Improved Approximation Algorithms for Relational Clustering[*]

ARYAN ESMAILPOUR, Department of Computer Science, University of Illinois Chicago, USA
STAVROS SINTOS, Department of Computer Science, University of Illinois Chicago, USA

Clustering plays a crucial role in computer science, facilitating data analysis and problem-solving across numerous fields. By partitioning large datasets into meaningful groups, clustering reveals hidden structures and relationships within the data, aiding tasks such as unsupervised learning, classification, anomaly detection, and recommendation systems. Particularly in relational databases, where data is distributed across multiple tables, efficient clustering is essential yet challenging due to the computational complexity of joining tables. This paper addresses this challenge by introducing efficient algorithms for k-median and k-means clustering on relational data without the need for pre-computing the join query results. For the relational k-median clustering, we propose the first efficient relative approximation algorithm. For the relational k-means clustering, our algorithm significantly improves both the approximation factor and the running time of the known relational k-means clustering algorithms, which suffer either from large constant approximation factors, or expensive running time. Given a join query $\boldsymbol{q}$ and a database instance $\mathbf{D}$ of $O(N)$ tuples, for both k-median and k-means clustering on the results of $\boldsymbol{q}$ on $\mathbf{D}$, we propose randomized $(1+\varepsilon)\gamma$-approximation algorithms that run in roughly $O(k^2 N^{\mathsf{fhw}}) + T_\gamma(k^2)$ time, where $\varepsilon \in (0,1)$ is a constant parameter decided by the user, fhw is the fractional hyper-tree width of $\boldsymbol{q}$, while γ and $T_\gamma(x)$ represent the approximation factor and the running time, respectively, of a traditional clustering algorithm in the standard computational setting over x points.

CCS Concepts: • **Theory of computation → Data structures and algorithms for data management**.

Additional Key Words and Phrases: relational data, clustering, k-median, k-means, coreset

ACM Reference Format:
Aryan Esmailpour and Stavros Sintos. 2024. Improved Approximation Algorithms for Relational Clustering. *Proc. ACM Manag. Data* 2, 5 (PODS), Article 213 (November 2024), 27 pages. https://doi.org/10.1145/3695831

1 INTRODUCTION

Clustering is a fundamental process in computer science, serving as a vital tool for data analysis, pattern recognition, and problem-solving across various domains. Through clustering, large datasets are partitioned into meaningful groups, unveiling hidden structures and relationships within the data. In machine learning, clustering algorithms are used to enable tasks such as classification, anomaly detection, and recommendation systems. Its significance lies in its ability to transform raw data into useful knowledge, empowering researchers, and businesses to make informed decisions.

In relational databases, data is gathered and stored across various tables. Each table consists of a set of tuples and two tuples stored in different tables might refer to the same entity. Relational data

[*]This work was partially supported by NSF grant IIS-2348919.

Authors' addresses: Aryan Esmailpour, Department of Computer Science, University of Illinois Chicago, Chicago, USA, aesmai2@uic.edu; Stavros Sintos, Department of Computer Science, University of Illinois Chicago, Chicago, USA, stavros@uic.edu.

ACM 2836-6573/2024/11-ART213
https://doi.org/10.1145/3695831

is decoupled into different relational tables, and we cannot obtain the full data unless all the tables are joined. This setting is quite common in real database management systems (DBMS). As shown in the DB-engines study [1] the vast majority of database systems are relational DBMS. Kaggle surveys [2] demonstrate that most of the learning tasks encountered by data scientists involve relational data. More specifically, 70% of database systems are relational DBMS and 65% of the data sets in learning tasks are relational data. As mentioned in [20], relational data is expected to reach \$122.38 billion by 2027 [49] in investments.

In order to explore and process relational data, usually two steps are required: data preparation, and data processing. In the data preparation step, tuples from different tables are joined to construct useful data, while in data processing, an algorithm (for example a clustering algorithm) is performed on the join results to analyze the data. This two-step approach is usually too expensive because the size of the join results can be polynomially larger than the total size of the input tables [46, 47].

In this paper, given a (full) conjunctive query and a database instance, we study efficient k-median and k-means clustering on the results of the query without first computing the query results. While there are recent papers [20, 23, 44] on designing clustering algorithms on relational data, they usually suffer from i) large additive approximation error, ii) large constant relative approximation error, and iii) expensive running time.

Despite recent attention to relational clustering, the problem of efficiently solving relational k-median clustering without additive approximation error remained unsolved. In this paper, we propose the first efficient relative approximation algorithm for the k-median clustering on relational data. Furthermore, by extending our methods, we design an algorithm for the k-means clustering on relational data that dominates (with respect to both the running time and the approximation ratio) all the known relative approximation algorithms.

1.1 Notation and problem definition

Conjunctive Queries. We are given a database schema $\mathbf{R}$ over a set of d attributes $\mathbf{A}$. The database schema $\mathbf{R}$ contains m relations $R_1, \ldots, R_m$. Let $\mathbf{A}_j \subseteq \mathbf{A}$ be the set of attributes associated with relation $R_j \in \mathbf{R}$. For an attribute $A \in \mathbf{A}$, let $\mathrm{dom}(A)$ be the domain of attribute A. We assume that $\mathrm{dom}(A) = \mathbb{R}$ for every $A \in \mathbf{A}$. Let $\mathbf{D}$ be a database instance over the database schema $\mathbf{R}$. For simplicity, we assume that each relation R_j contains N tuples in $\mathbf{D}$. Throughout the paper, we consider data complexity i.e., m and d are constants, while N is a large integer. We use R_j to denote both the relation and the set of tuples from $\mathbf{D}$ stored in the relation. For a subset of attributes $B \subseteq \mathbf{A}$ and a set $Y \subset \mathbb{R}^d$, let $\pi_B(Y)$ be the set containing the projection of the tuples in Y onto the attributes B. Notice that two different tuples in Y might have the same projection on B, however, $\pi_B(Y)$ is defined as a set, so the projected tuple is stored once. We also define the multi-set $\bar{\pi}_B(Y)$ so that if for two tuples $t_1, t_2 \in Y$ it holds that $\pi_B(t_1) = \pi_B(t_2)$, then the tuple $\pi_B(t_1)$ exists more than once in $\bar{\pi}_B(Y)$.

Following the related work on relational clustering [20, 23, 44], we are given a full conjunctive query (join query) $\boldsymbol{q} := R_1 \bowtie \ldots \bowtie R_m$. The set of results of a join query $\boldsymbol{q}$ over the database instance $\mathbf{D}$ is defined as $\boldsymbol{q}(\mathbf{D}) = \{t \in \mathbb{R}^d \mid \forall j \in [1, m] : \pi_{\mathbf{A}_j}(t) \in R_j\}$. For simplicity, all our algorithms are presented assuming that $\boldsymbol{q}$ is an acyclic join query, however in the end we extend to any general join query. A join query $\boldsymbol{q}$ is acyclic if there exists a tree, called join tree, such that the nodes of the tree are the relations in $\mathbf{R}$ and for every attribute $A \in \mathbf{A}$, the set of nodes/relations that contain A form a connected component. Let $\rho^*(\boldsymbol{q})$ be the fractional edge cover of query $\boldsymbol{q}$, which is a parameter that bounds the number of join results $\boldsymbol{q}(\mathbf{D})$ over any database instance. More formally, for every database instance $\mathbf{D}'$ with $O(N)$ tuples in each relation, it holds that $|\boldsymbol{q}(\mathbf{D}')| = O(N^{\rho^*(\boldsymbol{q})})$ as shown in [13].

We use the notation $\mathsf{fhw}(\boldsymbol{q})$ to denote the *fractional hypertree width* [29] of the query $\boldsymbol{q}$. The fractional hypertree width roughly measures how close $\boldsymbol{q}$ is to being acyclic. For every acyclic join query $\boldsymbol{q}$, we have $\mathsf{fhw}(\boldsymbol{q}) = 1$. Given a cyclic join query $\boldsymbol{q}$, we convert it to an equivalent acyclic query such that each relation is the result of a (possibly cyclic) join query with fractional edge cover at most $\mathsf{fhw}(\boldsymbol{q})$. Hence, a cyclic join query over a database instance with $O(N)$ tuples per relation can be converted, in $O(N^{\mathsf{fhw}(\boldsymbol{q})})$ time, to an equivalent acyclic join query over a database instance with $O(N^{\mathsf{fhw}(\boldsymbol{q})})$ tuples per relation [13]. A more formal definition of fhw is given in Appendix E. If $\boldsymbol{q}$ is clear from the context, we write fhw instead of $\mathsf{fhw}(\boldsymbol{q})$.

For two tuples/points[1] $p, q \in \mathbb{R}^d$ let $\phi(p,q) = ||p-q|| = \left(\sum_{j=1,\dots,d}(\pi_{A_j}(p) - \pi_{A_j}(q))^2\right)^{1/2}$ be the Euclidean distance between p and q. Throughout the paper, we use the Euclidean distance to measure the error of the clustering.

