm{W}}(S) \geq th$. □

In this work, we are mainly interested in the diversity of sets of tuples – either from the database itself or sets of tuples resulting from evaluating a query over the database. The above NP-hardness proof can be adapted so as to get NP-hardness also for the (decision variant of the) $\mathtt{DiversityComputation}[\delta_{\mathrm{W}}]$ problem if S is a set of tuples, even in a very restricted setting:

Theorem 3.2. *The* $\mathtt{DiversityComputation}[\delta_{\mathrm{W}}]$ *problem of the Weitzman diversity function* δ_{W} *is* NP*-hard, even if* S *is a set of tuples of arity 5 and we take the Hamming distance as distance between any two tuples.*

Proof Sketch. NP-hardness is again shown by reduction from the Independent Set problem. As was shown in [2], the Independent Set problem remains NP-complete even if we restrict the graphs to degree 3. Then the crux of the problem reduction is to construct, from a given graph with n vertices $\{v_1, \ldots, v_n\}$, a set of n tuples $S = \{t_1, \ldots, t_n\}$, such that, for the Hamming distance d between two tuples $t_i \neq t_j$, we have $\mathbf{d}(t_i, t_j) = 4$ if v_i, v_j are adjacent in G and $\mathbf{d}(t_i, t_j) = 5$ otherwise. For this step, we adapt a construction that was used in [22]. As threshold th, we set $th = 4(n-k) + 5(k-1)$. Analogously to Theorem 3.1, it can then be shown that G has an independent set of size k, if and only if $\delta_{\mathrm{W}}(S) \geq th$. □

When moving from $\mathtt{DiversityComputation}[\delta]$ to the $\mathtt{DiversityExplicit}[\delta]$ problem, of course, the complexity is at least as high. So for the Weitzman diversity function δ_{W}, the intractability clearly carries over. However, in case of the $\mathtt{DiversityExplicit}[\delta]$ problem, even simpler diversity settings lead to intractability. More specifically, it was shown in [22, 23] that the $\mathtt{DiversityExplicit}[\delta]$ problem[2] is NP-hard even in the simple setting where S is a set of tuples of arity 5, considering the Hamming distance and one of the simple diversity functions δ_{sum} or δ_{min}. Therefore, in [22], the parameterized complexity of this problem was considered (with k as parameter) and the $\mathtt{DiversityExplicit}[\delta]$ problem was shown to be fixed-parameter tractable, when S is a set of tuples, considering the Hamming distance and very general diversity functions δ satisfying a certain monotonicity property.

Clearly, for the $\mathtt{DiversityImplicit}[\delta]$ problem, things get yet more complex. Indeed, unless FPT = W[1], fixed-parameter tractability was ruled out in [22] by showing W[1]-hardness for the setting where S is the set of answers to an acyclic CQ, considering Hamming distance and one of the simple diversity functions δ_{sum} or δ_{min}. On the positive side, XP-membership was shown for this setting.

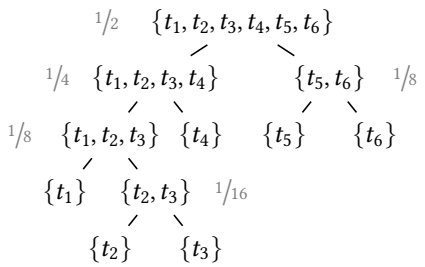
In the remainder of this work, we will consider ultrametrics as an important special case of distance functions. It will turn out that they allow us to prove several positive results for otherwise hard problems. For instance, the $\mathtt{DiversityImplicit}[\delta]$ problem becomes tractable in this case even when we consider the Weitzman diversity measure δ_{W}.

[2]Strictly speaking, the problem considered there was formulated as the task of finding a set of k answer tuples to an acyclic CQ Q over a database D with diversity $\geq th$. NP-hardness was shown for data complexity, which means that we may assume that the set S of answer tuples is explicitly given since, with polynomial-time effort, one can compute S.

CARS

	Make	Model	Color	Year
t_1	Honda	Civic	Green	2007
t_2	Honda	Civic	Black	2007
t_3	Honda	Civic	Black	2006
t_4	Honda	Accord	Blue	2007
t_5	Toyota	Corolla	Black	2007
t_6	Toyota	Corolla	Blue	2007

(a)



(b)

Fig. 1. On the left, a relation CARS where each tuple is a car model. On the right, the ultrametric tree of the ultrametric $\mathbf{u}_{\text{rel}}$ over the tuples S in CARS. On one side of each ball B (in grey) we display its radius $\mathrm{r}_S(B)$.

4 Ultrametrics to the rescue

In this section, we recall the definition of an ultrametric and present some of its structural properties. The results presented here are well-known in the literature of ultrametric spaces. Nevertheless, they are crucial to understand the algorithms for diversity measures shown in the following sections. We provide proofs of these properties in [4].

Ultrametrics. Let $\mathcal{U}$ be a possibly infinite set. An *ultrametric* $\mathbf{u}$ over $\mathcal{U}$ is a metric over $\mathcal{U}$ that additionally satisfies the *strong triangle inequality*:

$$\mathbf{u}(a,c) \leq \max\{\mathbf{u}(a,b), \mathbf{u}(b,c)\}.$$

We use $\mathbf{d}$ to denote a metric and $\mathbf{u}$ to denote an ultrametric, thereby making it explicit that we are using an ultrametric.

As an example, consider the following ultrametric for tuples in a relational database with schema σ. Let $\mathcal{U}$ be the set of all tuples of σ. Define the metric $\mathbf{u}_{\text{rel}}$ such that $\mathbf{u}_{\text{rel}}(R(\bar{a}), R(\bar{a})) = 0$, $\mathbf{u}_{\text{rel}}(R(\bar{a}), S(\bar{b})) = 1$, and $\mathbf{u}_{\text{rel}}(R(\bar{a}), R(\bar{a}')) = 2^{-i}$ with $i = \min\{j \mid \bar{a}[j] \neq \bar{a}'[j]\}$, for arbitrary tuples $R(\bar{a})$, $R(\bar{a}')$, and $S(\bar{b})$ of σ with $R \neq S, \bar{a} \neq \bar{a}'$. In other words, the distance is 1 if tuples comes from different relations, and otherwise 2^{-i} such that i is the first position where (the arguments of) the tuples differ. One can check that $\mathbf{u}_{\text{rel}}$ is an ultrametric since, for arbitrary tuples $R(\bar{a})$, $R(\bar{b})$, $R(\bar{c})$ such that i is the first position where $R(\bar{a})$ and $R(\bar{c})$ differ, it holds that $R(\bar{a})$ and $R(\bar{b})$ differ at position i or $R(\bar{b})$ and $R(\bar{c})$ differ at position i, so that

$$\mathbf{u}_{\text{rel}}(R(\bar{a}), R(\bar{c})) \leq \max\{\mathbf{u}_{\text{rel}}(R(\bar{a}), R(\bar{b})), \mathbf{u}_{\text{rel}}(R(\bar{b}), R(\bar{c}))\}.$$

Similarly, the strong triangle inequality holds for tuples of different relations.

Example 4.1. Consider the following running example which is a simplified version taken from [33, 34] where $\mathbf{u}_{\text{rel}}$ is used as a metric. In Figure 1a, we show the relation CARS that contains car models with the brand (i.e., "Make"), model, color, and year (in that order). Each row represents a tuple and t_i is the name given to refer to the i-th tuple. Then, one can check that $\mathbf{u}_{\text{rel}}(t_1, t_5) = 1/2$ given that t_1 is made by Honda and t_5 by Toyota. Similarly, $\mathbf{u}_{\text{rel}}(t_1, t_2) = 1/8$ given that t_1 and t_2 differ in the color, that is, at position 3 and, thus, $\mathbf{u}_{\text{rel}}(t_1, t_2) = 2^{-3}$. □

Structure of ultrametric spaces over finite sets. Ultrametrics form a class of well-studied metric spaces, which have useful structural properties. For instance, if $\mathbf{d}$ in Equation (2) from Section 2 is an ultrametric, then that equation holds for every $a \in S$. Moreover, every three elements form an isosceles triangle, namely, for every $a, b, c \in \mathcal{U}$ it holds that $\mathbf{u}(a,b) = \mathbf{u}(a,c)$, or $\mathbf{u}(a,b) = \mathbf{u}(b,c)$, or $\mathbf{u}(a,c) = \mathbf{u}(b,c)$. In particular, this implies some well-structured hierarchy on all balls centered at

elements of a finite set, that we introduce next. Fix an ultrametric $\mathbf{u} : \mathcal{U} \times \mathcal{U} \to \mathbb{Q}_{\geq 0}$ and fix a finite set $S \subseteq \mathcal{U}$. For every $a \in S$ and $r \in \mathbb{Q}_{\geq 0}$, let:

$$\mathcal{B}_S(a, r) := \{b \in S \mid \mathbf{u}(a, b) \leq r\}.$$

That is, $\mathcal{B}_S(a, r)$ is the (closed) *ball* centered at a with radius r. Let $\mathcal{B}_S = \{\mathcal{B}_S(a, r) \mid a \in S \wedge r \in \mathbb{Q}_{\geq 0}\}$ be the *set of all balls* of $\mathbf{u}$ over S. Given that S is finite, $\mathcal{B}_S$ is finite as well. Moreover, the set $\mathcal{B}_S$ follows a nested structure given by the following standard properties of ultrametrics.

Property 1. (a) For every $B_1, B_2 \in \mathcal{B}_S$, it holds that $B_1 \cap B_2 = \varnothing$ or $B_1 \subseteq B_2$ or $B_2 \subseteq B_1$.
(b) If $\mathbf{u}(a_1, a_2) \leq r$, then $\mathcal{B}_S(a_1, r) = \mathcal{B}_S(a_2, r)$.

Property 1(a) implies a nested structure among balls in $\mathcal{B}_S$ that we can represent as a tree structure as follows. First, for every $B \in \mathcal{B}_S$, define the set:

$$\text{parent}(B) := \{B' \in \mathcal{B}_S \mid B \subsetneq B' \wedge \neg \exists B'' \in \mathcal{B}_S : B \subsetneq B'' \subsetneq B'\}.$$

Since $(\mathcal{B}_S, \subseteq)$ is a partial order over a finite set, $\text{parent}(B)$ is non-empty for every $B \neq S$. Moreover, by Property 1(a), we have that $\text{parent}(B)$ has at most one element. Then, for every $B \in \mathcal{B}_S \setminus \{S\}$ we can write $\text{parent}(B)$ to denote this single element.

We define the *ultrametric tree of* $\mathbf{u}$ *over* S as the graph $\mathcal{T}_S = (V_S, E_S)$ such that $V_S := \mathcal{B}_S$ and

$$E_S := \{(\text{parent}(B), B) \mid B \in \mathcal{B}_S \setminus \{S\}\}.$$

Given that $(\mathcal{B}_S, \subseteq)$ is a partial order and every B has at most one incoming edge, we have that $\mathcal{T}_S$ is a (directed) tree and S is the root of this tree. Therefore, we can write $\text{children}(B) = \{B' \mid (B, B') \in E_S\}$ to denote the children of B in $\mathcal{T}_S$. Note that B is a leaf in the tree $\mathcal{T}_S$ iff $B = \{a\}$ for some $a \in S$. Further, if $|B| \geq 2$, then $\text{children}(B)$ forms a partition of B by Property 1(a).

Example 4.2. Let S be the set of all tuples in the relation CARS of Example 4.1. In Figure 1b, we display the ultrametric tree of $\mathbf{u}_{\text{rel}}$ over S. One can check in this figure that each leaf contains a single tuple, and the children of each ball form a partition. □

It will be also convenient to relate the distance between elements in S with the radius of the balls in $\mathcal{B}_S$. For this purpose, for every $B \in \mathcal{B}_S$ we define its *radius* as:

$$\text{r}_S(B) := \max\{\mathbf{u}(a, b) \mid a, b \in B\}.$$

Notice that $\text{r}_S(B)$ is well defined since B is a finite set. By Property 1(b), we have that $B = \mathcal{B}_S(a, \text{r}_S(B))$ for every ball $B \in \mathcal{B}_S$ and point $a \in B$. Hence, in what follows, we can use any $a \in B$ as the center of the ball B.

Another crucial property of the radius of B is that it determines the distance between elements of different children of B in $\mathcal{T}_S$ as follows.