Clustering. In this paper, we focus on k-median and k-means clustering. We start with some useful general definitions. Let P be a set of points in $\mathbb{R}^d$ and let C be a set of k centers/points in $\mathbb{R}^d$. Let $w : \mathbb{R}^d \to \mathbb{R}_{>0}$ be a weight function such that $w(p)$ is the weight of point $p \in P$. For a point $p \in \mathbb{R}^d$, let $\phi(p, C) = \min_{c \in C} \phi(p, c)$. We define

$$\mathbf{v}_C(P) = \sum_{p \in P} w(p)\phi(p, C), \quad \text{and} \quad \mu_C(P) = \sum_{p \in P} w(p)\phi^2(p, C).$$

If P is an unweighted set, then $w(p) = 1$ for every $p \in P$.

k-**median clustering**: Given a weight function w, a set of points P in $\mathbb{R}^d$ and a parameter k, the goal is to find a set $C \subset \mathbb{R}^d$ with $|C| = k$ such that $\mathbf{v}_C(P)$ is minimized. This is also called the *geometric k-median clustering* problem. Equivalently, we define the *discrete k-median clustering* problem, where the goal is to find a set $C \subseteq P$ with $|C| = k$ such that $\mathbf{v}_C(P)$ is minimized. Let $\mathsf{OPT}(P) = \arg\min_{S \in \mathbb{R}^d, |S|=k} \mathbf{v}_S(P)$ be a set of k centers in $\mathbb{R}^d$ with the minimum $\mathbf{v}_{\mathsf{OPT}(P)}(P)$. For the discrete k-median problem let $\mathsf{OPT}_{\mathsf{disc}}(P) = \arg\min_{S \subseteq P, |S|=k} \mathbf{v}_S(P)$. It is always true that $\mathbf{v}_{\mathsf{OPT}(P)}(P) \le \mathbf{v}_{\mathsf{OPT}_{\mathsf{disc}}(P)}(P) \le 2\mathbf{v}_{\mathsf{OPT}(P)}(P)$. Let $\mathsf{GkMedianAlg}_\gamma$ (resp. $\mathsf{DkMedianAlg}_\gamma$) be a (known) γ-approximation algorithm for the geometric k-median problem (resp. discrete k-median problem) in the standard computational setting[2] that runs in $T_\gamma^{\mathsf{med}}(|P|)$ time, where γ is a constant.

k-**means clustering**: Given a weight function w, a set of points P in $\mathbb{R}^d$ and a parameter k, the goal is to find a set $C \subset \mathbb{R}^d$ with $|C| = k$ such that $\mu_C(P)$ is minimized. This is also called the *geometric k-means clustering* problem. Equivalently, we define the *discrete k-means clustering* problem, where the goal is to find a set $C \subseteq P$ with $|C| = k$ such that $\mu_C(P)$ is minimized. Let $\mathsf{OPT}(P) = \arg\min_{S \in \mathbb{R}^d, |S|=k} \mu_S(P)$ be a set of k centers in $\mathbb{R}^d$ with the minimum $\mu_{\mathsf{OPT}(P)}(P)$. For the discrete k-means problem let $\mathsf{OPT}_{\mathsf{disc}}(P) = \arg\min_{S \subseteq P, |S|=k} \mu_S(P)$. It is always true that $\mu_{\mathsf{OPT}(P)}(P) \le \mu_{\mathsf{OPT}_{\mathsf{disc}}(P)}(P) \le 4\mu_{\mathsf{OPT}(P)}(P)$. Let $\mathsf{GkMeansAlg}_\gamma$ (resp. $\mathsf{DkMeansAlg}_\gamma$) be a (known) γ-approximation algorithm for the geometric k-means problem (resp. discrete k-means problem) in the standard computational setting that runs in $T_\gamma^{\mathsf{mean}}(|P|)$ time, where γ is a constant.

For simplicity, we use the same notation $T_\gamma^{\mathsf{med}}(\cdot)$ for the running time of GkMedianAlg and DkMedianAlg. Similarly, we use the same notation $T_\gamma^{\mathsf{mean}}(\cdot)$ for the running time of GkMeansAlg and DkMeansAlg. We also use the same notation $\mathsf{OPT}(\cdot)$ for the optimum solution for k-means and k-median clustering problems. It is always clear from the context whether we are referring to k-means or k-median clustering.

In this paper, we study (both geometric and discrete) k-median and k-means clustering on the result of a join query:

[1]The terms points and tuples are used interchangeably.

[2]In the standard computational setting data is stored in one table.

Definition 1.1. [Relational k-median clustering] Given a database instance $\mathbf{D}$, a join query $\boldsymbol{q}$, and a positive integer parameter k, the goal is to find a set $\mathcal{S} \subseteq \mathbb{R}^d$ of size $|\mathcal{S}| = k$ such that $\mathbf{v}_C(\boldsymbol{q}(\mathbf{D}))$ is minimized. In the discrete relational k-median clustering the set $\mathcal{S}$ should be a subset of $\boldsymbol{q}(\mathbf{D})$.

Definition 1.2. [Relational k-means clustering] Given a database instance $\mathbf{D}$, a join query $\boldsymbol{q}$, and a positive integer parameter k, the goal is to find a set $\mathcal{S} \subseteq \mathbb{R}^d$ of size $|\mathcal{S}| = k$ such that the $\mu_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))$ is minimized. In the discrete relational k-means clustering the set $\mathcal{S}$ should be a subset of $\boldsymbol{q}(\mathbf{D})$.

By relational k-median or k-means clustering, we are referring to the geometric versions, unless explicitly stated otherwise for the discrete variants. We propose results for both versions. We note that the clustering problem on relational data is defined over the unweighted set $\boldsymbol{q}(\mathbf{D})$. In order to propose efficient algorithms we will need to construct weighted subset of tuples (*coresets*) that are used to derive small approximation factors for the relational clustering problems.

Approximation. We say that an algorithm is a relative β-approximation algorithm for the relational k-median clustering if it returns a set $\mathcal{S}$ of size k such that $\mathbf{v}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D})) \leq \beta \cdot \mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}\boldsymbol{q}(\mathbf{D})$. Next, we say that an algorithm is an additive β-approximation algorithm for the relational k-median clustering if it returns a set $\mathcal{S}$ of size k such that $\mathbf{v}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D})) \leq \mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}\boldsymbol{q}(\mathbf{D}) + \beta$. In this paper, we focus on relative approximation algorithms, so when we refer to a β-approximation algorithm we always mean relative β-approximation. Equivalently, we define relative and additive approximation algorithms for the relational k-means clustering.

1.2 Related work

There is a lot of research on k-median and k-means clustering in the standard computational setting. For the k-median clustering there are several polynomial time algorithms with constant approximation ratio [12, 19, 40]. For the k-means clustering, there are also several constant approximation algorithms such as [35]. In practice, a local search algorithm [41] is mostly used with a $O(\log k)$ approximation ratio. Furthermore, coresets (formal definition in Section 2) have been used to design efficient clustering algorithms [14, 32, 33]. Coresets are also used to propose efficient clustering algorithms in the streaming setting [16, 30] or the MPC model [15, 26]. All these algorithms work in the standard computational setting and it is not clear how to efficiently extend them to relational clustering.

In its most general form, relational clustering is referring to clustering objects connected by links representing persistent relationships between them. Different variations of relational clustering have been studied over the years in the database and data mining community, for example [9, 17, 28, 34, 38, 42, 45]. However, papers in this line of work either do not handle clustering on join results or their methods do not have theoretical guarantees.

The discrete relational k-means clustering problem has been recently studied in the literature. Khamis et al. [36], gave an efficient implementation of the Lloyd's k-means heuristic in the relational setting, however it is known that the algorithm terminates in a local minimum without any guarantee on the approximation factor. Relative approximation algorithms are also known for the relational k-means clustering problem. Curtin et al. [23] construct a weighted set of tuples (*grid-coreset*) such that an approximation algorithm for the weighted k-means clustering in the grid-coreset returns an approximation solution to the relational k-means clustering. Their algorithm runs in $\tilde{O}(k^m N^{\mathrm{fhw}} + T_\gamma^{\mathrm{mean}}(k^m))$ time and has a $(\gamma^2 + 4\gamma\sqrt{\gamma} + 4\gamma)$-approximation factor. For some join queries, the approximation factor can be improved to $(4\gamma + 2\sqrt{\gamma} + 1)$. Moseley et al. [44] designed a relational implementation of the k-means++ algorithm [8, 11] to derive a better weighted coreset. For a constant $\varepsilon \in (0, 1)$, their algorithm runs in $O(k^4 N^{\mathrm{fhw}} \log N + k^2 N^{\mathrm{fhw}} \log^9 N + T_\gamma^{\mathrm{mean}}(k \log N))$ expected time and has a $(320 + 644(1+\varepsilon)\gamma)$-approximation factor. No efficient relative approximation algorithm is known for the relational k-median problem.

Problem	Method	Approximation	Running time	Type
k-median	**NEW**	$(2+\varepsilon)\gamma$	$\tilde{O}(k^{2d+2}N^{\mathsf{fhw}} + T_\gamma^{\mathsf{med}}(k^2))$	D
	NEW	$(2+\varepsilon)\gamma$	$\tilde{O}(k^2N^{\mathsf{fhw}} + k^4 + T_\gamma^{\mathsf{med}}(k^2))$	R
k-means	[23]	$\gamma^2 + 4\gamma\sqrt{\gamma} + 4\gamma$	$\tilde{O}(k^m N^{\mathsf{fhw}} + T_\gamma^{\mathsf{mean}}(k^m))$	D
	[44]	$320 + 644(1+\varepsilon)\gamma$	$\tilde{O}(k^4 N^{\mathsf{fhw}} + T_\gamma^{\mathsf{mean}}(k))$	R
	NEW	$(4+\varepsilon)\gamma$	$\tilde{O}(k^{2d+2}N^{\mathsf{fhw}} + T_\gamma^{\mathsf{mean}}(k^2))$	D
	NEW	$(4+\varepsilon)\gamma$	$\tilde{O}(k^2N^{\mathsf{fhw}} + k^4 + T_\gamma^{\mathsf{mean}}(k^2))$	R

Table 1. Comparison of our new algorithms with the state-of-the-art relative approximation algorithms. For our new algorithms we show the approximation for the discrete relational k-median and k-means clustering problems. For the geometric version of the studied problems the approximation factor of our algorithms is always $(1+\varepsilon)\gamma$. The running time is shown in data complexity. We assume that $\varepsilon \in (0,1)$ is a small constant. The notation $\tilde{O}$ is used to hide $\log^{O(1)} N$ factors from the running time. $T_\gamma^{\mathsf{med}}(y)$ (resp. $T_\gamma^{\mathsf{mean}}(y)$) is the running time of a known γ-approximation algorithm over y points for the k-median (resp. k-means) clustering in the standard computational setting. fhw is the fractional hypertree width of $\boldsymbol{q}$. The number of attributes in the query $\boldsymbol{q}$ is denoted by d. The letter R stands for randomized, while D stands for deterministic algorithm.

Additive approximation algorithms are also known for relational clustering problems. Chen et al. [20], constructed coresets for empirical risk minimization problems in relational data. Their algorithm is quite general and can support relational clustering, i.e., k-median and k-means can be formulated as risk minimization problems. They work independently in every relation to compute a good enough coreset and then by the *aggregation tree* algorithm they merge the solutions carefully using properties of the k-center clustering. We note that the authors do not design algorithms for relational clustering problems, instead, they compute a coreset such that any approximation algorithm (for k-median or k-means) on the coreset returns an (additive) approximation of the relational k-median or k-means problem. Running GkMedianAlg on top of their coreset leads to a randomized algorithm that runs in $O(N^{\mathsf{fhw}} + T_\gamma^{\mathsf{med}}(1))$ time and has an $\varepsilon \cdot \gamma \cdot \mathsf{diam}(\boldsymbol{q}(\mathbf{D}))$ additive approximation term, with high probability, where $\mathsf{diam}(\boldsymbol{q}(\mathbf{D}))$ is the largest Euclidean distance between two tuples in $\boldsymbol{q}(\mathbf{D})$. For the relational k-means clustering, their algorithm runs in $O(N^{\mathsf{fhw}} + T_\gamma^{\mathsf{mean}}(1))$ time, and has an $\varepsilon \cdot \gamma \cdot \mathsf{diam}^2(\boldsymbol{q}(\mathbf{D}))$ additive approximation term, with high probability. In all cases we assume that $\varepsilon \in (0,1)$ is a small constant.

Finally, there is a lot of recent work on relational algorithms for learning problems such as, linear regression and factorization [37, 39, 48, 51], SVMs [3, 4, 55], Independent Gaussian Mixture models [21, 22]. In [50] the authors give a nice survey about learning over relational data. Generally, in databases there is an interesting line of work solving combinatorial problems over relational data without first computing the join results, such as ranked enumeration [24, 25, 53], quantiles [54], direct access [18], diversity [6, 10, 43], and top-k [53].

1.3 Our results

The main results for the discrete relational clustering in this paper are summarized in Table 1. In all cases, we assume that $\varepsilon \in (0,1)$ is a small constant decided by the user.

First, we present a deterministic $(1+\varepsilon)\gamma$-approximation algorithm for the (geometric) relational k-median clustering that runs in $O(k^{2d+2}N^{\mathsf{fhw}} \log^{d+2} N + T_\gamma^{\mathsf{med}}(k^2 \log N))$ time. Then, we show a randomized $(1+\varepsilon)\gamma$-approximation algorithm for the (geometric) relational k-median clustering that works with high probability and runs in $O(k^2N^{\mathsf{fhw}} \log N + k^4 \log^3(N) \log^d(k) + T_\gamma^{\mathsf{med}}(k^2 \log N))$ time. If $k^2 \log(N) < \log^d(k)$ the running time can be improved to $O(k^2N^{\mathsf{fhw}} \log N + k^6 \log^4(N) + T_\gamma^{\mathsf{med}}(k^2 \log N))$. In the discrete case, the approximation factor is $(2+\varepsilon)\gamma$. These are the first known efficient relative approximation algorithms for the relational k-median clustering problem.

We extend our methods to show algorithms with the same guarantees for the relational k-means clustering. We give a deterministic $(1+\varepsilon)\gamma$-approximation algorithm for the (geometric) relational k-means clustering that runs in $O(k^{2d+2}N^{\mathsf{fhw}}\log^{d+2}N + T_{\gamma}^{\mathsf{mean}}(k^2\log N))$ time. Furthermore, we give a randomized $(1+\varepsilon)\gamma$-approximation algorithm that runs in $O(k^2N^{\mathsf{fhw}}\log N + k^4\log^3(N)\log^d(k) + T_{\gamma}^{\mathsf{mean}}(k^2\log N))$ time. If $k^2\log(N) < \log^d(k)$ the running time can be improved to $O(k^2N^{\mathsf{fhw}}\log N + k^6\log^4(N) + T_{\gamma}^{\mathsf{mean}}(k^2\log N))$. In the discrete case the approximation factor is $(4+\varepsilon)\gamma$. Our algorithms significantly improve both the approximation factor and the running time of the known algorithms on relational k-means clustering. Due to space limit, in the next sections we focus on the relational k-median clustering. We show all the results for the k-means clustering in Appendix A.

Remark 1. While our algorithms work for both acyclic and cyclic join queries, we first present all our results assuming that $\boldsymbol{q}$ is an acyclic join query. In Section 4.2 we extend our results for every join query using the generalized hypertree decomposition [29]. From now on we consider that $\boldsymbol{q}$ is an acyclic join query, so $\mathsf{fhw} = 1$.

Remark 2. For the relational k-means clustering, the algorithm in [44] is generally faster, by $\log N$ factors, than the other algorithms for $k = O(1)$.

2 PRELIMINARIES

2.1 Coreset

We give the definition of coresets for both the relational k-median and k-means clustering problems. Our algorithms construct small enough coresets to approximate the cost of the relational clustering.

A weighted set $C \subseteq \mathbb{R}^d$ is an ε-coreset for the relational k-median clustering problem on $\boldsymbol{q}(\mathbf{D})$ if for any set of k centers $Y \subset \mathbb{R}^d$,

$$(1-\varepsilon)\mathbf{v}_Y(\boldsymbol{q}(\mathbf{D})) \leq \mathbf{v}_Y(C) \leq (1+\varepsilon)\mathbf{v}_Y(\boldsymbol{q}(\mathbf{D})). \tag{1}$$

Similarly, a weighted set C is an ε-coreset for the relational k-means clustering problem, if for any set of k centers $Y \subset \mathbb{R}^d$,

$$(1-\varepsilon)\mu_Y(\boldsymbol{q}(\mathbf{D})) \leq \mu_Y(C) \leq (1+\varepsilon)\mu_Y(\boldsymbol{q}(\mathbf{D})). \tag{2}$$

2.2 Data structures for aggregation queries

In this subsection, we show how we can combine known results in database theory to answer aggregation queries in linear time with respect to the size of the database.

Let $\mathcal{R}$ be an axis-parallel hyper-rectangle in $\mathbb{R}^d$.The goal is i) count the number of tuples $|\boldsymbol{q}(\mathbf{D})\cap\mathcal{R}|$, and ii) sample uniformly at random from $\boldsymbol{q}(\mathbf{D}) \cap \mathcal{R}$. The axis-parallel hyper-rectangle $\mathcal{R}$ is defined as the product of d intervals over the attributes, i.e., $\mathcal{R} = \{I_1 \times \ldots \times I_d\}$, where $I_i = [a_i, b_i]$ for $a_i, b_i \in \mathbb{R}$. Hence, $\mathcal{R}$ defines a set of d linear inequalities over the attributes, i.e., a tuple t lies in $\mathcal{R}$ if and only if $a_j \leq \pi_{A_j}(t) \leq b_j$ for every $A_j \in \mathbf{A}$. Let p be a tuple in a relation R_i. If $a_j \leq \pi_{A_j}(p) \leq b_j$ for every $A_j \in \mathbf{A}_i$, then we keep p in R_i. Otherwise, we remove it. The set of surviving tuples is exactly the set of tuples that might lead to join results in $\mathcal{R}$. Let $\mathbf{D}' \subseteq \mathbf{D}$ be the new database instance such that $\boldsymbol{q}(\mathbf{D}') = \boldsymbol{q}(\mathbf{D}) \cap \mathcal{R}$. The set $\mathbf{D}'$ is found in $O(N)$ time. Using Yannakakis algorithm [56] we can count $|\boldsymbol{q}(\mathbf{D}')|$ in $O(N\log N)$ time and using [57] we can sample z tuples from $\boldsymbol{q}(\mathbf{D}')$ in $O((N+z)\log N)$ time. Using hashing, we can improve the running time of the algorithms to $O(N)$ and $O(N + z\log N)$, respectively.

Lemma 2.1. *Let $\mathcal{R}$ be a rectangle in $\mathbb{R}^d$. There exists an algorithm* CountRect$(\boldsymbol{q}, \mathbf{D}, \mathcal{R})$ *to count $|\boldsymbol{q}(\mathbf{D}) \cap \mathcal{R}|$ in $O(N\log N)$ time or $O(N)$ time with high probability. Furthermore, there exists an algorithm* SampleRect$(\boldsymbol{q}, \mathbf{D}, \mathcal{R}, z)$ *to sample z samples from $\boldsymbol{q}(\mathbf{D}) \cap \mathcal{R}$ in $O((N+z)\log N)$ time or $O(N + z\log N)$ time with high probability.*

Let $B \subseteq \mathbf{A}$ be a subset of the attributes. The result in Lemma 2.1 can also be used (straightforwardly) to compute $|\bar{\pi}_B(\boldsymbol{q}(\mathbf{D})) \cap \mathcal{R}|$ or sample z samples from $\bar{\pi}_B(\boldsymbol{q}(\mathbf{D})) \cap \mathcal{R}$ in the same running time, since $|\bar{\pi}_B(\boldsymbol{q}(\mathbf{D})) \cap \mathcal{R}| = |\boldsymbol{q}(\mathbf{D}) \cap \mathcal{R}|$.

2.3 High level ideas

Before we start with the technical sections of the paper, we give high level ideas of our methods.

In Section 3, we assume that a set of centers X for the relational k-median clustering is given such that $\mathbf{v}_X(\boldsymbol{q}(\mathbf{D}))$ is within a constant factor from the optimum k-median error with $|X| > k$. Using X, we construct a coreset C for the relational k-median problem. Then we run $\mathsf{GkMedianAlg}_\gamma$ (or $\mathsf{DkMeansAlg}_\gamma$ for the discrete version) on C to obtain the final solution $\mathcal{S}$. While all previous papers on relational clustering [20, 23, 44] construct weighted coresets, the coresets that were used are sub-optimal. Instead, we use and modify the coreset construction from [33], which is a more optimized coreset for the k-median (and k-means) clustering under the Euclidean metric, in the standard computational setting. Of course, using such a coreset comes at a cost. First, it is not clear how to construct a set X with the desired properties on relational data, efficiently. In Section 4 we show how to construct X designing a hierarchical method over the attributes $\mathbf{A}$. Second, it is not straightforward to construct the coreset in [33] on relational data, given X. In fact, the coreset construction in [33] cannot be applied (efficiently) in the relational setting. Hence, inspired by [33], we design a novel small coreset based on X and show that it can be applied to relational data.