Property 2. Let $B_1, B_2 \in \text{children}(B)$ with $B_1 \neq B_2$. Then $\mathbf{u}(a_1, a_2) = \text{r}_S(B)$ for every $a_1 \in B_1, a_2 \in B_2$.

Example 4.3. In the ultrametric tree of Figure 1b we display the radius of each ball. One can check that the root node $S = \{t_1, \ldots, t_6\}$ satisfies that $\text{r}_S(S) = 1/2$ and $\mathbf{u}_{\text{rel}}(t, t') = 1/2$ for every $t \in \{t_1, \ldots, t_4\}$ and $t' \in \{t_5, t_6\}$. Hence, Property 2 holds in this case. □

By Properties 1 and 2, the ultrametric tree $\mathcal{T}_S$ and the radius function r_S completely determine the ultrametric over a finite set S and they will be the starting point for our algorithms. By the following property, we can always construct both in quadratic time over the size of $|S|$.

Property 3. Given an ultrametric $\mathbf{u}$ and a finite set S, we can construct $\mathcal{T}_S$ and r_S in $O(|S|^2)$.

Property 3 above can be easily seen by making use of the classical result that a minimum-weight spanning tree $\mathcal{T}$ of a graph $G = (S, S \times S)$, where $\mathbf{u}$ expresses the edge weight, can be computed in time $O(|S|^2)$ and the fact that $\mathcal{T}_S$ and r_S can be easily computed from $\mathcal{T}$. On the other hand, it is also easy to see that, in general, one cannot compute the ultrametric tree $\mathcal{T}_S$ in time $o(|S|^2)$. Indeed, we just have to consider an ultrametric on a set $S = \{a_1, \ldots, a_n\}$ where only one pair (a_i, a_j) of elements has distance 1 and all other pairs have distance 2. Then one cannot compute $\mathcal{T}_S$ and r_S without finding this pair (a_i, a_j).

5 Ultrametrics for explicit representation

In this section, we present our first algorithmic results for the $\texttt{DiversityExplicit}[\delta]$ problem, i.e., the problem of finding a k-diverse subset of a finite set S given a diversity function δ of an ultrametric $\mathbf{u}$. Here, we assume that S is represented explicitly, namely, S is given as a finite list $a_1, \ldots, a_n$. In order to find tractable scenarios for the explicit case, we introduce the notion of subset-monotonicity for diversity functions.

Definition 5.1 (Subset-monotonicity). A diversity function δ extending a metric $\mathbf{d}$ over a universe $\mathcal{U}$ is said to be *subset-monotone* if, and only if, for every $A, B, B' \subseteq \mathcal{U}$ such that $B = \{b_1, \ldots, b_\ell\}$, $B' = \{b'_1, \ldots, b'_\ell\}$, $A \cap B = A \cap B' = \varnothing$, $\delta(B) \leq \delta(B')$, and $\mathbf{d}(a, b_i) \leq \mathbf{d}(a, b'_i)$ for every $a \in A$ and $i \in \{1, \ldots, \ell\}$, it holds that $\delta(A \cup B) \leq \delta(A \cup B')$.

Subset-monotonicity captures the natural intuition that, if we replace B with B' such that B' is at least as diverse as B and B' is at least as from A as B is, then $A \cup B'$ is at least as diverse as $A \cup B$. We observe that, for ultrametrics, all diversity functions from Section 2 are subset-monotone. The diversity functions δ_{sum} and δ_{min} have this property even for arbitrary metrics.

Proposition 5.2. *The diversity functions δ_{sum} and δ_{min} are subset-monotone no matter the metric they extend. If δ_{W} extends an ultrametric, then it is also subset-monotone. If δ_{W} extends an arbitrary metric, then it is, in general, not subset-monotone.*

Interestingly, if δ is a subset-monotone diversity function extending an ultrametric $\mathbf{u}$, then we can always find a k-diverse subset of a finite set S efficiently. In fact, it is possible to prove this result even if we consider a weaker notion of subset-monotonicity.

Definition 5.3 (Weak subset-monotonicity). A diversity function δ extending a metric $\mathbf{d}$ over a universe $\mathcal{U}$ is said to be *weakly subset-monotone* if, and only if, for every $A, B, B' \subseteq \mathcal{U}$ such that $B = \{b_1, \ldots, b_\ell\}$, $B' = \{b'_1, \ldots, b'_\ell\}$, $A \cap B = A \cap B' = \varnothing$, $\delta(B) \leq \delta(B')$ and $\mathbf{d}(a, b_i) = \mathbf{d}(a, b'_i)$ for every $a \in A$ and $i \in \{1, \ldots, \ell\}$, it holds that $\delta(A \cup B) \leq \delta(A \cup B')$.

In other words, if we replace B with B' such that B' is at least as diverse as B and both have the *same pairwise distance* to A, then $A \cup B'$ is at least as diverse as $A \cup B$. Note that this weaker version is almost verbatim from subset monotonicity, but with $\mathbf{d}(a, b_i) \leq \mathbf{d}(a, b'_i)$ replaced by $\mathbf{d}(a, b_i) = \mathbf{d}(a, b'_i)$. Clearly, subset-monotonicity implies weak subset-monotonicity but not vice versa.

Theorem 5.4. *Let δ be a weakly subset-monotone diversity function extending an ultrametric $\mathbf{u}$. Then $\texttt{DiversityExplicit}[\delta]$ can be solved in time $O(k^2 \cdot f_\delta(k) \cdot |S| + |S|^2)$ where $O(f_\delta(k))$ is the time required to compute δ over a set of size k.*

Note that $f_\delta(k) \leq k^2$ for δ_{sum}, δ_{min}, and δ_{W} (when extending an ultrametric), so we conclude from Theorem 5.4 that the problem $\texttt{DiversityExplicit}[\delta]$ can be solved in polynomial time for these fundamental diversity functions.

Proof Sketch of Theorem 5.4. The main ideas of the algorithm for Theorem 5.4 are the following. Let S be a finite subset of the universe $\mathcal{U}$, $\mathbf{u}$ an ultrametric over $\mathcal{U}$, and δ a weakly subset-monotone diversity function extending $\mathbf{u}$. By Proposition 3, we can construct an ultrametric tree

$\mathcal{T}_S$ of S in time $O(|S|^2)$. For the sake of simplification, assume that $\mathcal{T}_S$ is a binary tree. Otherwise, one can easily extend the following ideas to the non-binary case. For each vertex B (i.e., a ball) of $\mathcal{T}_S$, we maintain a function $\mathrm{C}_B\colon \{0,\ldots,\min\{k,|B|\}\} \to 2^B$ where $\mathrm{C}_B(i)$ is an i-diverse subset of B. That is, for every $i \in \{0,\ldots,\min\{k,|B|\}\}$ we have $\mathrm{C}_B(i) := \arg\max_{A\subseteq B:|A|=i} \delta(A)$. Clearly, if we can compute C_S for the root S of $\mathcal{T}_S$, then $\mathrm{C}_S(k)$ is a k-diverse subset for S.

The algorithm follows a dynamic programming approach, computing C_B for each $B \in \mathcal{B}_S$ in a bottom-up fashion over $\mathcal{T}_S$. For every ball B, we can easily check that $\mathrm{C}_B(0) = \varnothing$ and $\mathrm{C}_B(1) = \{a\}$ for some $a \in B$. In particular, $\mathrm{C}_{\{a\}} = \{0 \mapsto \varnothing, 1 \mapsto \{a\}\}$ is our base case for every leaf $\{a\}$ of $\mathcal{T}_S$. For an inner vertex B of $\mathcal{T}_S$, the process is a bit more involved. Let B_1 and B_2 be the two children of B in $\mathcal{T}_S$ and assume that we have already computed C_{B_1} and C_{B_2}. We claim that, for every $i \in \{0,\ldots,\min\{k,|B|\}\}$, we can calculate $\mathrm{C}_B(i)$ as $\mathrm{C}_B(i) = \mathrm{C}_{B_1}(i_1) \cup \mathrm{C}_{B_2}(i_2)$, where i_1, i_2 are obtained as follows:

$$(i_1, i_2) = \arg\max_{(j_1,j_2):j_1+j_2=i} \delta(\mathrm{C}_{B_1}(j_1) \cup \mathrm{C}_{B_2}(j_2)).$$

Intuitively, when maximizing diversity with i elements of B, one must try all combinations of a j_1-diverse subset from B_1 and a j_2-diverse subset from B_2, such that $i = j_1 + j_2$ holds. That is, the best elements to pick from B_1 and B_2 are found in C_{B_1} and C_{B_2}, respectively.

To see that the claim holds, let A be a subset of B with i-elements maximizing $\delta(A)$. Define $A_1 := A \cap B_1$ and $A_2 := A \cap B_2$, and their sizes $i_1 := |A_1|$ and $i_2 := |A_2|$, respectively. Due to the optimality of C_{B_1}, we know that $\delta(A_1) \le \delta(\mathrm{C}_{B_1}(i_1))$. Further, $\mathbf{u}(a_1, a_2) = \mathbf{u}(a_1', a_2) = \mathrm{r}_S(B)$ for every $a_1 \in A_1, a_1' \in \mathrm{C}_{B_1}(i_1), a_2 \in A_2$ by Property 2. Then the conditions of weak subset-monotonicity are satisfied and $\delta(A_1 \cup A_2) \le \delta(\mathrm{C}_{B_1}(i_1) \cup A_2)$. Following the same argument, we can conclude that $\delta(\mathrm{C}_{B_1}(i_1) \cup A_2) \le \delta(\mathrm{C}_{B_1}(i_1) \cup \mathrm{C}_{B_2}(i_2))$, proving that $\mathrm{C}_{B_1}(i_1) \cup \mathrm{C}_{B_2}(i_2)$ is optimal.

By the previous ideas, the desired algorithm with time complexity $O(k^2 \cdot f_\delta(k) \cdot |S| + |S|^2)$ for solving the `DiversityExplicit`$[\delta]$ problem follows. □

By Theorem 5.4, we get the following result for finding k-diverse outputs of CQ query evaluation, where $f_Q(D) \le |D|^{|Q|}$.

Corollary 5.5. *Let $\mathbf{u}$ be an ultrametric over tuples, δ be a weakly subset-monotone diverse function extending $\mathbf{u}$, and Q be a fixed CQ (i.e., data complexity). Given a relational database D, and a value k (in unary), we can compute a k-diverse subset of $[\![Q]\!](D)$ with respect to δ in time $O(k^2 \cdot f_\delta(k) \cdot |[\![Q]\!](D)| + |[\![Q]\!](D)|^2 + f_Q(D))$ where $f_Q(D)$ is the time required to evaluate Q over D.*

An open question is whether we can extend Theorem 5.4 beyond weakly subset-monotone diversity functions (extending ultrametrics). We provide here a partial answer by focusing on monotone diversity functions.

Definition 5.6 (Monotonicity). A diversity function δ extending a metric $\mathbf{d}$ is said to be *monotone* if, and only if, for every $A, A' \subseteq \mathcal{U}$ such that $A = \{a_1,\ldots,a_\ell\}$, $A' = \{a_1',\ldots,a_\ell'\}$, and $\mathbf{d}(a_i, a_j) \le \mathbf{d}(a_i', a_j')$ for every $i, j \in \{1,\ldots,\ell\}$, it holds that $\delta(A) \le \delta(A')$.

Monotone diversity functions were considered in [22] as a general class of natural diversity functions. In the next result, we show that monotonicity of the diversity function δ extending some ultrametric is, in general, not enough to make the `DiversityExplicit`$[\delta]$ problem tractable.

Theorem 5.7. *The* `DiversityExplicit`$[\delta]$ *problem is* NP*-hard even for a monotone, efficiently computable diversity function δ extending an ultrametric.*

That is, the previous result implies that there are monotone diversity functions beyond the weakly subset-monotone class where the algorithmic strategy of Theorem 5.4 cannot be used.