More specifically, for any center $x_i \in X$, in [33], the authors first compute all points in the dataset that have x_i as the closest center in X. Let P_i be this set of points. Then, they construct a grid around x_i, and from every cell $\square$ in the grid, they add one representative point from $P_i \cap \square$ with weight $|P_i \cap \square|$. Unfortunately, in our setting we cannot compute P_i and/or $|P_i \cap \square|$ efficiently.

We resolve the two issues as follows. First, we construct a grid around x_i, but instead of computing all tuples that have x_i as the closest center and process all cells, we check whether a cell $\square$ is close enough to the center x_i (Equation (3) in the next section). If not then we skip the cell. If yes, then we take a representative tuple from $\square$ and we set its weight to be the number of tuples in $\square$ that do not lie in a different cell that has been already processed by our algorithm. Of course, the weight of a representative tuple now is not the same as in [33] and we might count tuples even outside of P_i, however with a careful analysis we make sure that the overall error is still bounded. The second problem though, still remains; we need to count the number of tuples in $\square$ excluding the cells we have already visited from previous centers in X (notice that the grid cells of two different centers in X might intersect). We propose two methods to achieve it. In Section 3.1 we give a deterministic method that constructs the arrangement of the complement of the visited cells and we use Lemma 2.1 to count the number of tuples, exactly. However, this algorithm requires $\Omega(|X|^d N)$ time. In Section 3.2 we give a more involved and faster randomized approximation algorithm to count the number of tuples based on sampling.

3 FROM MANY CENTERS TO EXACTLY k CENTERS

We describe an algorithm that constructs a coreset for the relational k-median clustering over the (multi-set) projection of $\boldsymbol{q}(\mathbf{D})$ on an arbitrary subset of attributes. Let $\mathbf{A}_u \subseteq \mathbf{A}$ be a subset of the attributes. Let $\boldsymbol{q}_u(\mathbf{D})$ be the *multi-set* $\boldsymbol{q}_u(\mathbf{D}) := \bar{\pi}_{\mathbf{A}_u}(\boldsymbol{q}(\mathbf{D}))$. Notice that $|\boldsymbol{q}_u(\mathbf{D})| = |\boldsymbol{q}(\mathbf{D})|$ and its size can be computed in $O(N)$ time using Yannakakis algorithm. Let $n = |\boldsymbol{q}_u(\mathbf{D})|$ and $d_u = |\mathbf{A}_u|$. Assume that X is a set of points in $\mathbb{R}^{d_u}$ such that $\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D})) \leq \alpha \cdot \mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, where $\alpha > 1$ is a constant. Notice that

$$\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D})) = \sum_{t \in \boldsymbol{q}(\mathbf{D})} \phi(\pi_{\mathbf{A}_u}(t), X).$$

We also assume that r is a real number such that $\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D})) \le r \le \alpha \cdot \mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$. In fact, it follows from the following sections that we can also assume $\frac{r}{(1+\varepsilon)\sqrt{2}} \le \mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D}))$. Note that we do not assume anything about the size of X. In the next section, we show that we can always consider $|X| = O(k^2)$.

In this section we propose two algorithms that take as input X and r and return a set $\mathcal{S} \subset \mathbb{R}^{d_u}$ of cardinality $|\mathcal{S}| = k$ such that $\mathbf{v}_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \le (1+\varepsilon)\gamma\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, i.e., $\mathcal{S}$ is a $(1+\varepsilon)\gamma$-approximation for the k-median problem in $\boldsymbol{q}_u(\mathbf{D})$. They also return a number r_u such that $\mathbf{v}_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \le r_u \le (1+\varepsilon)\gamma\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$. We note that given a set of centers S, we cannot compute the error $\mathbf{v}_S(\boldsymbol{q}_u(\mathbf{D}))$ efficiently, hence r_u is needed to estimate the error of the clustering. We also note that if $\mathbf{A}_u = \mathbf{A}$, then the returned set $\mathcal{S}$ is an $(1+\varepsilon)\gamma$ approximation solution for the relational k-median clustering problem on $\boldsymbol{q}(\mathbf{D})$. For the discrete relational k-median clustering, we return $\mathcal{S} \subseteq \boldsymbol{q}(\mathbf{D})$ and r_u such that, $\mathbf{v}_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \le r_u \le (2+\varepsilon)\gamma\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$.

The first algorithm is a slow deterministic algorithm that runs in $\Omega(|X|^{d_u+1}N)$ time. The second algorithm is a faster randomized algorithm that runs in time roughly $O(|X|N)$ time. We mostly focus on the geometric version of the relational k-median clustering, however, we always highlight the differences with the discrete version.

3.1 Slow deterministic algorithm

For the slow deterministic algorithm we construct a hierarchical grid around every center $x_i \in X$. Then, for every cell that is close enough to x_i, we compute the number of join results in $\boldsymbol{q}_u(\mathbf{D})$ inside the cell that do not lie inside another cell previously processed by our algorithm. We count the number of join results using Lemma 2.1.

Algorithm. The pseudocode of this algorithm is shown in Algorithm 1. We first set $\varepsilon' = \varepsilon/4$ and define $\Phi = \frac{r}{\alpha n}$ as a lower bound estimate of the average radius $\frac{\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))}{n}$. For every point $x_i \in X$ we construct an exponential grid around x_i. Let $Q_{i,j}$ be an axis parallel square with side length $\Phi \cdot 2^j$ centered at x_i, for $j = 0, 1, \ldots, 2\log(\alpha n)$. Let $V_{i,0} = Q_{i,0}$ and let $V_{i,j} = Q_{i,j} \setminus Q_{i,j-1}$. We partition $V_{i,j}$ into a grid $\overline{V}_{i,j}$ of side length $\varepsilon'\Phi 2^j/(10\alpha d_u)$. Let $\overline{V}_i = \bigcup_j \overline{V}_{i,j}$.

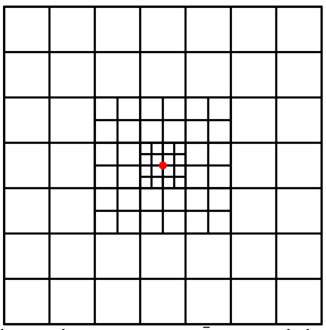
Fig. 1. The grid construction $\bar{V}_i$ around the red point $x_i \in X$.

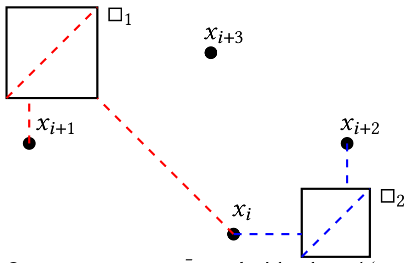

Fig. 2. Let $\square_1, \square_2 \in \bar{V}_i$. It holds that $\phi(x_i, \square_1) > \phi(x_{i+1}, \square_1) + \mathrm{diam}(\square_1)$ so $\square_1$ is not processed by the algorithm. x_{i+2} is the closest center to $\square_2$, i.e., $\phi(X, \square_2) = \phi(x_{i+2}, \square_2)$ and it holds $\phi(x_i, \square_2) < \phi(x_{i+2}, \square_2) + \mathrm{diam}(\square_2)$, so $\square_2$ is processed by the algorithm. The red (blue) dashed segments represent the distances of x_i to $\square_1$ ($\square_2$), x_{i+1} (x_{i+2}) to $\square_1$ ($\square_2$), and the diameter of $\square_1$ ($\square_2$).

An example of the grid construction can be seen in Figure 1. The construction so far is similar to the exponential grid in [33]. However, from now on the algorithm and the analysis is different. Let

Algorithm 1: $\textsc{RelClusteringSlow}(\boldsymbol{q}, \mathbf{D}, \mathrm{A}_u, X, \alpha, r, \varepsilon)$

```
1  $\varepsilon' = \varepsilon/4$;
2  $n = |\boldsymbol{q}_u(\mathbf{D})| = |\boldsymbol{q}(\mathbf{D}))|$        (using Lemma 2.1);
3  $\Phi = \frac{r}{\alpha n}$;   $G = \emptyset$;   $C = \emptyset$;
4  foreach $x_i \in X$ do
5      foreach $j = 0, 1, \ldots, 2\log(\alpha n)$ do
6          $Q_{i,j} \leftarrow$ axis parallel square with length $2^j\Phi$ centered at $x_i$;
7          $V_{i,j} = Q_{i,j} \setminus Q_{i,j-1}$;   $\overline{V}_{i,j} \leftarrow$ grid of side length $\varepsilon' 2^j \Phi/(10\alpha d_u)$ in $V_{i,j}$;
8      $\overline{V}_i = \bigcup_j \overline{V}_{i,j}$;
9      foreach $\square \in \overline{V}_i$ do
10         if $\phi(x_i, \square) \leq \phi(X, \square) + diam(\square)$ then
11             $\mathsf{Arr}(G) \leftarrow$ arrangement of $G$;   $\overline{\mathsf{Arr}}(G) \leftarrow$ complement of $\mathsf{Arr}(G)$;
12             $\mathsf{Arr}'(G) \leftarrow$ partition of $\overline{\mathsf{Arr}}(G)$ into hyper-rectangles;
13             $K_\square = 0$;
14             foreach $\mathcal{R} \in \mathsf{Arr}'(G)$ do
15                 $\square_\mathcal{R} = \square \cap \mathcal{R}$;   $K_\square = K_\square + \mathsf{CountRect}(\boldsymbol{q}, \mathbf{D}, \square_\mathcal{R})$      (using Lemma 2.1);
16             if $K_\square > 0$ then
17                 $s_\square \leftarrow$ arbitrary tuple in $\boldsymbol{q}_u(\mathbf{D}) \cap (\square \setminus G)$;
18                 $w(s_\square) = K_\square$;
19                 $C \leftarrow C \cup \{s_\square\}$;
20             $G \leftarrow G \cup \{\square\}$;
21 $\mathcal{S} = \mathsf{GkMedianAlg}_\gamma(C)$      (or $\mathcal{S} = \mathsf{DkMedianAlg}_\gamma(C)$ for the discrete version);
22 $r_u = \frac{1}{1-\varepsilon'}\mathbf{v}_\mathcal{S}(C)$;
23 return $(\mathcal{S}, r_u)$;
```

$G = \emptyset$ be an empty set of grid cells. Let also $C = \emptyset$ be an empty point set in $\mathbb{R}^{d_u}$ and w be a weight function that we are going to define on C.

For every $x_i \in X$, and for every cell $\square \in \overline{V}_i$ we repeat the following steps. Let $\mathsf{diam}(\square)$ be the diameter of $\square$. If

$$\phi(x_i, \square) \leq \phi(X, \square) + \mathsf{diam}(\square), \tag{3}$$

(an example of applying Equation (3) is shown in Figure 2) then we proceed as follows. Let $G_\square$ be the set G just before the algorithm processes the cell $\square$. The goal is to identify $|\boldsymbol{q}_u(\mathbf{D}) \cap (\square \setminus G_\square)|$ and if $|\boldsymbol{q}_u(\mathbf{D}) \cap (\square \setminus G_\square)| > 0$ then add a representative point $s_\square \in \boldsymbol{q}_u(\mathbf{D}) \cap (\square \setminus G_\square)$ in C with weight $w(s_\square) = |\boldsymbol{q}_u(\mathbf{D}) \cap (\square \setminus G_\square)|$. Next, we construct the arrangement of $G_\square$. The arrangement [7, 31] of $G_\square$, denoted $\mathsf{Arr}(G_\square)$, is a partitioning of $G_\square$ into rectangular contiguous regions, such that for every region reg in the arrangement, reg lies in the same subset of $G_\square$. Let $\overline{\mathsf{Arr}}(G_\square)$ be the complement of $\mathsf{Arr}(G_\square)$, and let $\mathsf{Arr}'(G_\square)$ be a partition of $\overline{\mathsf{Arr}}(G_\square)$ into hyper-rectangles. For each rectangle $\mathcal{R} \in \mathsf{Arr}'(G_\square)$, we compute $\square_\mathcal{R} = \square \cap \mathcal{R}$, which is also a hyper-rectangle in $\mathbb{R}^{d_u}$. Using the $\mathsf{CountRect}(\cdot)$ procedure form Lemma 2.1 we compute $K_\square = \sum_{\mathcal{R} \in \mathsf{Arr}'(G)} |\square_\mathcal{R} \cap \boldsymbol{q}_u(\mathbf{D})|$. If $K_\square > 0$ we also get an arbitrary point $s_\square$ from $\bigcup_{\mathcal{R} \in \mathsf{Arr}'(G)} \square_\mathcal{R} \cap \boldsymbol{q}_u(\mathbf{D})$. We add $s_\square$ in C and we set its weight $w(s_\square) = K_\square$. We say that all tuples in $\boldsymbol{q}_u(\mathbf{D}) \cap (\square \setminus G_\square)$ are *assigned* to $s_\square$, and we add $\square$ in G. An example of the arrangement can be seen in Figure 3.

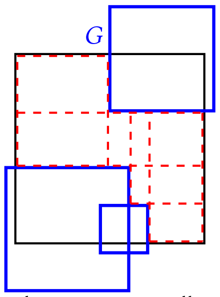

Fig. 3. The black square is a cell $\square \in \bar{V}_i$. The blue squares define the set of cells in $G_\square$ (cells already processed by the algorithm) that intersect $\square$. The red dashed segments show the arrangement of the complement of $G_\square$ in $\square$, i.e., the red dashed segments show $\square \cap \mathrm{Arr}'(G_\square)$.

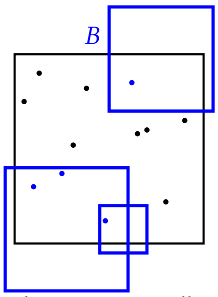

Fig. 4. The black square is a cell $\square \in \bar{V}_i$. The blue squares define the set of heavy cells in $B_\square$ that intersect $\square$. The points represent the set of samples $H_\square$. Blue points are the samples in $B_\square$ while black points are the samples in $\square \setminus B_\square$. Hence, $g_\square = 8$ and $M = |H_\square| = 12$. If $\tau = 0.3$ then $g_\square/M \geq 2 \cdot \tau$, and a black point is selected as $s_\square$ with weight $\frac{8}{12} \cdot \frac{n_\square}{1-\varepsilon'}$.

On the other hand, if the condition (3) is not satisfied, then we skip $\square$ and we continue with the next cell in the exponential grid.

In the end, after repeating the algorithm for each $x_i \in X$, we get a weighted set C. We run the standard algorithm for the weighted k-median problem $\mathsf{GkMedianAlg}_\gamma$ (or $\mathsf{DkMedianAlg}_\gamma$ for the discrete k-median problem) on C to get a set of k centers $\mathcal{S}$, and we return $\mathcal{S}$ as the answer. Furthermore, we return $r_u = \frac{1}{1-\varepsilon'}\mathbf{v}_\mathcal{S}(C)$.

Correctness. We first show that every tuple in $\boldsymbol{q}_u(\mathbf{D})$ is assigned to a point in C. Then, we show that C is an ε'-coreset for $\boldsymbol{q}_u(\mathbf{D})$. Next, we show that $\mathcal{S}$ is a good approximation of the optimum k-median solution and r_u is a good approximation of $\mathbf{v}_\mathcal{S}(\boldsymbol{q}_u(\mathbf{D}))$. Finally, we show that $\mathbf{v}_\mathcal{S}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (1+\varepsilon)\gamma \mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$. All missing proofs can be found in Appendix B.

Lemma 3.1. *Every tuple $t \in \boldsymbol{q}_u(\mathbf{D})$ is assigned to a point in C. Furthermore, the number of tuples in $\boldsymbol{q}_u(\mathbf{D})$ that are assigned to a point $s \in C$ is $w(s)$.*

Proof. Let $x_i \in X$ be the center that is closest to t. By definition, there will be a cell $\square$ defined by the exponential grid around x_i that contains t, since $\phi(t, x_i) \leq \alpha n \Phi$. We show that for the cell $\square$ the condition (3) holds. Let x_j be the center such that $\phi(x_j, \square) = \phi(X, \square)$. We have $\phi(x_i, \square) \leq \phi(x_i, t) \leq \phi(x_j, t) \leq \phi(X, \square) + \mathrm{diam}(\square)$. Hence, t is assigned in $s_\square$. The second part of the lemma holds by definition. $\square$

For a tuple $t \in \boldsymbol{q}_u(\mathbf{D})$, let $\square_t$ be the cell from the exponential grid defined by the algorithm such that $\square_t$ is the first cell processed by the algorithm that contains t, i.e., $t \in \boldsymbol{q}_u(\mathbf{D}) \cap (\square_t \setminus G_{\square_t})$. Next, we denote the assignment of each tuple $t \in \boldsymbol{q}_u(\mathbf{D})$ to a point in C by σ, i.e., $\sigma(t) := s_{\square_t} \in C$. Let $i(t)$ be the index such that $\square_t \in \overline{V}_{i(t)}$. By definition, notice that $x_{i(t)} \in X$ is the center visited by our algorithm at the moment that t is assigned to $\sigma(t)$.

Lemma 3.2. *C is an ε'-coreset for $\boldsymbol{q}_u(\mathbf{D})$.*

Proof. Let Y be an arbitrary set of k points in $\mathbb{R}^{d_u}$. The error is defined as $\mathcal{E} = |\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D})) - \mathbf{v}_Y(C)| \leq \sum_{t \in \boldsymbol{q}_u(\mathbf{D})} |\phi(t, Y) - \phi(\sigma(t), Y)|$. By the triangle inequality, $\phi(t, Y) \leq \phi(\sigma(t), Y) + \phi(t, \sigma(t))$ and $\phi(\sigma(t), Y) \leq \phi(t, Y) + \phi(t, \sigma(t))$. Hence, $|\phi(t, Y) - \phi(\sigma(t), Y)| \leq \phi(t, \sigma(t))$. We have,

$$\mathcal{E} \leq \sum_{t \in \boldsymbol{q}_u(\mathbf{D})} \phi(t, \sigma(t)) = \sum_{t \in \boldsymbol{q}_u(\mathbf{D}), \phi(t, x_{i(t)}) \leq \Phi} \phi(t, \sigma(t)) \quad + \sum_{t \in \boldsymbol{q}_u(\mathbf{D}), \phi(t, x_{i(t)}) > \Phi} \phi(t, \sigma(t)).$$

For $\phi(t, x_{i(t)}) \le \Phi$, by the construction of the exponential grid, we have $\phi(t, \sigma(t)) \le \mathrm{diam}(\Box_t) \le \frac{\varepsilon'}{10\alpha}\Phi$, hence

$$\sum_{t\in \boldsymbol{q}_u(\mathbf{D}), \phi(t, x_{i(t)})\le\Phi} \phi(t, \sigma(t)) \le \frac{\varepsilon'}{10\alpha} n\Phi \le \frac{\varepsilon'}{10\alpha}\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

For $\phi(t, x_{i(t)}) > \Phi$, by the construction of the exponential grid we have $\phi(t, \sigma(t)) \le \mathrm{diam}(\Box_t) \le \frac{\varepsilon'}{10\alpha}\phi(t, x_{i(t)})$. Notice that $\phi(x_{i(t)}, \Box_t) \le \phi(X, \Box_t) + \mathrm{diam}(\Box_t)$, by condition (3). We have,

$$\phi(t, x_{i(t)}) \le \phi(x_{i(t)}, \Box_t) + \mathrm{diam}(\Box_t) \le \phi(X, \Box_t) + 2\mathrm{diam}(\Box_t) \le \phi(t, X) + \frac{2\varepsilon'}{10\alpha}\phi(t, x_{i(t)})$$
$$\Leftrightarrow \phi(t, x_{i(t)}) \le \frac{1}{1-\frac{2\varepsilon'}{10\alpha}}\phi(t, X) \le (1 + \frac{4\varepsilon'}{10\alpha})\phi(t, X),$$

so $\phi(t, \sigma(t)) \le \frac{\varepsilon'}{10\alpha}(1 + \frac{4\varepsilon'}{10\alpha})\phi(t, X)$. Then, we get

$$\mathcal{E} \le \frac{\varepsilon'}{10\alpha}\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})) + \frac{\varepsilon'}{10\alpha}(1 + \frac{4\varepsilon'}{10\alpha})\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D})) \le (\frac{2\varepsilon'}{10} + \frac{4(\varepsilon')^2}{100\alpha})\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$$
$$\le \varepsilon'\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

This implies that $|\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D})) - \mathbf{v}_Y(C)| \le \varepsilon'\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D}))$. □

From the previous lemma, we conclude with the main result establishing the correctness of the algorithm.