6 Ultrametrics for implicit representations

We move now to study the case when S is represented implicitly. Our motivation for implicit representations is to model the query evaluation setting: we receive as input a query Q and a database D, and we want to compute a k-diverse subset of $S = [\![Q]\!](D)$. The main challenge is that S could be of exponential size concerning $|Q|$ and $|D|$; namely, S is implicitly encoded by Q and D, and it is not efficient first to compute S to find a k-diverse subset of S. To formalize this setting in general, given a universe $\mathcal{U}$, we say that an *implicit schema over* $\mathcal{U}$ is a tuple $(\mathcal{I}, [\![\cdot]\!])$ where $\mathcal{I}$ is a set of objects called implicit representations, and $[\![\cdot]\!]$ is a function that maps every implicit representation $I \in \mathcal{I}$ to a finite subset of $\mathcal{U}$. Further, we assume the existence of a size function $|\cdot| : \mathcal{I} \to \mathbb{N}$ that represents the size $|I|$ of each implicit representation $I \in \mathcal{I}$. For example, $\mathcal{I}$ can be all pairs (Q, D) where Q is a CQ and D is a relational database, $[\![\cdot]\!]$ maps each pair (Q, D) to $[\![Q]\!](D)$, and $|(Q, D)| = |Q| + |D|$. Note that we do not impose any restriction on the number of elements of $[\![I]\!]$, so it can be arbitrarily large with respect to $|I|$. Our goal in this section is to compute efficiently, given an implicit representation $I \in \mathcal{I}$ and $k > 1$, a k-diverse subset of $[\![I]\!]$ with respect to a diversity function δ extending an ultrametric $\mathbf{u}$.

Given this general scenario, we need a way to navigate through the elements of $[\![I]\!]$. In particular, we need a way to navigate the ultrametric tree $\mathcal{T}_{[\![I]\!]}$ of $\mathbf{u}$ over $[\![I]\!]$. Like $[\![I]\!]$, $\mathcal{T}_{[\![I]\!]}$ could be arbitrarily large with respect to $|I|$, so it could be unfeasible to construct $\mathcal{T}_{[\![I]\!]}$ explicitly. For this reason, we will assume that our implicit schemas admit some efficient algorithms for traversing ultrametric trees. Formally, an *implicit ultrametric tree* for an implicit schema $(\mathcal{I}, [\![\cdot]\!])$ consists of three algorithms $(\texttt{Root}, \texttt{Children}, \texttt{Member})$ such that, given an implicit representation $I \in \mathcal{I}$ and a ball $B \in \mathcal{B}_{[\![I]\!]}$:

(1) $\texttt{Root}(I)$ computes the root of $\mathcal{T}_{[\![I]\!]}$ in polynomial time with respect to $|I|$;
(2) $\texttt{Children}(I, B)$ enumerates all children of B in $\mathcal{T}_{[\![I]\!]}$ with polynomial delay w.r.t. $|I|$; and
(3) $\texttt{Member}(I, B)$ outputs one solution in B in polynomial time with respect to $|I|$.

Further, when we say that a method receives a ball B as input or enumerates B as output, it means an ID representing the ball B. Recall that $B \subseteq [\![I]\!]$ and then B could be large with respect to $|I|$. For this reason, methods $\texttt{Root}(I)$ and $\texttt{Children}(I, B)$ output IDs representing balls in $\mathcal{B}_{[\![I]\!]}$, that one later uses to call `Children` and `Member`. Here, we assume that each ID has the size of one register of the RAM or a small number of registers that one can bound by some parameter on I (e.g., the arity of query answers if $(\mathcal{I}, [\![\cdot]\!])$ models the query evaluation setting). For example, in the next section we show that such a representation exists in the case of acyclic CQs. Regarding performance, we say that we can compute an implicit ultrametric tree in time $O(f_{\mathcal{T}}(I))$, for some function $f_{\mathcal{T}}$, if the running time of `Root` and `Member`, and the delay of `Children` are in $O(f_{\mathcal{T}}(I))$. Note that we can always assume that $f_{\mathcal{T}}(I) \leq |I|^{\ell}$ for some constant ℓ.

Unlike the results presented in Section 5, there is a difference in the complexity of the problem $\texttt{DiversityImplicit}[\delta]$ depending on whether a diversity function is subset-monotone or weakly subset-monotone. First, it is possible to show that $\texttt{DiversityImplicit}[\delta]$ is tractable when restricted to the class of subset-monotone diversity functions.

Theorem 6.1. *Let $(\mathcal{I}, [\![\cdot]\!])$ be an implicit schema and $\mathbf{u}$ an ultrametric over a common universe $\mathcal{U}$ that admit an implicit ultrametric tree, and δ be a subset-monotone diversity function extending $\mathbf{u}$. Further, assume that the running time of computing δ over a k-subset of $\mathcal{U}$ is bounded by $O(f_\delta(k))$, and we can compute the implicit ultrametric tree in time $O(f_{\mathcal{T}}(I))$. Then, the problem $\texttt{DiversityImplicit}[\delta]$ can be solved in time $O(k \cdot f_{\mathcal{T}}(I) + k^2 \cdot f_\delta(k))$.*

Proof sketch. To achieve this run time, we employ Algorithm 1. This algorithm assumes a fixed implicit representation $(\mathcal{I}, [\![\cdot]\!])$ over $\mathcal{U}$, including a fixed implicit ultrametric tree given by the methods $(\texttt{Root}, \texttt{Children}, \texttt{Member})$. In addition, the ultrametric $\mathbf{u}$ over $\mathcal{U}$ and the subset-monotone

Algorithm 1: For fixed ultrametric $\mathbf{u}$ and implicit representation $(\mathcal{I}, [\![\cdot]\!])$ over a common universe $\mathcal{U}$, implicit ultrametric tree $(\texttt{Root}, \texttt{Children}, \texttt{Member})$, and subset-monotone diversity function δ extending $\mathbf{u}$, compute, for an instance $I \in \mathcal{I}$, a k-diverse subset of $[\![I]\!]$.

Input: An instance $I \in \mathcal{I}$ and $k \in \mathbb{N}$.
Output: A k-diversity set $S \subseteq [\![I]\!]$ with respect to δ.

```
1  B_root ← Root(I)
2  S ← {Member(I, B_root)}
3  L ← {B_root}
4  Children(I, B_root).init
5  Children(I, B_root).next
6  while |S| < k ∧ L ≠ ∅ do
7      B ← arg max_{B∈L} δ(S ∪ {Member(I, Children(I, B).current)})
8      S ← S ∪ {Member(I, Children(I, B).current)}
9      if Children(I, B).next = false then
10         L ← L ∖ {B}
11         for B′ ∈ Children(I, B) do
12             if |B′| > 1 then
13                 L ← L ∪ {B′}
14                 Children(I, B′).init
15                 Children(I, B′).next
16 return S
```

diversity function δ extending $\mathbf{u}$ are fixed. Then, given an instance $I \in \mathcal{I}$ and a $k \in \mathbb{N}$, the algorithm computes a k-diverse set $S \subseteq [\![I]\!]$ with respect to δ.

Recall that, given a ball $B \in \mathcal{B}([\![I]\!])$, the method $\texttt{Children}(I, B)$ enumerates the children of B in $\mathcal{T}_{[\![I]\!]}$ with polynomial delay. For using this enumeration process, we assume an iterator interface with methods `init`, `next`, and `current`, such that: (1) $\texttt{Children}(I, B)$.`init` starts the iteration, placing the `current` pointer to the first child of B; (2) $\texttt{Children}(I, B)$.`current` retrieves the current child of B; (3) $\texttt{Children}(I, B)$.`next` moves the `current` pointer to the next child of B, outputting `true` if a next child exists, and `false`, otherwise. The running times of these methods are $O(f_{\mathcal{T}}(I))$.

For the sake of simplification, we assume that the method $\texttt{Member}(I, B)$ always outputs the same solution as $\texttt{Member}(I, B_{\text{fc}})$ where B_{fc} is the first-child of B in the implicit ultrametric tree. In other words, if we call $\texttt{Children}(I, B)$.`init`, then it always holds that:

$$\texttt{Member}(I, B) = \texttt{Member}(I, \texttt{Children}(I, B).\texttt{current}). \qquad (\ddagger)$$

Intuitively, Algorithm 1 keeps a set S of solutions, and its primary goal is to maximize the "incremental" diversity of adding a new element to S. This new element is chosen from balls of the ultrametric tree that have not been visited yet. For this purpose, the algorithm maintains a set L of balls such that, for each $B \in L$, the algorithm is iterating through the children of B. The algorithm picks the new element that maximizes the incremental diversity of S from L by computing (line 7):

$$B \leftarrow \arg\max_{B \in L} \delta(S \cup \{\texttt{Member}(I, \texttt{Children}(I, B).\texttt{current})\}.$$

Then, the algorithm adds $\texttt{Member}(I, \texttt{Children}(I, B).\texttt{current})$ to S (line 8) and moves to the next children of B (line 9) until S has size k or L is empty (line 6).

Following the above strategy, Algorithm 1 starts by picking the root ball $B_{\texttt{root}} = \texttt{Root}(I)$ and adds a first solution to S (line 2). Then, it adds $B_{\texttt{root}}$ to L for iterating through its children (lines 3 and 4). By assumption (‡), the solution in $B_{\texttt{root}}$ added to S is the same as the first children of $B_{\texttt{root}}$. For this reason, the algorithm skips the first children of $B_{\texttt{root}}$ by calling $\texttt{Children}(I, B_{\texttt{root}}).\texttt{next}$.

When Algorithm 1 reaches the end of the children of a ball $B \in L$ (line 9), it removes B from L (line 10). After this, it iterates over all the children B' of B (line 11), adding B' to L whenever B' is not a leaf node in the ultrametric tree, namely, $|B'| > 1$ (lines 12 and 13). Again, by assumption (‡) we can skip the first child of B', given that a solution of the first child is already in S (lines 14-15).

For the correctness of the algorithm it remains to show the following: (1) The element $l := \texttt{Member}(I, \texttt{Children}(I, B).\texttt{current})$ added to S in line 8 maximizes the incremental diversity, i.e., $\delta(S \cup \{l\}) = \max_{l' \in [\![I]\!]} \delta(S \cup \{l'\})$; (2) For subset-monotone diversity functions, proceeding greedily always leads to an optimal k-diverse set.

For the run time, note that the number of balls of the ultrametric tree that are used is at most $O(k)$. By caching the result of the functions `Children` and `Member`, we can bound the running time of these methods in the algorithm by $O(k \cdot f_{\mathcal{T}}(I))$. Furthermore, one can check that $|L| = O(k)$ and, for each iteration, we compute B by calling δ exactly $|L|$-times over a set of size at most k. Then, the running time of line 7 takes at most $O(k \cdot f_\delta(k))$. Overall, we can bound the running time of Algorithm 1 by $O(k \cdot f_{\mathcal{T}}(I) + k^2 \cdot f_\delta(k))$. □

Unfortunately, and in contrast with Theorem 5.4 in Section 5, the tractability of the problem $\texttt{DiversityImplicit}[\delta]$ no longer holds if we consider weakly subset-monotone diversity functions. To prove this, let $\mathbf{d}$ be any metric over a universe $\mathcal{U}$, and define the *sum-min diversity function* $\delta_{\text{sum-min}}$ as follows. For every set $S \in \text{finite}(\mathcal{U})$: $\delta(S) = 0$ if $|S| = 1$, and $\delta(S)$ is given by the following expression if $|S| > 1$:

$$\delta_{\text{sum-min}}(S) \;:=\; \sum_{a \in S} \mathbf{d}(a, S \setminus \{a\}) \;=\; \sum_{a \in S} \min_{b \in S \,:\, b \neq a} \mathbf{d}(a, b).$$

Intuitively, $\delta_{\text{sum-min}}$ is summing the contribution of each element $a \in S$ to the diversity of S, namely, how far is a from the other elements in S. One can see $\delta_{\text{sum-min}}$ as a non-recursive version of the Weitzman diversity function. Like the other diversity functions used before, we can prove that $\delta_{\text{sum-min}}$ is also a weakly subset-monotone diversity function.

Proposition 6.2. *$\delta_{\text{sum-min}}$ is weakly subset-monotone if it extends an ultrametric.*

As we show next, $\delta_{\text{sum-min}}$ serves as an example of a weakly subset-monotone diversity function that extends an ultrametric $\mathbf{u}$ for which one can find an implicit representation that admits an implicit ultrametric tree, but where it is hard to find a k-diverse subset.