Lemma 3.3. *If* GkMedianAlg$_\gamma$ *is used, then* $\mathcal{S} \subset \mathbb{R}^d$ *and* $\boldsymbol{v}_\mathcal{S}(\boldsymbol{q}_u(\mathbf{D})) \le r_u \le (1+\varepsilon)\gamma\boldsymbol{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$. *If* DkMedianAlg$_\gamma$ *is used, then* $\mathcal{S} \subseteq \boldsymbol{q}_u(\mathbf{D})$ *and* $\boldsymbol{v}_\mathcal{S}(\boldsymbol{q}_u(\mathbf{D})) \le r_u \le (2+\varepsilon)\gamma\boldsymbol{v}_{\mathrm{OPT}_{disc}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$.

Proof. By the definition of GkMedianAlg$_\gamma$ and Lemma 3.2, we have $\mathbf{v}_\mathcal{S}(C) \le \gamma\mathbf{v}_{\mathrm{OPT}(C)}(C) \le \gamma\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(C) \le (1+\varepsilon')\gamma\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, and $\mathbf{v}_\mathcal{S}(C) \ge (1-\varepsilon')\mathbf{v}_\mathcal{S}(\boldsymbol{q}(\mathbf{D}))$, so

$$\mathbf{v}_\mathcal{S}(\boldsymbol{q}_u(\mathbf{D})) \le \frac{1}{1-\varepsilon'}\mathbf{v}_\mathcal{S}(C) = r_u \le \gamma\frac{1}{1-\varepsilon'}\mathbf{v}_{\mathrm{OPT}(C)}(C) \le \frac{1+\varepsilon'}{1-\varepsilon'}\gamma\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$$
$$\le (1+4\varepsilon')\gamma\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})) = (1+\varepsilon)\gamma\mathbf{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

The proof using the DkMedianAlg$_\gamma$ algorithm is shown in Appendix B. □

Running time.

Lemma 3.4. $|C| = O(|X|\varepsilon^{-d_u}\log N)$.

Proof. For each $x_i \in X$ there are $2\log(\alpha N) = O(\log N)$ boxes $Q_{i,j}$. For each $V_{i,j}$, we construct $O(\varepsilon^{-d_u})$ cells. In the worst case, C contains a point for every cell. □

The algorithm visits $O(|X|\varepsilon^{-d_u}\log N)$ cells. For each cell $\Box$, we check the condition (3) in $O(|X|)$ time and we compute the arrangement $\mathrm{Arr}(G_\Box)$ in $O(|X|^{d_u}\varepsilon^{-d_u^2}\log^{d_u} N)$ time. For each hyper-rectangle in the complement, we run a query as in Lemma 2.1 in $O(N\log N)$ time. Overall, C is constructed in $O(|X|^{d_u+1}\varepsilon^{-d_u^2-d_u}N\log^{d_u+2} N)$ time, and we get $\mathcal{S}$ in $O(T_\gamma^{\mathrm{med}}(|X|\varepsilon^{-d_u}\log N))$ time.

Theorem 3.5. *Let* $\mathbf{D}$ *be a database instance with* N *tuples,* $\boldsymbol{q}$ *be an acyclic join query over a set of attributes* $\mathbf{A}$ *and* $\mathbf{A}_u \subseteq \mathbf{A}$. *Given a set* $X \subset \mathbb{R}^d$, *a constant* α *such that* $\boldsymbol{v}_X(\boldsymbol{q}_u(\mathbf{D})) \le \alpha\boldsymbol{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, *and a constant parameter* $\varepsilon \in (0,1)$, *there exists an algorithm that computes a set* $\mathcal{S} \subset \mathbb{R}^d$ *of* k *points and a number* r_u *in* $O(|X|^{d_u+1}N\log^{d_u+2} N + T_\gamma^{\mathrm{med}}(|X|\log N))$ *time such that* $\boldsymbol{v}_\mathcal{S}(\boldsymbol{q}_u(\mathbf{D})) \le r_u \le (1+\varepsilon)\gamma\boldsymbol{v}_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$. *There also exists an algorithm that computes a set* $\mathcal{S} \subseteq \boldsymbol{q}_u(\mathbf{D})$ *of* k *points and a number* r_u *with the same running time, such that* $\boldsymbol{v}_\mathcal{S}(\boldsymbol{q}_u(\mathbf{D})) \le r_u \le (2+\varepsilon)\gamma\boldsymbol{v}_{\mathrm{OPT}_{disc}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$.

Algorithm 2: RelClusteringFast$(\boldsymbol{q}, \mathbf{D}, \mathrm{A}_u, X, \alpha, r, \varepsilon)$

```
1  $\varepsilon' = \varepsilon/34$;
2  Lines 2–3 from Algorithm 1;
3  foreach $x_i \in X$ do
4      Lines 5–8 from Algorithm 1;
5      $B = \emptyset$;
6      $\tau = \frac{1}{16|X|(\varepsilon')^{-d_u-1}\log N}$;
7      $M = \frac{3}{(\varepsilon')^2\tau}\log(2N^{10d})$;
8      foreach $\square \in \overline{V}_i$ do
9          if $\phi(x_i, \square) \le \phi(X, \square) + diam(\square)$ then
10             $H_\square = \overline{\pi}_{\mathrm{A}_u}$ (SampleRect$(\boldsymbol{q}, \mathbf{D}, \square, M)$)        (Using Lemma 2.1);
11             $g_\square = |H_\square \setminus (B \cap H_\square)|$;
12             if $\frac{g_\square}{M} \ge 2\tau$ then
13                 $s_\square \leftarrow$ arbitrary tuple in $H_\square \setminus B$;
14                 $n_\square =$ CountRect$(\boldsymbol{q}, \mathbf{D}, \square)$        (Using Lemma 2.1);
15                 $w(s_\square) = \frac{1}{1-\varepsilon'} \cdot \frac{g_\square}{M} \cdot n_\square$;
16                 $C \leftarrow C \cup \{s_\square\}$;    $B \leftarrow B \cup \{\square\}$;
17 $\mathcal{S} =$ GkMedianAlg$_\gamma(C)$        (or $\mathcal{S} =$ DkMedianAlg$_\gamma(C)$ for the discrete version);
18 $r_u = \frac{1+4\varepsilon'}{1-9\varepsilon'}\mathbf{v}_{\mathcal{S}}(C)$;
19 return $(\mathcal{S}, r_u)$;
```

3.2 Fast randomized algorithm

As we show in Section 4.1, the algorithm in the previous section can be used to get a constant approximation for the relational k-median problem. However, the running time is $\Omega(|X|^{d_u+1}N)$. As we will see in the next section $|X| = k^2$, leading to an $\Omega(k^{2d_u+2}N)$ algorithm. In this section, we propose a more involved randomized algorithm that improves the factor $|X|^{d_u+1}N$ to only $|X| \cdot N$. Undoubtedly, the expensive part of the deterministic algorithm is the cardinality estimation $|\boldsymbol{q}_u(\mathbf{D}) \cap (\square \setminus G_\square)|$. Next, we design a faster algorithm to overcome this obstacle. The algorithm constructs exactly the same exponential grid as described above. However, in this algorithm, we use a more involved approach to estimate the weights $w(s_\square)$ faster using random uniform sampling. We use the same notation as in the previous subsection. Let $C = \emptyset$ and let Φ as defined above. In this algorithm, we will characterize each cell we visit as *heavy* or *light*. Let B denote the set of the processed heavy cells. So, we initialize with $B = \emptyset$.

Algorithm. The pseudocode of the algorithm is shown in Algorithm 2. We set $\varepsilon' = \varepsilon/34$. For each $x_i \in X$ and for each cell $\square \in \overline{V}_i$, we check condition (3). If it is satisfied, then we process $\square$. Otherwise, we skip it and continue with the next cell. Let $\square$ be a cell in the exponential grid that the algorithm processes. We compute $n_\square = |\boldsymbol{q}_u(\mathbf{D}) \cap \square|$ and we set $\tau = \frac{1}{16|X|(\varepsilon')^{-d_u-1}\log N}$. Using the algorithm from Lemma 2.1, we sample with replacement a multi-set $H_\square$ of $M = \frac{3}{(\varepsilon')^2\tau}\log(2N^{10d}) = O(|X|(\varepsilon')^{-d_u-3}\log^2 N)$ points from $\boldsymbol{q}_u(\mathbf{D}) \cap \square$ and we set $g_\square = |H_\square \setminus (B \cap H_\square)|$, i.e., the number of samples that are not currently contained in heavy cells we processed. If $\frac{g_\square}{M} \ge 2\tau$, then let $s_\square$ be any of the sampled points in $H_\square \setminus B$ as the representative point of $\square$. We add $s_\square$ in C with weight $w(s_\square) = \frac{1}{1-\varepsilon'} \cdot \frac{g_\square}{M} \cdot n_\square$. We say that all the points in $\boldsymbol{q}_u(\mathbf{D}) \cap (\square \setminus B)$ are *mapped* to $s_\square$. We characterize $\square$ as a heavy cell and we add it to B. Otherwise, if $\frac{g_\square}{M} < 2\tau$, then $\square$ is a light cell, we skip it and continue processing the next cell. An example of the sampling procedure is shown in Figure 4. In the end, after repeating the algorithm for each $x_i \in X$, we have a weighted set C of size $O(|X|\varepsilon^{-d_u}\log n)$. We run the standard algorithm for the weighted k-median problem $\mathsf{GkMedianAlg}_\gamma$ (or discrete k-median problem $\mathsf{DkMedianAlg}_\gamma$) on C to get a set of k centers $\mathcal{S}$. We return the set of centers $\mathcal{S}$. Furthermore, we return $r_u = \frac{1+4\varepsilon'}{1-9\varepsilon'}\mathbf{v}_\mathcal{S}(C)$.