Theorem 6.3. *There exists an implicit schema $(\mathcal{I}, [\![\cdot]\!])$ and an ultrametric $\mathbf{u}$ over a common universe $\mathcal{U}$ which admit an implicit ultrametric tree but for which $\texttt{DiversityImplicit}[\delta_{\text{sum-min}}]$ is* NP*-hard.*

A natural question is whether a relaxed notion of tractability in the form of fixed-parameter tractable (FPT) can lead to algorithm for $\texttt{DiversityImplicit}[\delta_{\text{sum-min}}]$. We conclude this section by providing an answer to this question. The crucial property of $\delta_{\text{sum-min}}$ is *incremental monotonicity*, which we define next.

Definition 6.4 (Incremental monotonicity). A diversity function δ extending a metric $\mathbf{d}$ over a universe $\mathcal{U}$ is *incrementally monotone* if, and only if, for every set $A \subseteq \mathcal{U}$ and pair $b, b' \in \mathcal{U}$ such that $A \cap \{b, b'\} = \varnothing$ and $\mathbf{d}(a, b) \leq \mathbf{d}(a, b')$ for every $a \in A$, it holds that $\delta(A \cup \{b\}) \leq \delta(A \cup \{b'\})$.

In other words, if we want to grow a set A with b or b', then $A \cup \{b'\}$ will be at least as diverse as $A \cup \{b\}$ given that b' is farther from A than b. Every subset-monotone function is also incrementally

Algorithm 2: For fixed ultrametric $\mathbf{u}$ and implicit representation $(\mathcal{I}, [\![\cdot]\!])$ over a common universe $\mathcal{U}$, implicit ultrametric tree $(\texttt{Root}, \texttt{Children}, \texttt{Member})$, incrementally monotone diversity function δ extending $\mathbf{u}$, and instance $I \in \mathcal{I}$, compute a k-diverse subset of $[\![I]\!]$.

Input: An instance $I \in \mathcal{I}$ and $k \in \mathbb{N}$.
Output: A k-diversity set $S' \subseteq [\![I]\!]$ with respect to δ.

1 $B_{\texttt{root}} \leftarrow \texttt{Root}(I)$
2 $S \leftarrow \mathit{RelevantElements}(I, B_{\texttt{root}}, k)$
3 **return** $\arg\max_{S' \subseteq S, |S'|=k} \delta(S')$
4 **Function** $\mathit{RelevantElements}(I, B, k)$**:**
5 **if** $k = 1$ *or* $|B| = 1$ **then**
6 **return** $\{\texttt{Member}(I, B)\}$
7 $\texttt{Children}(I, B).\texttt{init}$
8 $C \leftarrow \{\texttt{Children}(I, B).\texttt{current}\}$
9 **while** $\texttt{Children}(I, B).\texttt{next} = \texttt{true} \wedge |C| < k$ **do**
10 $C \leftarrow C \cup \{\texttt{Children}(I, B).\texttt{current}\}$
11 $S \leftarrow \{\}$
12 **for** $B_{\texttt{child}} \in C$ **do**
13 $S \leftarrow S \cup \mathit{RelevantElements}(I, B_{\texttt{child}}, k - |C| + 1)$
14 **return** S

monotone, but weak subset-monotonicity does not necessarily imply incremental monotonicity. However, one can check that all diversity functions used in this paper are incrementally monotone, in particular, $\delta_{\text{sum-min}}$.

Theorem 6.5. *For every implicit schema $(\mathcal{I}, [\![\cdot]\!])$ and ultrametric $\mathbf{u}$ over a common universe $\mathcal{U}$ which admits an implicit ultrametric tree, and for every computable incrementally monotone diversity function δ extending $\mathbf{u}$, the problem* $\texttt{DiversityImplicit}[\delta]$ *is fixed-parameter tractable in k.*

Proof sketch. The FPT computation of the $\texttt{DiversityImplicit}[\delta]$ problem is realized in Algorithm 2. We have the same input and output as in Algorithm 1 (apart for the diversity function now being incrementally monotone), and we again assume an iterator interface for Children, i.e., with methods init, next, and current. Furthermore, Root, Children, Member all run in time $O(f_{\mathcal{T}}(I))$ with $f_{\mathcal{T}}(I) \leq |I|^{\ell}$ for some constant ℓ.

Intuitively, Algorithm 2 navigates the ultrametric tree top-down by using its implicit representation, but this time it considers balls up to distance k from the root. Moreover, it selects one solution from each ball. In this way, we get a set of "relevant elements" $S \subseteq [\![I]\!]$ such that there exists a k-diverse subset S' of $[\![I]\!]$ which is also a subset of S.

To compute S, the algorithm proceeds recursively, starting with $B_{\texttt{root}} = [\![I]\!]$ for which we are looking for elements such that the k most diverse ones are among them (call in line 2). In the recursion, instead of $B_{\texttt{root}}$ we could have any ball $B \in \mathcal{B}_{[\![I]\!]}$.

Then, if $k = 1$, it does not matter which element $a \in B$ is selected. Or, if $|B| = 1$, we can simply select the $a \in B$ as it is the only element we have at our disposal (lines 5-6).

Otherwise, there are at least 2 children of B. To that end, let $B_1, \ldots, B_l$ be the children of B. In that case, we recurse on the children B_i with $i \in \{1, \ldots, \min\{k, l\}\}$ and we are looking for elements of B_i such that the $k - \min\{k, l\} + 1$ most diverse ones are among them (the children as collected in

lines 7-10 and the recursion happens in line 13). The reason behind this is that (due to incremental monotonicity) there exists a k-diverse subset S' of B such that S' has at least 1 element from each of the children B_i with $i \in \{1, \ldots, \min\{k, l\}\}$. Intuitively, if S' does not intersect some B_i, we can simply replace any $a \in S' \cap B_j$ from any $B_j \neq B_i$ with any $a' \in B_i$. The union of the elements deemed relevant for the children B_i are then together the elements deemed relevant for B (lines 11-14).

For the correctness of the algorithm, it remains to show that, if δ is incrementally monotone, then a k-diverse subset of S is a k-diverse subset of $[\![I]\!]$. For the FPT running time, we note that, in the worst case, the ultrametric tree is binary and we have to consider all balls up to depth k. However, this means that the size of S is bounded by 2^k and, thus, computing a k-diverse subset of S is possible in time $O(\binom{2^k}{k} \cdot f_\delta(k))$. □

7 Efficient computation of diverse answers to ACQs

In this section, we use the results for implicit representations from the previous section to obtain efficient algorithms for finding k-diverse subsets of the answers to acyclic CQ (ACQ) with respect to the ultrametric $\mathbf{u}_{\text{rel}}$ over tuples presented in Section 4. Note that here we study algorithms for ACQ in combined complexity (i.e., the query Q is not fixed), in contrast to Corollary 5.5 whose analysis is in data complexity (i.e., Q is fixed). In the following, we start by recalling the definition of ACQ and discussing the ultrametric $\mathbf{u}_{\text{rel}}$. Then, we show our main results concerning computing diverse query answers for ACQ.

Acyclic CQ is the prototypical subclass of conjunctive queries that allow for tractable query evaluation (combined complexity) [5, 39]. We therefore also take ACQ as the natural starting point in our effort to develop efficient algorithms for finding k-diverse sets. Let Q be a CQ like in (†). A *join tree* for Q is a labeled tree $T = (V, E, \lambda)$ where (V, E) is a undirected tree and λ is a bijective function from V to the atoms $\{R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)\}$. Further, a join tree T must satisfy that each variable $x \in \mathcal{X}$ forms a connected component in T, namely, the set $\{v \in V \mid x \text{ appears in the atom } \lambda(v)\}$ is connected in T. Then Q is *acyclic* iff there exists a join tree for Q. Also, we say that Q is a *free-connex ACQ* iff Q is acyclic and the body together with the head of Q is acyclic (i.e., admits a join tree).

For our algorithmic results over ACQ, we restrict to the ultrametric $\mathbf{u}_{\text{rel}}$ over tuples of a schema σ, previously defined in Section 4. Arguably, $\mathbf{u}_{\text{rel}}$ is a natural ultrametric for comparing tuples of relations that has been used for computing diverse subsets in previous work [33, 34]. Let Q be a CQ with head $Q(\bar{x})$. Note that the variable order $\bar{x}$ in $Q(\bar{x})$ is important for measuring the diversity of subsets of $[\![Q]\!](D)$ for some database D. Concretely, if we have two CQ Q and Q' with the same body but with different orders in their heads, then diversity of the subsets of $[\![Q]\!](D)$ and $[\![Q']\!](D)$ could be totally different. Furthermore, $\mathbf{u}_{\text{rel}}$ allows us to characterize balls in $\mathcal{B}_{[\![Q]\!](D)}$ by their common prefix. This property will be crucial in the following and the developed proof techniques can, therefore, be naturally adapted to any ultrametric with this property.

By using Theorem 6.1 over ACQ and the ultrametric $\mathbf{u}_{\text{rel}}$, we can find quasilinear time algorithms with respect to $|D|$ for finding k-diverse sets of query answers.

Theorem 7.1. *Let δ be a subset-monotone diversity function extending the ultrametric $\mathbf{u}_{\text{rel}}$ such that the running time of computing δ over a set of size k is bounded by $O(f_\delta(k))$. Given an ACQ Q, a relational database D, and a value k (in unary), a k-diverse subset of $[\![Q]\!](D)$ with respect to δ can be computed in time $O(k \cdot |Q| \cdot |D| \cdot \log(|D|) + k^2 \cdot f_\delta(k))$.*

Proof Sketch. To apply Theorem 6.1, we need to describe an implicit ultrametric tree for tuples in $[\![Q]\!](D)$. To that end, we use the fact that for $\mathbf{u}_{\text{rel}}$, the balls in $\mathcal{B}_{[\![Q]\!](D)}$ can be represented by the common prefix of their tuples. More specifically, for every ball $B \in \mathcal{B}_{[\![Q]\!](D)}$ there exist values $c_1, \ldots, c_i$ such that $B = \{Q(\bar{a}) \in [\![Q]\!](D) \mid \forall j \leq i.\, \bar{a}[j] = c_j\}$. Then, we can traverse the ultrametric tree of $\mathbf{u}_{\text{rel}}$ over $[\![Q]\!](D)$, by managing partial outputs (i.e., prefixes) of $[\![Q]\!](D)$.

Let Q be an ACQ like (†). To arrive at the methods `Root`, `Children`, and `Member`, we modify Yannakakis algorithm [39] to find the following data values that extend a given prefix. Concretely, given values $c_1, \ldots, c_i$ that represent a ball B, we want to find all values c such that $c_1, \ldots, c_i, c$ is the prefix of a tuple in $[\![Q]\!](D)$. For this, we can consider the subquery $Q'(\bar{x}[i{+}1])) \leftarrow R_1(h(\bar{x}_1)), \ldots, R_m(h(\bar{x}_m))$ where h is a partial assignment that maps $h(\bar{x}[j]) = c_j$ for every $j \leq i$ and $h(x) = x$ for any other variable $x \in \mathcal{X}$. The subquery Q' is also acyclic and returns all the desired values c, such that $c_1 \ldots, c_i, c$ represents a child of B. Thus, running Yannakakis algorithm over Q' and D, we can compute `Children` in time $O(|Q| \cdot |D| \cdot \log(|D|))$ and similarly for the methods `Root` or `Member`. □

A natural next step to improve the running time of Theorem 7.1 is to break the dependency between k and $|Q| \cdot |D| \cdot \log(|D|)$. Towards this goal, we build on the work of Carmeli et al. [7], which studied direct access to ranked answers of conjunctive queries. In this work, the algorithmic results also depend on the attribute order, characterizing which CQs and orders admit direct access to the results. For this characterization, the presence of a disruptive trio in the query is crucial. Let Q be a CQ like (†). We say that two variables x and y in Q are neighbors if they appear together in Q in some atom. Then we say that three positions i, j, k (i.e., variables $\bar{x}[i]$, $\bar{x}[j]$, and $\bar{x}[k]$) in the head of Q form a *disruptive trio* iff $\bar{x}[i]$ and $\bar{x}[j]$ are not neighbors in Q, and $\bar{x}[k]$ is a neighbor of $\bar{x}[i]$ and $\bar{x}[j]$ in Q, but $i < k$ and $j < k$ (i.e., $\bar{x}[k]$ appears after $\bar{x}[i]$ and $\bar{x}[j]$). For example, for the query $Q(x_1, x_2, x_3, x_4) \leftarrow R(x_1, x_2), S(x_2, x_4), T(x_4, x_3)$, the positions $2, 3, 4$ form a disruptive trio, but $1, 2, 3$ do not.