Correctness. Let P_u (heavy tuples) be the tuples in $\boldsymbol{q}_u(\mathbf{D})$ that belong to at least one heavy cell in B and let $\bar{P}_u = \boldsymbol{q}_u(\mathbf{D}) \setminus P_u$ (light tuples), at the end of the algorithm. By construction, every tuple $t \in \boldsymbol{q}_u(\mathbf{D})$, belongs to at least one heavy or light cell. Notice that a point $t \in \boldsymbol{q}_u(\mathbf{D})$ might first lie in a light cell and only later in the algorithm it might be found in a heavy cell. It is straightforward to see that any heavy point $t \in P_u$ is mapped to a point in C. For each $t \in P_u$, let $\square_t$ be the first heavy cell visited by the algorithm such that $t \in \boldsymbol{q}_u(\mathbf{D}) \cap \square_t$. Let $i(t)$ be the index such that $\square_t \in \overline{V}_{i(t)}$. In the deterministic algorithm, we had the assignment function $\sigma(\cdot)$ for all the points in $\boldsymbol{q}_u(D)$. In this algorithm, we only map the points in P_u, so we define a new mapping function $\hat{\sigma}(\cdot)$, i.e., for a point $t \in P_u$, $\hat{\sigma}(t) := s_{\square_t} \in C$. For a cell $\square$ processed by the algorithm, let $B_\square$ be the set of heavy cells found by the algorithm just before $\square$ was processed. For a point $p \in C$, let $\square_p$ be the cell that the algorithm processed while p was added in C, and let $n_p = |\boldsymbol{q}_u(\mathbf{D}) \cap (\square_p \setminus B_{\square_p})|$ be the number of the points mapped to p.

We show the correctness of our algorithm through a number of technical lemmas. We first show a crucial observation. There exists a *charging process* where i) every light tuple (i.e., a tuple that does not belong to a heavy cell) charges $\frac{1}{\varepsilon'}$ heavy tuples that lie to the same cell as the light tuple, and ii) every heavy tuple is charged at most once. This observation is then used to show that for any set of k centers, the k-median error of $\boldsymbol{q}_u(\mathbf{D})$ is close enough to the k-median error with respect to the heavy tuples so we can safely ignore the other tuples. Using this argument, we prove that C is a $9\varepsilon'$-coreset for the heavy points. Using all previous observations, we conclude that $\mathcal{S}$ is a $(1+\varepsilon)\gamma$-approximation for the relational k-median clustering. All missing lemmas and proofs can be found in Appendix C.

Lemma 3.6. *For every point $p \in C$, $n_p \le w(p) \le (1+4\varepsilon')n_p$ with probability at least $1 - \frac{1}{N^{O(1)}}$.*

Proof. Let $p \in C$ be a point in the coreset. By definition $\frac{g_{\square_p}}{M} \ge 2\tau$, so using the Chernoff bound, as shown in [20] (Lemma 2), with probability at least $1 - \frac{1}{N^{O(1)}}$ it holds that

$$(1-\varepsilon')|\boldsymbol{q}_u(\mathbf{D}) \cap (\square_p \setminus B_{\square_p})| \le \frac{g_{\square_p}}{M} n_{\square_p} \le (1+\varepsilon')|\boldsymbol{q}_u(\mathbf{D}) \cap (\square_p \setminus B_{\square_p})|.$$

Hence, $n_p \le w(p) \le \frac{1+\varepsilon'}{1-\varepsilon'} n_p \le (1+4\varepsilon')n_p$ with probability at least $1 - 1/N^{O(1)}$. $\square$

Let L be the set of light cells found by the algorithm. For a cell $\square \in L$, let $\mathcal{P}^L_\square$ be the set of the points in $\boldsymbol{q}_u(\mathbf{D}) \cap \square$ that do not belong in a heavy cell at the time that the algorithm processes $\square$. Notice that a point $p \in \mathcal{P}^L_\square$ might also belong to P_u (points that lie in at least one heavy cell) because p was found inside a heavy cell later in the algorithm. In addition, we note that it might be possible that $\mathcal{P}^L_{\square_j} \cap \mathcal{P}^L_{\square_h} \neq \emptyset$, for $j \neq h$. Furthermore, let $\mathcal{P}^B_\square$ be the set of points in $\boldsymbol{q}_u(\mathbf{D}) \cap \square$ that belong in at least one heavy cell at the time that the algorithm processes $\square$.

In the next technical lemma, we show the existence of a charging process that is later used to estimate the k-median error of $\boldsymbol{q}(\mathbf{D})$ using only the heavy tuples P_u.

Lemma 3.7. *There exists a charging process that works with probability at least $1 - \frac{1}{N^{O(1)}}$ having the following properties. For every cell $\square \in L$, each point $p \in \mathcal{P}^L_\square$* charges $\frac{1}{\varepsilon'}$ *points in $\mathcal{P}^B_\square$, such that, in the end, every point $t \in P_u$ has been charged at most once.*

Proof. Let $L = \{\square_1, \dots, \square_\eta\}$ be sorted in ascending order of $|\mathcal{P}^L_\square|$, i.e., $|\mathcal{P}^L_{\square_j}| \le |\mathcal{P}^L_{\square_{j+1}}|$. It is sufficient to show that for every $j = 1, \dots, \eta$, $|\mathcal{P}^B_{\square_j}| - \frac{1}{\varepsilon'}\sum_{h<j}|\mathcal{P}^L_{\square_h}| \ge \frac{1}{\varepsilon}|\mathcal{P}^L_{\square_j}|$. Indeed, if this inequality holds for every j, then there are always at least $\frac{1}{\varepsilon'}|\mathcal{P}^L_{\square_j}|$ uncharged points in $\mathcal{P}^B_{\square_j}$, and we can charge each point in $\mathcal{P}^L_{\square_j}$ to $\frac{1}{\varepsilon'}$ points in $\mathcal{P}^B_{\square_j}$.

Recall that $\square_j \in L$ is a light cell because the algorithm found that the ratio $\frac{|H_{\square_j} \cap \mathcal{P}^L_{\square_j}|}{M} \le 2\tau$. In this case, from the Chernoff bound (Lemma 2 in the full version of [20]) we have $\frac{|\mathcal{P}^L_{\square_j}|}{|\mathcal{P}^B_{\square_j}|+|\mathcal{P}^L_{\square_j}|} \le 4\tau$ with probability at least $1 - \frac{1}{N^{O(1)}}$. Solving the inequality with respect to $|\mathcal{P}^B_{\square_j}|$, we get $|\mathcal{P}^B_{\square_j}| \ge \frac{1-4\tau}{4\tau}|\mathcal{P}^L_{\square_j}| \ge \frac{1}{8\tau}|\mathcal{P}^L_{\square_j}|$, because $\tau < \frac{1}{8}$. Hence, $|\mathcal{P}^B_{\square_j}| \ge 2|X|(\varepsilon')^{-d_u-1}\log(N) \cdot |\mathcal{P}^L_{\square_j}|$.

Next, we find an upper bound for $\frac{1}{\varepsilon'}\sum_{h<j}|\mathcal{P}^L_{\square_h}|$. Because of sorting in ascending order of $|\mathcal{P}^L_{\square_j}|$, we have $\frac{1}{\varepsilon'}\sum_{h<j}|\mathcal{P}^L_{\square_h}| \le \frac{j}{\varepsilon'}|\mathcal{P}^L_{\square_j}| \le \frac{|L|}{\varepsilon'}|\mathcal{P}^L_{\square_j}| \le |X|(\varepsilon')^{-d_u-1}\log(N) \cdot |\mathcal{P}^L_{\square_j}|$.

Hence, we conclude that

$$|\mathcal{P}^B_{\square_j}| - \frac{1}{\varepsilon'}\sum_{h<j}|\mathcal{P}^L_{\square_j}| \ge 2|X|(\varepsilon')^{-d_u-1}\log(N)|\mathcal{P}^L_{\square_j}| - |X|(\varepsilon')^{-d_u-1}\log(N)|\mathcal{P}^L_{\square_j}| \ge \frac{1}{\varepsilon'}|\mathcal{P}^L_{\square_j}|.$$

□

For each point $p \in \bar{P}_u$, let $\square_{i(p)} \in L$ be the cell in L such that $p \in \square_{i(p)}$ and p charges $\frac{1}{\varepsilon'}$ points in $\mathcal{P}^B_{\square_{i(p)}}$, as shown in Lemma 3.7. Let $t_{j_1(p)}, \dots, t_{j_{1/\varepsilon'}(p)}$ be these points in $\mathcal{P}^B_{\square_{i(p)}}$.

Next, we show that the k-median error with respect to the heavy points P_u approximates the k-median error of all tuples in $\boldsymbol{q}_u(\mathbf{D})$.