In the following result, we show that free-connex ACQ and the absence of a disruptive trio are what we need to get better algorithms for computing k-diverse subsets.

Theorem 7.2. *Let δ be a subset-monotone diversity function extending the ultrametric $\boldsymbol{u}_{\text{rel}}$ such that the running time of computing δ over a set of size k is bounded by $O(f_\delta(k))$. Given a free-connex ACQ Q without a disruptive trio, a relational database D, and a value k (in unary), a k-diverse subset of $[\![Q]\!](D)$ with respect to δ can be computed in time $O(|Q| \cdot |D| \cdot \log(|D|) + k \cdot |Q| + k^2 \cdot f_\delta(k))$.*

Proof Sketch. Similar to Theorem 7.1, we use the prefix of tuples to represent balls and take advantage of the structure of a join tree and the absence of disruptive trios to implement an index over D. By [7], the absence of disruptive trios ensures the existence of a *layered join tree* whose layers follow the order of the variables in the head of Q and which can be computed in time $O(|Q| \cdot |D| \cdot \log(|D|))$. Then, by building on this constructed layered join tree, we can set up an index structure that we can use to calculate the next data value for a given prefix, like in Theorem 7.1, but now in time $O(|Q|)$. Thus, after a common preprocessing phase, `Root`, `Children` and `Member` run in time $O(|Q|)$. □

Note that in data complexity, the only remaining non-(quasi)linear term in Theorem 7.2 is $k^2 \cdot f_\delta(k)$ which arises since we have to reevaluate δ at each step to find the next greedily best pick. For some specific diversity function this may not be necessary. As an example, for the Weitzman diversity function δ_{W} we can get rid of this term $k^2 \cdot f_\delta(k)$ by smartly keeping track of which answer maximizes the diversity next.

Theorem 7.3. *Let δ_{W} be the Weitzman diversity function extending the ultrametric $\boldsymbol{u}_{\text{rel}}$. Given a free-connex ACQ Q without a disruptive trio, a relational database D, and a value k (in unary), a k-diverse subset of $[\![Q]\!](D)$ with respect to δ_{W} can be computed in time $O(|Q| \cdot |D| \cdot \log(|D|) + k \cdot |Q|)$.*

8 Related work

Diversification. Aiming for a small, *diverse* subset of the solutions has been adopted in many areas as a viable strategy of dealing with a solution space that might possibly be overwhelmingly big.

This is, in particular, the case in data mining, information retrieval, and web science, where the term *"diversification of search results"* is commonly used for the process of extracting a small diverse subset from a huge set of solutions, see e.g., [8, 12, 27, 28, 31] and the surveys [37, 40]. The diversity of solutions has also been intensively studied by the Artificial Intelligence (AI) community. Notably, this is the case in subfields of AI which are most closely related to database research, namely constraint satisfaction (recall that, from a logical point of view, solving constraint satisfaction problems and evaluating conjunctive queries are equivalent tasks) [14, 15, 17, 26] and answer set programming (which corresponds to datalog with unrestricted negation under stable model semantics) [10]. In the database community, the diversification of query answers has been on the agenda for over a decade: the computation of diverse query results was studied in [33, 34] for relational data and in [21] for XML data. In [35], a system was presented with an extension of SQL to allow for requesting diverse answers. In [9], query result diversification is studied as a "bi-criteria" optimization problem that aims at finding k query answers that maximize both, the diversity and the relevance of the answers. Recent publications witness the renewed interest in the diversity of query answers by the database systems [18, 25] and the database theory community [3, 22].

Measuring diversity. In [17], a whole framework for dealing with the diversity of subsets of the solutions has been proposed. There, diversity is defined by first defining the distance between two solutions and then combining the pairwise distances via an aggregate function such as, for instance, sum, min, or max. In [36], a more sophisticated way of aggregating pairwise distances has led to the definition of the diversity function δ_{W}, which we have had a closer look at in our work. Yet more complexity was introduced in [29], where the diversity of a subset of solutions not only takes the relationships between the chosen solutions into account but also their relationship with the solutions excluded from the subset.

For the distance between two solutions, any metric can be used. As is argued in [36], it ultimately depends on the application context which distance function (and, consequently, which diversity function) is appropriate. Note that diversity and similarity can be seen as two sides of the same medal. Hence, all kinds of similarity measures studied in the data mining and information retrieval communities are, in principle, also candidates for the distance function; e.g., the Minkowski distance with Manhattan and Euclidean distance as important special cases as well as Cosine distance when solutions are represented as vectors, the edit distance for solutions as strings, or the Jaccard Index for solutions as sets, etc., see e.g., [11, 38].

Ultrametrics. The study of ultrametrics started in various areas of mathematics (such as real analysis, number theory, and general topology – see [20]) in the early 20th century. Ultrametrics are particularly well suited for hierarchical clustering and, as such, they have many applications in various sciences such as psychology, physics, and biology (see, e.g., [19, 24, 36]) and, of course, also in data mining, see e.g., [30]. Ultrametrics have also been used in database research and related areas: in [16], several convergence criteria for the fixed-point iteration of datalog programs (or, more generally, logic programs) with negation are defined. To this end, the set of possible ground atoms is divided into *levels* and two sets of ground atoms are considered as more diverse if they differ on an earlier level. Clearly, this is an ultrametric. In [33, 34], the distance between tuples is defined by imposing an order on the attributes and considering two tuples as more diverse if they differ on an earlier attribute in this ordering (see also Example 4.1 in the current paper). Again, this is clearly an ultrametric, even though it was not explicitly named as such in [33, 34].

Computing diversity. It should be noted that searching for diverse sets is, in general, an intractable problem. For instance, in [22], NP-completeness of the `DiversityExplicit`$[\delta]$ problem[3] was

[3]Strictly speaking, in [22], S was defined as the result set of an FO-query. However, since data complexity was considered, we can of course compute S upfront in polynomial time and may, therefore, assume S to be explicitly given.

proved even for the simple setting where S is a set of tuples of arity five and defining the diversity via the sum or min of the pairwise Hamming distances. Consequently, approximations or heuristics are typically proposed to compute diverse sets (see [37] for a very recent survey on diversification methods). In [6], the parameterized version of the $\texttt{DiversityExplicit}[\delta]$ problem was studied for cases where the problem of deciding the existence of a solution is fixed-parameter tractable (FPT) w.r.t. the treewidth. As a prototypical problem, the Vertex Cover problem was studied and it was shown that, when defining the diversity as the sum of the pairwise Hamming distances, then the $\texttt{DiversityExplicit}[\delta]$ problem is FPT w.r.t. the treewidth w and the size k of the desired diversity set. Moreover, it was argued in [6] that analogous FPT-results for the $\texttt{DiversityExplicit}[\delta]$ problem apply to virtually any problem where the decision of the existence of a solution is FPT w.r.t. the treewidth. In [22], the $\texttt{DiversityExplicit}[\delta]$ problem was shown FPT w.r.t. the size k of the desired diversity set, when considering S as a set of tuples and defining the diversity via a monotone aggregate function over the pairwise Hamming distances of the tuples.

However, *tractable*, *exact* methods for computing diverse sets are largely missing with one notable exception: in [34], an efficient method for solving the $\texttt{DiversityExplicit}[\delta]$ problem in a very specific setting is presented, where the diversity δ is defined as the sum over the ultrametric defined via an ordering of the attributes as recalled in Example 4.1. Assuming the existence of a tree representation of the relation S in the style of a Dewey tree known from XML query processing [32], the algorithm in [34] finds a k-diverse set in $O(k)$ time. Other than that, the field of tractable diversity computation is wide open and the main goal of this work is to fill this gap.

9 Conclusions

In this work, we have studied the complexity of 3 levels of *diversity* problems. For the most basic problem $\texttt{DiversityComputation}[\delta]$ of computing the diversity $\delta(S)$ for a given set S of elements, we have closed a problem left open in [36] by proving intractability of this problem in case of the Weitzman diversity measure δ_{W}. We have then pinpointed the boundary between tractability and intractability for both, the $\texttt{DiversityExplicit}[\delta]$ and the $\texttt{DiversityImplicit}[\delta]$ problems (i.e., the problems of maximizing the diversity of an explicitly or implicitly given set S, respectively) in terms of monotonicity properties of the diversity function δ extending an ultrametric. In particular, this has allowed us to identify tractable cases of the $\texttt{DiversityImplicit}[\delta]$ problem when considering acyclic conjunctive queries.

There are several natural directions of generalizing our results: clearly, they are naturally extended to more general query classes than acyclic CQs such as CQs with bounded (generalized or fractional) hypertree-width [13]. Less obvious is the extension of our tractability results to more general ultrametrics, which – in addition to determining the first attribute (in a given order) *where* two tuples differ – also introduce a measure *by how much* (again expressed as an ultrametric) the tuples differ in that attribute. Other directions of future work are concerned with studying other query languages over other data models such as (possibly restricted forms of) RPQs over graph data.

Acknowledgements

The work of Merkl and Pichler was funded by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT2201]. The work of Arenas and Riveros was funded by ANID – Millennium Science Initiative Program – Code ICN17002. The work of Riveros was also funded by ANID Fondecyt Regular project 1230935. Part of this work was done when Arenas, Merkl and Pichler were visiting the Simons Institute for the Theory of Computing.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. In G. C. Bongiovanni, D. P. Bovet, and G. D. Battista, editors, *Algorithms and Complexity, Third Italian Conference, CIAC '97, Rome, Italy, March 12-14, 1997, Proceedings*, volume 1203 of *Lecture Notes in Computer Science*, pages 288–298. Springer, 1997.

[3] M. Arenas, L. A. Croquevielle, R. Jayaram, and C. Riveros. #NFA admits an FPRAS: efficient enumeration, counting, and uniform generation for logspace classes. *J. ACM*, 68(6):48:1–48:40, 2021.

[4] M. Arenas, T. C. Merkl, R. Pichler, and C. Riveros. Towards tractability of the diversity of query answers: Ultrametrics to the rescue. *CoRR*, abs/2408.01657, 2024.

[5] G. Bagan, A. Durand, and E. Grandjean. On acyclic conjunctive queries and constant delay enumeration. In J. Duparc and T. A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 2007.

[6] J. Baste, M. R. Fellows, L. Jaffke, T. Masarík, M. de Oliveira Oliveira, G. Philip, and F. A. Rosamond. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artif. Intell.*, 303:103644, 2022.

[7] N. Carmeli, N. Tziavelis, W. Gatterbauer, B. Kimelfeld, and M. Riedewald. Tractable orders for direct access to ranked answers of conjunctive queries. *ACM Trans. Database Syst.*, 48(1):1:1–1:45, 2023.

[8] E. Demidova, P. Fankhauser, X. Zhou, and W. Nejdl. *DivQ*: diversification for keyword search over structured databases. In F. Crestani, S. Marchand-Maillet, H. Chen, E. N. Efthimiadis, and J. Savoy, editors, *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 331–338. ACM, 2010.

[9] T. Deng and W. Fan. On the complexity of query result diversification. *ACM Trans. Database Syst.*, 39(2):15:1–15:46, 2014.

[10] T. Eiter, E. Erdem, H. Erdogan, and M. Fink. Finding similar/diverse solutions in answer set programming. *Theory Pract. Log. Program.*, 13(3):303–359, 2013.

[11] G. Gan, C. Ma, and J. Wu. *Data Clustering: Theory, Algorithms, and Applications*, volume 20 of *ASA-SIAM Series on Statistics and Applied Probability*. SIAM, 2007.

[12] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In J. Quemada, G. León, Y. S. Maarek, and W. Nejdl, editors, *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 381–390. ACM, 2009.

[13] G. Gottlob, G. Greco, N. Leone, and F. Scarcello. Hypertree decompositions: Questions and answers. In T. Milo and W. Tan, editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 57–74. ACM, 2016.

[14] E. Hebrard, B. Hnich, B. O'Sullivan, and T. Walsh. Finding diverse and similar solutions in constraint programming. In M. M. Veloso and S. Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 372–377. AAAI Press / The MIT Press, 2005.

[15] E. Hebrard, B. O'Sullivan, and T. Walsh. Distance constraints in constraint satisfaction. In M. M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 106–111, 2007.

[16] P. Hitzler and A. K. Seda. Generalized metrics and uniquely determined logic programs. *Theor. Comput. Sci.*, 305(1-3):187–219, 2003.

[17] L. Ingmar, M. G. de la Banda, P. J. Stuckey, and G. Tack. Modelling diversity of solutions. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1528–1535. AAAI Press, 2020.

[18] M. M. Islam, M. Asadi, S. Amer-Yahia, and S. B. Roy. A generic framework for efficient computation of top-k diverse results. *VLDB J.*, 32(4):737–761, 2023.

[19] S. V. Kozyrev. Methods and applications of ultrametric and p-adic analysis: From wavelet theory to biophysics. *Proceedings of the Steklov Institute of Mathematics*, 274(1):1–84, 2011.

[20] A. J. Lemin. On ultrametrization of general metric spaces. *Proceedings of the American Mathematical Society*, 131(3):979–989, 2001.

[21] Z. Liu, P. Sun, and Y. Chen. Structured search result differentiation. *Proc. VLDB Endow.*, 2(1):313–324, 2009.

[22] T. C. Merkl, R. Pichler, and S. Skritek. Diversity of answers to conjunctive queries. In F. Geerts and B. Vandevoort, editors, *26th International Conference on Database Theory, ICDT 2023, March 28-31, 2023, Ioannina, Greece*, volume 255 of *LIPIcs*, pages 10:1–10:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[23] T. C. Merkl, R. Pichler, and S. Skritek. Diversity of answers to conjunctive queries. *CoRR*, abs/2301.08848, 2023.

[24] F. Murtagh. Ultrametric model of mind, ii: Application to text content analysis. *p-Adic Numbers, Ultrametric Analysis and Applications*, 4(3):207–221, 2012.

[25] S. Nikookar, M. Esfandiari, R. M. Borromeo, P. Sakharkar, S. Amer-Yahia, and S. B. Roy. Diversifying recommendations on sequences of sets. *VLDB J.*, 32(2):283–304, 2023.

[26] T. Petit and A. C. Trapp. Finding diverse solutions of high quality to constraint optimization problems. In Q. Yang and M. J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 260–267. AAAI Press, 2015.

[27] R. L. T. Santos, C. Macdonald, and I. Ounis. Exploiting query reformulations for web search result diversification. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 881–890. ACM, 2010.

[28] R. L. T. Santos, C. MacDonald, and I. Ounis. Search result diversification. *Found. Trends Inf. Retr.*, 9(1):1–90, 2015.

[29] N. Schwind, T. Okimoto, M. Clement, and K. Inoue. Representative solutions for multi-objective constraint optimization problems. In C. Baral, J. P. Delgrande, and F. Wolter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*, pages 601–604. AAAI Press, 2016.

[30] D. A. Simovici. Data mining algorithms i: Clustering. In A. Nayak and I. Stojmenovic, editors, *Handbook of Applied Algorithms: Solving Scientific, Engineering and Practical Problems*, pages 10:1–10:19. Wiley-IEEE Press, 2007.

[31] Z. Su, Z. Dou, Y. Zhu, and J. Wen. Knowledge enhanced search result diversification. In A. Zhang and H. Rangwala, editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 1687–1695. ACM, 2022.

[32] I. Tatarinov, S. Viglas, K. S. Beyer, J. Shanmugasundaram, E. J. Shekita, and C. Zhang. Storing and querying ordered XML using a relational database system. In M. J. Franklin, B. Moon, and A. Ailamaki, editors, *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002*, pages 204–215. ACM, 2002.

[33] E. Vee, J. Shanmugasundaram, and S. Amer-Yahia. Efficient computation of diverse query results. *IEEE Data Eng. Bull.*, 32(4):57–64, 2009.

[34] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. Amer-Yahia. Efficient computation of diverse query results. In G. Alonso, J. A. Blakeley, and A. L. P. Chen, editors, *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*, pages 228–236. IEEE Computer Society, 2008.

[35] M. R. Vieira, H. L. Razente, M. C. N. Barioni, M. Hadjieleftheriou, D. Srivastava, C. T. Jr., and V. J. Tsotras. Divdb: A system for diversifying query results. *Proc. VLDB Endow.*, 4(12):1395–1398, 2011.

[36] M. L. Weitzman. On diversity. *The quarterly journal of economics*, 107(2):363–405, 1992.

[37] H. Wu, Y. Zhang, C. Ma, F. Lyu, X. Liu, B. He, B. Mitra, and X. Liu. Result diversification in search and recommendation: A survey. *IEEE Trans. Knowl. Data Eng. (Early Access)*, 2024.

[38] D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Ann. Data. Sci.*, 2(2):165–193, 2015.

[39] M. Yannakakis. Algorithms for acyclic database schemes. In *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings*, pages 82–94. IEEE Computer Society, 1981.

[40] K. Zheng, H. Wang, Z. Qi, J. Li, and H. Gao. A survey of query result diversification. *Knowl. Inf. Syst.*, 51(1):1–36, 2017.

A Additional Details for Section 5

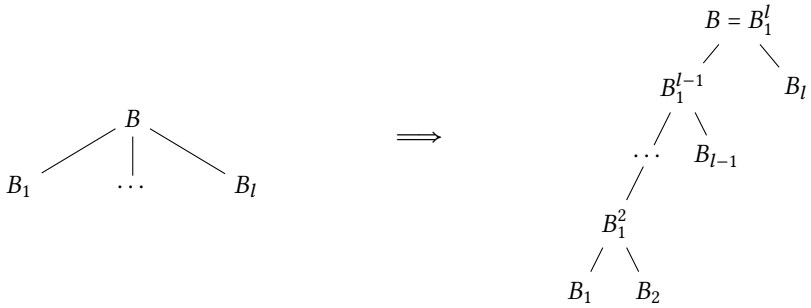
Proof of Theorem 5.4

Proof. Let S be a finite subset of the universe $\mathcal{U}$, $\mathbf{u}$ an ultrametric over $\mathcal{U}$, and δ a subset-monotone diversity function of $\mathbf{u}$. By Proposition 3, we can construct an ultrametric tree $\mathcal{T}_S$ of S in time $O(|S|^2)$. For each vertex B (i.e., a ball) of $\mathcal{T}_S$, we maintain a function $\mathrm{C}_B\colon \{0,\dots,\min\{k,|B|\}\} \to 2^B$ where $\mathrm{C}_B(i)$ is a *candidate* diverse subset of B of size i. Formally, for every $i \in \{0,\dots,\min\{k,|B|\}\}$ we define:

$$\mathrm{C}_B(i) \;:=\; \operatorname*{arg\,max}_{A\subseteq B:|A|=i} \delta(A).$$

Clearly, if we can compute C_S for the root S of $\mathcal{T}_S$, then $\mathrm{C}_S(k)$ is a diverse subset of S of size k. We can compute these functions C_B in polynomial time using a dynamic programming approach. To that end, we compute C_B for each $B \in \mathcal{B}_S$ in a bottom-up fashion over $\mathcal{T}_S$. For every ball B, we can easily check that $\mathrm{C}_B(0) = \varnothing$ and $\mathrm{C}_B(1) = \{a\}$ for some $a \in B$. In particular, $\mathrm{C}_{\{a\}} = \{0 \mapsto \varnothing, 1 \mapsto \{a\}\}$ is our base case for every leaf $\{a\}$ of $\mathcal{T}_S$. For an inner vertex B of $\mathcal{T}_S$, the process is a bit more involved. Let $B_1,\dots,B_l$ the children of B in $\mathcal{T}_S$ and assume that we already computed $\mathrm{C}_{B_1},\dots,\mathrm{C}_{B_l}$.

We can now construct (see below) a binary tree T_B with vertices $B_1,\dots,B_l, B_1^2 := \bigcup_{i=1}^2 B_i,\dots, B_1^l := \bigcup_{i=1}^l B_i = B$ and edges from $B_1^m = \bigcup_{i=1}^m B_i$ to $B_1^{m-1} = \bigcup_{i=1}^{m-1} B_i$ and B_m for $1 < m \le l$. Thus, $B_1,\dots,B_l$ are the leaves and B is the root of T_B.



To construct C_B, we first construct intermediate results $\mathrm{C}_{B_1^m}$ with

$$\mathrm{C}_{B_1^m}(i) \;:=\; \operatorname*{arg\,max}_{A\subseteq B_1^m:|A|=i} \delta(A).$$

We claim that, for every $i \in \{0,\dots,\min\{k,|B_1^m|\}\}$, we can calculate $\mathrm{C}_{B_1^m}(i)$ as $\mathrm{C}_{B_1^m}(i) = \mathrm{C}_{B_1^{m-1}}(i_1) \cup \mathrm{C}_{B_m}(i_2)$ where:

$$(i_1,i_2) = \operatorname*{arg\,max}_{(j_1,j_2):j_1+j_2=i} \delta(\mathrm{C}_{B_1^{m-1}}(j_1) \cup \mathrm{C}_{B_m}(j_2)).$$

To see that the claim holds, let A be a subset of B_1^m with i-elements maximizing $\delta(A)$. Define $A_1 := A \cap B_1^{m-1}$ and $A_2 := A \cap B_m$, and their sizes $i_1 := |A_1|$ and $i_2 := |A_2|$, respectively. Due to the correctness of $\mathrm{C}_{B_1^{m-1}}$, we know that $\delta(A_1) \le \delta(\mathrm{C}_{B_1^{m-1}}(i_1))$. Further, $\mathbf{u}(a_1,a_2) = \mathbf{u}(a_1',a_2) = \mathrm{r}_S(B)$ for every $a_1 \in A_1, a_1' \in \mathrm{C}_{B_1^{m-1}}(i_1), a_2 \in A_2$ by Property 2. Then the conditions of weak subset-monotonicity are satisfied and $\delta(A_1 \cup A_2) \le \delta(\mathrm{C}_{B_1^{m-1}}(i_1) \cup A_2)$. Following the same argument, we can conclude that $\delta(\mathrm{C}_{B_1^{m-1}}(i_1) \cup A_2) \le \delta(\mathrm{C}_{B_1^{m-1}}(i_1) \cup \mathrm{C}_{B_m}(i_2))$, proving that $\mathrm{C}_{B_1^{m-1}}(i_1) \cup \mathrm{C}_{B_m}(i_2)$ is optimal.

Thus, given $\mathrm{C}_{B_1^{m-1}}$ and C_{B_m}, computing $\mathrm{C}_{B_1^m}$ takes time $O(k^2 \cdot f_\delta(k))$. Consequently, given $\mathrm{C}_{B_1},\dots,\mathrm{C}_{B_l}$, computing C_B requires time $O(k^2 \cdot f_\delta(k) \cdot l)$. In total, given the ultrametric tree $\mathcal{T}_S$ and proceeding bottom-up, we can compute C_S in time $O(k^2 \cdot f_\delta(k) \cdot |S|)$. □

B Additional Details for Section 6

Proof of Theorem 6.1

Proof. It remains to prove the correctness of Algorithm 1. We proceed in three steps. For the first step, we prove that, at the beginning of each iteration (line 6), S and L cover all solutions $[\![I]\!]$, i.e., $S \cup \bigcup_{B \in L} B = [\![I]\!]$. We show this by induction on the number of iterations. This is certainly true before the first iteration, since $B_{\texttt{root}}$ is the root of the ultrametric tree and $B_{\texttt{root}} = [\![I]\!]$. For any iteration, S only grows while L only changes if $\texttt{Children}(I, B).\texttt{next} = \texttt{false}$ (line 9). Then, the algorithm removes B from L and adds all $B' \in \texttt{Children}(I, B)$ to L for which $|B'| > 1$. Note that, if $|B'| = 1$, then $B' \subseteq S$ as at least one element $a \in B'$ of each child B' of B was added to S. Given that $\texttt{Children}(I, B)$ forms a partition of B, we can assert that $B \subseteq S \cup \bigcup_{B' \in \texttt{Children}(I,B), |B'|>1} B'$. We conclude that $S \cup \bigcup_{B \in L} B = [\![I]\!]$ still holds.

For the second step, we show that, at the beginning of each iteration, for any $a \in [\![I]\!] \setminus S$, there exists $B \in L$ such that $\delta(S \cup \{a\}) \leq \delta(S \cup \{b\})$ for $b = \texttt{Member}(I, \texttt{Children}(I, B).\texttt{current})$. In other words, in steps 7 and 8 the algorithm chooses an element that maximizes the incremental diversity of S. To prove this, take any $a \in [\![I]\!] \setminus S$. By the first step, there must exists $B \in L$ such that $a \in B$. Let $B' = \texttt{Children}(I, B).\texttt{current}$ and $b = \texttt{Member}(I, B')$. On the one hand, for every $s \in S \cap B$, it holds that $\mathbf{u}(s, a) \leq r(B)$. Given that $S \cap B' = \varnothing$ (i.e., B' has not been considered yet), $\mathbf{u}(s, b) = r(B)$ (by Property 2). In particular, $\mathbf{u}(s, a) \leq \mathbf{u}(s, b)$. On the other hand, for every $s \in S \setminus B$, it holds that $\mathbf{u}(s, a) = \mathbf{u}(s, b)$ given that $a, b \in B$. Combining both cases, we have that $\mathbf{u}(s, a) \leq \mathbf{u}(s, b)$ for every $s \in S$. Now, if we choose $A = S$, $B = \{a\}$, and $B' = \{b\}$, we conclude by subset-monotonicity that $\delta(S \cup \{a\}) \leq \delta(S \cup \{b\})$.

For the last step, we prove that the algorithm always outputs a k-diverse set with respect to δ. Let $S = \{s_1, \ldots, s_k\}$ be the output of the algorithm where $s_1, \ldots, s_k$ is the order how the algorithm added the elements to S. Towards a contradiction, assume that there exists $S' \subseteq [\![I]\!]$ of size k such that $\delta(S) < \delta(S')$. Let M be the set of all S' such that $\delta(S) < \delta(S')$ and $|S'| = k$. Pick one $S^* \in M$ that contains the longest prefix of $s_1, \ldots, s_k$, namely, $S^* = \arg\max_{S' \in M} \{m \mid s_1, \ldots, s_m \in S'\}$. Then $s_1, \ldots, s_m \in S^*$ but $s_{m+1} \notin S^*$. Also, let $s^* \in S^*$ be such that $\mathbf{u}(s_{m+1}, S^* \setminus \{s_1, \ldots, s_m\}) = \mathbf{u}(s_{m+1}, s^*)$ (i.e., s^* is one of the closest elements to s_{m+1} in $S^* \setminus \{s_1, \ldots, s_m\}$). Define $A = S^* \setminus \{s_1, \ldots, s_m, s^*\}$, $B = \{s_1, \ldots, s_m, s^*\}$, and $B' = \{s_1, \ldots, s_m, s_{m+1}\}$. By the second step, we know that $\delta(B) \leq \delta(B')$. Furthermore, for every $a \in A$ we have that:

$$\mathbf{u}(a, s^*) \leq \max\{\mathbf{u}(a, s_{m+1}), \mathbf{u}(s_{m+1}, s^*)\} = \max\{\mathbf{u}(a, s_{m+1}), \mathbf{u}(s_{m+1}, S^* \setminus \{s_1, \ldots, s_m\})\} = \mathbf{u}(a, s_{m+1}).$$

The remaining elements $B \setminus \{s^*\}$ are the same as $B' \setminus \{s_{m+1}\}$ and $\mathbf{u}(a, s_i) = \mathbf{u}(a, s_i)$. Then, applying subset-monotonicity, we get that:

$$\delta(S^*) = \delta(A \cup B) \leq \delta(A \cup B')$$

This means that $A \cup B' \in M$ but $A \cup B'$ has a longer prefix of $s_1, \ldots, s_k$ than S^*, which is a contradiction. We conclude that the output S of Algorithm 1 is a k-diverse set of $[\![I]\!]$ with respect to δ. □

Proof of Theorem 6.5

Proof. It remains to prove the correctness and the FPT running time of Algorithm 2. To that end, we verify the following condition for any $I \in \mathcal{I}, B \in \mathcal{B}_{[\![I]\!]}, k' \leq k$ by induction on k': Let $S = \mathit{RelevantElements}(I, B, k')$. For any k-subset $S' \subseteq [\![I]\!]$ with $|S' \cap B| =: m \leq k'$, there exists a m-subset $A \subseteq S$ such that $\delta(S') \leq \delta(S' \setminus B \cup A)$ (‡).

For $|B| = 1$ we have $S = \mathit{RelevantElements}(I, B, k') = B$ and thus we can simply select $A := B$.

For $k' = 1$ we have $S = \mathit{RelevantElements}(I, B, k') = \{a\}$ where $a = \texttt{Member}(I, B)$. Then, let $S' \subseteq [\![I]\!]$ be as required. If $m = 0$ we can again simply select $A := \varnothing$. Thus, assume $m = 1$ and let b be

such that $S' \cap B = \{b\}$. Now consider $A = \{a\}$. For any $s \in S' \setminus B = S' \setminus \{b\}$ we have

$$\mathbf{u}(b,s) \leq \max\{\mathbf{u}(b,a), \mathbf{u}(a,s)\} = \mathbf{u}(a,s)$$

since a, b are in the same ball $B \in \mathcal{B}_{[\![I]\!]}$ but s is not in B. Thus, due to incremental monotonicity, $\delta(S') \leq \delta(S' \setminus B \cup A)$

Now consider a $I \in \mathcal{I}, B \in \mathcal{B}_{[\![I]\!]}, 1 < |B|, 1 < k' \leq k$ and assume ($\ddagger$) holds for any $k'' < k'$. Let $B_1, \ldots, B_l$ be the children of B. Furthermore, let $S' \subseteq [\![I]\!]$ be as required. We define $S'_I := S' \setminus B$ and $S'_B := S' \cap B$. Let B_i be a child of B such that $B_i \cap S'_B = \emptyset$. Then, for any $a \in B_i, b \in S'_B$ and $s \in S' \setminus \{b\}$, again

$$\mathbf{u}(b,s) \leq \max\{\mathbf{u}(b,a), \mathbf{u}(a,s)\} = \mathbf{u}(a,s).$$

Thus, $\delta(S') \leq \delta(S' \setminus \{b\} \cup \{a\})$. It suffices to show Condition ($\ddagger$) for $S' \setminus \{b\} \cup \{a\}$ (playing the role of S') as this is strictly harder to achieve. Thus, we can require, w.l.o.g., $S' \cap B_1 \neq \emptyset, \ldots, S' \cap B_{\min\{l,m\}} \neq \emptyset$. Furthermore, $|S' \cap B_i| \leq m - \min\{l,m\} + 1 \leq k' - \min\{l,k'\} + 1 < k'$ for all $i \leq \min\{l,m\}$. Thus, by the Condition ($\ddagger$), there exist $(|S' \cap B_i|)$-subsets $A_i \subseteq \mathit{RelevantElements}(I, B_i, k' - \min\{l,k'\} + 1)$ for which

$$\delta(S' \setminus B \cup \bigcup_{j<i} A_j \cup \bigcup_{i \leq j} (S' \cap B_j)) \leq \delta(S' \setminus B \cup \bigcup_{j \leq i} A_j \cup \bigcup_{i<j} (S' \cap B_j)).$$

Applying this from $i = 1$ to $i = \min\{l,m\}$ and defining

$$A := \bigcup_{i=1}^{\min\{l,m\}} A_i \subseteq \bigcup_{i=1}^{\min\{l,m\}} \mathit{RelevantElements}(I, B_i, k' - \min\{l,k'\} + 1) = \mathit{RelevantElements}(I, B, k')$$

this gives us

$$\delta(S') = \delta(S' \setminus B \cup \bigcup_{1 \leq j} (S' \cap B_j)) \leq \cdots \leq \delta(S' \setminus B \cup \bigcup_{j \leq \min\{l,m\}} A_j) = \delta(S' \setminus B \cup A)$$

as required.

We can conclude that Condition ($\ddagger$) holds for $B = [\![I]\!]$ and $k' = k$. Thus, for any k-subset $A \subseteq [\![I]\!]$, there exists a k-subset $S' \subseteq S := \mathit{RelevantElements}(I, [\![I]\!], k)$ such that $\delta(A) \leq \delta(S')$. Consequently, S contains at least one k-diverse subset of $[\![I]\!]$. □

C Additional Details for Section 7

Proof of Theorem 7.1

Proof. We want to apply Theorem 6.1 and, therefore, we need to give an implicit ultrametric tree $(\texttt{Root}, \texttt{Children}, \texttt{Member})$ which runs in time $O(|Q| \cdot |D| \cdot \log(|D|))$. To that end, we first define IDs for the balls $\mathcal{B}_{[\![Q]\!](D)}$ which will be subsequently used by the implicit ultrametric tree. Concretely, for every ball $B \in \mathcal{B}_{[\![Q]\!](D)}$, there exists values $c_1, \ldots, c_i$ such that $B = \{Q(\bar{a}) \in [\![Q]\!](D) \mid \forall j \leq i.\, \bar{a}[j] = c_j\}$. This means that all answers in B agree on the values $c_1, \ldots, c_i$ and these form a common prefix. Moreover, these prefixes are as long as possible, i.e., i is as big as possible, and we can use $c_1, \ldots, c_i$ to uniquely identify B.

To implement the methods `Root`, `Children`, and `Member`, we modify Yannakakis algorithm [39]. Recall that Yannakakis algorithm proceeds in the following manner (we only sketch the preprocessing phase) on an ACQ $Q(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)$:

(1) The R_i are arranged in a tree structure (in a join tree).
(2) Each R_i gets assigned a unique copy R_i^D of the corresponding table in D.
(3) The R_i^D are semijoined as to delete dangling tuples.

Thus, after the preprocessing phase, $R_i(h(\bar{x}_i)) \in R_i^D$ for some $h_i\colon \bar{x}_i \to \mathbb{D}$ if and only if h_i can be extended to a $h\colon \mathcal{X} \to \mathbb{D}$ such that $Q(h(\bar{x})) \in [\![Q]\!](D)$. Thus, the admissible values

$$ad(x) := \{d \in \mathbb{D} \mid \exists h\colon \mathcal{X} \to \mathbb{D} \text{ s.t., } h(x) = d \text{ and } Q(h(\bar{x})) \in [\![Q]\!](D)\}$$

of a variable $x \in \mathcal{X}$ can be compute in time $O(|D|)$ by inspecting a table R_i^D where x appears in $\bar{x}_i$. Therefore, to compute the common prefix of $B_{\texttt{root}} := [\![Q]\!](D)$, we simply have to iteratively go through $\bar{x}[1], \ldots, \bar{x}[|\bar{x}|]$ to find the first $\bar{x}[i]$ such that $|ad(\bar{x}[i])| \neq 1$. The common prefix of $B_{\texttt{root}}$ then is $c_1, \ldots, c_{i-1}$ where $\{c_1\} = ad(\bar{x}[1]), \ldots, \{c_{i-1}\} = ad(\bar{x}[i-1])$ (for $i = 1$ the common prefix is the empty prefix ϵ). Computing this prefix takes time $O(|Q| \cdot |D| \cdot \log(|D|))$ and is exactly what `Root` does.

To compute the children of a ball $B \in \mathcal{B}_{[\![Q]\!](D)}$ with common prefix $\bar{c} = (c_1, \ldots, c_{i-1})$ we can do the following: Consider the query $Q'(\bar{x}[i], \ldots, \bar{x}[|\bar{x}|]) \leftarrow R_1(h(\bar{x}_1)), \ldots, R_m(h(\bar{x}_m))$ where h is an partial assignment that maps $h(\bar{x}[j]) = c_j$ for every $j \leq i-1$ and $h(x) = x$ for any other variable $x \in \mathcal{X}$. I.e., we plugged the prefix $\bar{c}$ into the query Q. We can then compute the admissible values of the next variable $\bar{x}[i]$ for the prefix $\bar{c}$, i.e., the set

$$ad_{\bar{c}}(\bar{x}[i]) := \{d \in \mathbb{D} \mid \exists h\colon \mathcal{X} \to \mathbb{D} \text{ s.t., } h(\bar{x}[i]) = d \text{ and } Q'(h(\bar{x}[i], \ldots, \bar{x}[|\bar{x}|])) \in [\![Q']\!](D)\}.$$

This may take time up to $O(|Q| \cdot |D| \cdot \log(|D|))$. Note that $|ad_{\bar{c}}(\bar{x}[i])| \geq 1$ as otherwise the prefix of B would have been longer. Then, to enumerate the children we iterate through $c_i \in ad_{\bar{c}}(\bar{x}[i])$. For a c_i, we plug in the new prefix $(\bar{c}, c_i)$ into Q which results in the query $Q'_{c_i}(\bar{x}[i+1], \ldots, \bar{x}[|\bar{x}|]) \leftarrow R_1(h(\bar{x}_1)), \ldots, R_m(h(\bar{x}_m))$. Then, let B_{c_i} be the answers $[\![Q'_{c_i}]\!](D)$ prepended by the prefix $(\bar{c}, c_i)$. Note that B_{c_i} is a child of B and we can compute the prefix of it as the prefix $\bar{c}'_{c_i}$ of $[\![Q'_{c_i}]\!](D)$ prepended by the prefix $(\bar{c}, c_i)$, i.e., it is $(\bar{c}, c_i, \bar{c}'_{c_i})$ ($\bar{c}'_{c_i}$ may be the empty prefix). All of this takes time $O(|Q| \cdot |D| \cdot \log(|D|))$ for each c_i and, thus, this is also the delay we get for `Children`.

Lastly, given a ball $B \in \mathcal{B}_{[\![Q]\!](D)}$ with common prefix $\bar{c} = (c_1, \ldots, c_{i-1})$ we can easily compute a $b \in B$ by computing any answer $a \in [\![Q']\!](D)$ with Yannakakis algorithm and prepend it with $\bar{c}$. Thus, `Member` also only requires time $O(|Q| \cdot |D| \cdot \log(|D|))$. □

Proof of Theorem 7.2

Proof. We proceed similar to the proof of Theorem 7.1 but in the absence of a disruptive trio we can move to an extension of Yannakakis algorithm developed in [7] which takes the order of the head variables $\bar{x}$ in consideration. Intuitively, the absence of a disruptive trio ensures the existence of a *layered join tree* whose layers follow the order of the variables in the head of Q. This will allow us to improve the runtime of `Root`, `Children` and `Member` to $O(|Q|)$ if we allow a common preprocessing of $O(|Q| \cdot |D| \cdot \log(|D|))$. Thus, be inspecting how we arrive at the run time in Theorem 6.1 this then totals to the run time as required.

We start by recalling the steps taken by the algorithm presented in [7] (we only sketch the preprocessing phase) on a free-connex ACQ $Q(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)$ without a disruptive trio:

(1) Projections of some $R_i(\bar{x}_i)$ are possibly added to the query such that the resulting query has a layered join tree. However, the semantics of the query remains unchanged and, thus, we assume that Q already includes all of these projections needed.
(2) The R_i are arranged in a rooted tree structure T (a layered join tree). As Q is free-connex we assume that the free variables $\bar{x}$ appear in a subtree which includes the root.
(3) Each R_i gets assigned a unique copy R_i^D of the corresponding table in D.
(4) The R_i^D are semijoined as to delete dangling tuples. Furthermore, for each R_i with child R_j, indexes are created such that we can access the join partners $t_j \in R_j^D$ of each $t_i \in R_i^D$ in constant time and with constant delay.

(5) As Q is free-connex, we can now remove the bounded variables. Thus, w.l.o.g., we may assume Q to be a full CQ, i.e., all $\bar{x}_i$ are all sequences of variables in $\bar{x}$.

All of this preprocessing only requires $O(|Q| \cdot |D| \cdot \log(|D|))$ time. Furthermore, due to the layered join tree we can assume, w.l.o.g., that R_i is the root of the subtree of T containing the variable $\bar{x}[i]$ and if R_i is the parent of R_j that $i < j$. Also, to simplify the subsequent presentation we assume the variables in $\bar{x}_i$ to be ordered according to $\bar{x}$ and that variables are not repeated within $\bar{x}_i$ nor $\bar{x}$.

Now, we can proceed to define the algorithms $(\texttt{Root}, \texttt{Children}, \texttt{Member})$ similar to the proof of Theorem 7.1 but it will no longer be necessary to recompute join trees and we can always remain in T. To that end, recall the definition of the prefix of a ball $B \in \mathcal{B}_{[\![Q]\!](D)}$ as the values $c_1, \ldots, c_i$ such that $B = \{Q(\bar{a}) \in [\![Q]\!](D) \mid \forall j \leq i.\, \bar{a}[j] = c_j\}$, and the admissible values for a variable $x \in \mathcal{X}$, i.e.,

$$ad(x) := \{d \in \mathbb{D} \mid \exists h\colon \mathcal{X} \to \mathbb{D} \text{ s.t., } h(x) = d \text{ and } Q(h(\bar{x})) \in [\![Q]\!](D)\}.$$

Furthermore, we also define the admissible values of a variable $x \in \mathcal{X}$ given a prefix $\bar{c} = (c_1, \ldots, c_{i-1})$. Slightly different to before, we define

$$ad_{\bar{c}}(\bar{x}[i]) := \{d \in \mathbb{D} \mid \exists h\colon \mathcal{X} \to \mathbb{D} \text{ s.t., } h((\bar{x}[1], \ldots, \bar{x}[i-1])) = \bar{c}, h(x[i]) = d \text{ and } Q(h(\bar{x})) \in [\![Q]\!](D)\}.$$

Note that computing $ad(\bar{x}[1])$ is easy as $\bar{x}_1$ can only contain the variable $\bar{x}[1]$ due to the fact that R_1 is the root of T but only the root of the subtree containing the variable $\bar{x}[1]$. Thus, $R_1(\bar{x}[1])$ is part of the query and $ad(\bar{x}[1]) = R_1^D$. If $|R_1^D| = |ad(\bar{x}[1])| > 1$, the prefix of $B_{\texttt{root}} := [\![Q]\!](D)$ is the empty prefix ϵ and there is nothing more to do for $\texttt{Root}$. Otherwise, we proceed to $ad(\bar{x}[2])$.

To that end, let in general $\{c_1\} = ad(\bar{x}[1]), \ldots, \{c_{i-1}\} = ad(\bar{x}[i-1])$ and we are looking at whether $ad(\bar{x}[i])$ is of size 1 or greater than 1. To that end, consider R_i which is the root of the subtree of T containing the variable $\bar{x}[i]$. Hence, in Q, it may only appear together with variables $\bar{x}[j]$ with $j \leq i$. But we know all of them only have 1 admissible value, thus, R_i^D is the same as $ad(\bar{x}[i])$ where values c_j for the $x[j]$ which appear in $\bar{x}_i$ and where $j < i$ are prepended to $ad(\bar{x}[i])$. I.e., for $h\colon \{\bar{x}[1], \ldots, \bar{x}[i-1]\} \to \mathbb{D}, h(\bar{x}[l]) := c_l$

$$R_i^D = \{(h(\bar{x}_i \setminus \{\bar{x}[i]\}), d) \mid d \in ad(\bar{x}[i])\}.$$

Thus, if $|R_i^D| = |ad(\bar{x}[i])| > 1$, the prefix of $B_{\texttt{root}} := [\![Q]\!](D)$ is $(c_1, \ldots, c_{i-1})$ and there is nothing more to do for $\texttt{Root}$. Otherwise, we proceed to $ad(\bar{x}[i+1])$.

In total, applying the preprocessing as sketched above, $\texttt{Root}$ only requires $O(|Q|)$ time.

Now lets proceed to computing the children of a ball $B \in \mathcal{B}_{[\![Q]\!](D)}$. To that end, let $\bar{c} = (c_1, \ldots, c_{i-1})$ be the common prefix of B. We go to R_i and its parent R_j where $j < i$. Similar to before, we can determine $ad_{\bar{c}}(\bar{x}[i])$ by inspecting R_i. To that end, let $h\colon \{\bar{x}[1], \ldots, \bar{x}[i-1]\} \to \mathbb{D}, h(\bar{x}[l]) := c_l$. Then,

$$\{(h(\bar{x}_i \setminus \{\bar{x}[i]\}), d) \in R_i^D\} = \{(h(\bar{x}_i \setminus \{\bar{x}[i]\}), d) \mid d \in ad_{\bar{c}}(\bar{x}[i])\}.$$

The left hand side are the tuples of R_i^D that adhere to the prefix $\bar{c}$ while the right hand side are the admissible values of $\bar{x}[i]$ given the prefix $\bar{c}$ prepended by the values of $\bar{c}$ that correspond to variables appearing in $\bar{x}_i$. However, given the indexes on the parent R_j, we can compute the left hand side with constant delay. To see let consider $t_j := h(\bar{x}_j) \in R_j^D$ and notice that $\{(h(\bar{x}_i \setminus \{\bar{x}[i]\}), d) \in R_i^D\}$ are exactly the join partners of t_j in R_i^D. Thus, we can iterate through $c_i \in ad_{\bar{c}}(\bar{x}[i])$ with constant delay.

Now let $B_{c_i} \in \mathcal{B}_{[\![Q]\!](D)}$ be the answers with prefixes $(\bar{c}, c_i)$. Note that B_{c_i} is a child of B but we still have to extend this prefix to the maximal prefix for B_{c_i}. To that end, we have to inspect $ad_{(\bar{c},c_i)}(\bar{x}[i+1])$. However, we already know how to compute this by following the same argumentation as for $ad_{\bar{c}}(\bar{x}[i])$. If $|ad_{(\bar{c},c_i)}(\bar{x}[i+1])| > 1$ we stop and assert that $(\bar{c}, c_i)$ is the correct maximal prefix. Otherwise, we continue to a $l > 1$ such that $\{c_{i+1}\} = ad_{(\bar{c},c_i)}(\bar{x}[i+1]), \ldots, \{c_{i+l-1}\} =$

$ad_{(\bar{c},c_i)}(\bar{x}[i+l-1])$ and $|ad_{(\bar{c},c_i)}(\bar{x}[i+l])| > 1$. Then, $(\bar{c}, c_i, \ldots, c_{i+l-1})$ is the maximal prefix of B_{c_i}. In total, `Children` has a delay of at most $O(|Q|)$.

Lastly, given a ball $B \in \mathcal{B}_{[\![Q]\!](D)}$ with common prefix $\bar{c} = (c_1, \ldots, c_{i-1})$ we can easily compute a $b \in B$. To do that, we iterate through $R_i, \ldots, R_m$ in this order. Let $h\colon \{\bar{x}[1], \ldots, \bar{x}[i-1]\} \to \mathbb{D}, h(\bar{x}[l]) := c_l$. We process R_i with parent R_j by considering the tuple $t_j := h(\bar{x}_j) \in R_j^D$ and simply select the first join partner $t_i \in R_i$. Then, t_i assigns a value to $\bar{x}[i]$ which we call c_i. Now we can proceed to R_{i+1} with the prefix $(c_1, \ldots, c_i)$, i.e., the previous prefix extended by c_i. Also this process, i.e., `Member`, only requires $O(|Q|)$ time.

By reinspecting how we arrive at the run time in Theorem 6.1 – in particular what role the run times of (`Root`, `Children`, `Member`) play – this then totals to the run time as required. □

Proof of Theorem 7.3

Proof. Let us reconsider the proof of Theorem 7.2 and the definitions used there. Furthermore, let us consider the execution of Algorithm 1 using the implicit ultrametric tree developed in the proof of Theorem 7.2. To that end, let S, L be as they are at the start of some loop iteration, i.e., in line 6. Moreover, let $L = \{B_1, \ldots, B_l\}$ and let $\bar{p}_i$ be the prefix corresponding to the ball $B_i \in \mathcal{B}_{[\![Q]\!](D)}$ and $\bar{c}_i$ be the prefix corresponding to the current children, i.e., of the ball $\texttt{Children}(I, B_i).\texttt{current}$. This means that there is an answer $h \in S \subseteq [\![Q]\!](D)$ with the prefix p_i for any $i = 1, \ldots, l$ but there is no answer with the prefix $\bar{c}_i[1], \ldots, \bar{c}_i[|\bar{p}_i + 1|]$. Therefore, the incremental diversity of each $b_i := \texttt{Member}(I, \texttt{Children}(I, B_i).\texttt{current})$ is

$$\delta(S \cup \{b_i\}) - \delta(S) = \mathbf{u}_{\text{rel}}(b_i, S) = 2^{-|\bar{p}_i|-1}.$$

Thus, by storing L as an array (of length $|Q|$) of sets with B_i in the set at position $|\bar{p}_i + 1|$, we can find a B as required in line 7 in time $|Q|$.

By reinspecting how we arrive at the run time in Theorem 6.1 – in particular why the term $k^2 \cdot f(k)$ arises – this then totals to the run time as required. □

Received May 2024; accepted August 2024