Lemma 3.8. *Let Y be an arbitrary set of k points in $\mathbb{R}^{d_u}$. It holds that $\mathbf{v}_Y(\bar{P}_u) \le \varepsilon'\mathbf{v}_Y(P_u) + \varepsilon'\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D}))$ and $\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D})) \le (1+4\varepsilon')\mathbf{v}_Y(P_u)$ with probability at least $1 - \frac{1}{N^{O(1)}}$.*

Proof. We start showing the first inequality $\mathbf{v}_Y(\bar{P}_u) \le \varepsilon'\mathbf{v}_Y(P_u) + \varepsilon'\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D}))$. We have $\mathbf{v}_Y(\bar{P}_u) = \sum_{p\in\bar{P}_u}\phi(p,Y)$. For a point $p \in \bar{P}_u$ and each $h \le \frac{1}{\varepsilon'}$, by triangle inequality, we have

$$\phi(p,Y) \le \phi(t_{j_h(p)}, Y) + \phi(p, t_{j_h(p)}).$$

Taking the sum of these $\frac{1}{\varepsilon'}$ inequalities, we have $\phi(p,Y) \le \varepsilon'\sum_{h=1}^{1/\varepsilon'}\phi(t_{j_h(p)},Y) + \varepsilon'\sum_{h=1}^{1/\varepsilon'}\phi(p,t_{j_h(p)})$, so we get

$$\mathbf{v}_Y(\bar{P}_u) \le \varepsilon'\sum_{p\in\bar{P}_u}\sum_{h=1}^{1/\varepsilon'}\phi(t_{j_h(p)},Y) + \varepsilon'\sum_{p\in\bar{P}_u}\sum_{h=1}^{1/\varepsilon'}\phi(p,t_{j_h(p)}).$$

From Lemma 3.7, we proved that any $t \in P_u$ is charged by at most one point in $\bar{P}_u$, so the first term in the sum can be bounded as $\varepsilon'\sum_{p\in\bar{P}_u}\sum_{h=1}^{1/\varepsilon'}\phi(t_{j_h(p)},Y) \le \varepsilon'\sum_{t\in P_u}\phi(t,Y) = \varepsilon'\mathbf{v}_Y(P_u)$. In Appendix C we bound the second term in the sum showing that $\varepsilon'\sum_{p\in\bar{P}_u}\sum_{h=1}^{1/\varepsilon'}\phi(p,t_{j_h(p)}) \le (\varepsilon')^2\mathbf{v}_Y(\boldsymbol{q}(\mathbf{D})) < \varepsilon'\mathbf{v}_Y(\boldsymbol{q}(\mathbf{D}))$. Hence, the first inequality follows.

For the second inequality we have,

$$\mathbf{v}_Y(P_u) = \mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D})) - \mathbf{v}_Y(\bar{P}_u) \ge \mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D})) - \varepsilon'\mathbf{v}_Y(P_u) - \varepsilon'\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D})),$$

so $\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D})) \le \frac{1+\varepsilon'}{1-\varepsilon'}\mathbf{v}_Y(P_u) \le (1+4\varepsilon')\mathbf{v}_Y(P_u)$. □

Using the inequalities in Lemma 3.8 and Lemma 3.6, we follow the proof of Lemma 3.2 and we show the next result.

Lemma 3.9. *C is an $9\varepsilon'$-coreset of P_u with probability at least $1 - \frac{1}{N^{O(1)}}$.*

Proof. Let Y be an arbitrary set of k points in $\mathbb{R}^{d_u}$, as we had in Lemma 3.2. From Lemma 3.6, let $\varepsilon_p \in [0, 4\varepsilon']$ be a real number such that $w(p) = (1+\varepsilon_p)n_p$, for $p \in C$. With probability at least $1 - \frac{1}{N^{O(1)}}$, we have,

$$\mathcal{E} = |\mathbf{v}_Y(P_u) - \mathbf{v}_Y(C)| = |\sum_{t\in P_u}\phi(t,Y) - \sum_{p\in C} w(p)\phi(p,Y)| = |\sum_{t\in P_u}\phi(t,Y) - \sum_{p\in C}(1+\varepsilon_p)n_p\phi(p,Y)|$$
$$= |\sum_{t\in P_u}\phi(t,Y) - \sum_{t\in P_u}(1+\varepsilon_{\hat{\sigma}(t)})\phi(\hat{\sigma}(t),Y)| \le \sum_{t\in P_u}|\phi(t,Y) - (1+\varepsilon_{\hat{\sigma}(t)})\phi(\hat{\sigma}(t),Y)|.$$

This inequality also follows from Lemma 3.6. Indeed, each $\hat{\sigma}(t)$ has a weight that is $(1+\varepsilon_{\hat{\sigma}(t)})$ times larger than the number of points in P_u that are assigned to $\hat{\sigma}(t)$. We have,

$$\mathcal{E} \le \sum_{t\in P_u}|\phi(t,Y) - (1+\varepsilon_{\hat{\sigma}(t)})\phi(\hat{\sigma}(t),Y)| \le \sum_{t\in P_u}|\phi(t,Y) - \phi(\hat{\sigma}(t),Y)| + 4\varepsilon'\sum_{t\in P_u}\phi(\hat{\sigma}(t),Y)$$
$$\le \sum_{t\in P_u}\phi(t,\hat{\sigma}(t)) + 4\varepsilon'\mathbf{v}_Y(P_u).$$

Following the proof of Lemma 3.2 we have $\sum_{t\in P_u}\phi(t,\hat{\sigma}(t)) \le \varepsilon'\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D}))$. From Lemma 3.8 we get $\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D})) \le (1+4\varepsilon')\mathbf{v}_Y(P_u)$. We conclude that $\mathcal{E} \le (5\varepsilon' + 4(\varepsilon')^2)\mathbf{v}_Y(P_u) \le 9\varepsilon'\mathbf{v}_Y(P_u)$. □

Form Lemma 3.9, we conclude to the main result.

Lemma 3.10. *If* GkMedianAlg$_\gamma$ *is used, then* $\mathcal{S} \subset \mathbb{R}^d$ *and* $\boldsymbol{v}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D})) \le r_u \le (1+\varepsilon)\gamma\boldsymbol{v}_{\mathsf{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$, *with probability at least* $1-\frac{1}{N^{O(1)}}$. *If* DkMedianAlg$_\gamma$ *is used, then* $\mathcal{S} \subseteq \boldsymbol{q}_u(\mathbf{D})$ *and* $\boldsymbol{v}_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \le r_u \le (2+\varepsilon)\gamma\boldsymbol{v}_{\mathsf{OPT}_{disc}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, *with probability at least* $1-\frac{1}{N^{O(1)}}$.

Proof. We first consider the case where GkMedianAlg$_\gamma$ is used. From Lemma 3.9, we have that for any set of k points Y in $\mathbb{R}^{d_u}$, $(1-9\varepsilon')\mathbf{v}_Y(P_u) \le \mathbf{v}_Y(C) \le (1+9\varepsilon')\mathbf{v}_Y(P_u)$, with probability at least $1-\frac{1}{N^{O(1)}}$. By definition,

$$\mathbf{v}_{\mathcal{S}}(C) \le \gamma\mathbf{v}_{\mathsf{OPT}(C)}(C) \le \gamma\mathbf{v}_{\mathsf{OPT}(P_u)}(C) \le (1+9\varepsilon')\gamma\mathbf{v}_{\mathsf{OPT}(P_u)}(P_u). \tag{4}$$

The last inequality follows by the definition of the coreset for $Y = \mathsf{OPT}(P_u)$. Since $P_u \subseteq \boldsymbol{q}_u(\mathbf{D})$ and $\mathsf{OPT}(P_u), \mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D})) \subset \mathbb{R}^d$, it also holds that $\mathbf{v}_{\mathsf{OPT}(P_u)}(P_u) \le \mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$.

From Lemma 3.8 (for $Y = \mathcal{S}$) we have $\mathbf{v}_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \le (1+4\varepsilon')\mathbf{v}_{\mathcal{S}}(P_u)$. Hence,

$$\mathbf{v}_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \le (1+4\varepsilon')\mathbf{v}_{\mathcal{S}}(P_u) \le \frac{1+4\varepsilon'}{1-9\varepsilon'}\mathbf{v}_{\mathcal{S}}(C) = r_u \le \frac{(1+4\varepsilon')(1+9\varepsilon')}{1-9\varepsilon'}\gamma\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$$
$$\le (1+34\varepsilon')\gamma\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})) = (1+\varepsilon)\gamma\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

The proof using the DkMedianAlg$_\gamma$ algorithm is shown in Appendix C. □

Running time. The total number of cells that the algorithm processes is $O(|X|\varepsilon^{-d_u}\log N)$. In each cell, we take $M = O(|X|\varepsilon^{-d_u-3}\log^2 N)$ samples. Using Lemma 2.1, for each cell we spend $O(N + M\log N)$ time to get the set of samples. For each sample, we check whether it belongs to a heavy cell in B. This can be checked trivially in $O(|B|) = O(|X|\varepsilon^{-d_u}\log N)$ time, or in $O(\log^{d_u}(|X|\varepsilon^{-d_u}\log N))$ time using a dynamic geometric data structure for stabbing queries [5, 52]. Finally, the standard algorithm for k-median clustering on $O(|X|\varepsilon^{-d_u}\log N)$ points takes $O(T_\gamma^{\mathsf{med}}(|X|\varepsilon^{-d_u}\log N))$ time. The overall time of the algorithm is

$$O\left(N|X|\varepsilon^{-d_u}\log(N) + |X|^2\varepsilon^{-2d_u-3}\log^3(N)\min\left\{|X|\varepsilon^{-d_u}\log(N), \log^{d_u}(|X|)\right\} + T_\gamma^{\mathsf{med}}(|X|\varepsilon^{-d_u}\log N)\right).$$

Algorithm 3: Rel-k-Median$(\boldsymbol{q}, \mathbf{D}, \mathbf{R}, \varepsilon, u)$

```
1  if $u$ is a leaf node then
2      Let $R_j$ be a relation in $\mathbf{R}$ such that $A_u \in \mathbf{A}_j$;
3      Construct join tree $T$ of $\boldsymbol{q}$ with $R_j$ in the root;
4      Run Yannakakis algorithm on $T$ to compute $c(h) = |\{t \in \boldsymbol{q}(\mathbf{D}) \mid \pi_{\mathbf{A}_j}(t) = h\}|, \forall h \in R_j$;
5      $H_u = \pi_{A_u}(R_j)$;
6      foreach $p \in H_u$ do
7          $w(p) = \sum_{h \in R_j, \pi_{A_u}(h)=p} c(h)$;
8      $\mathcal{S}_u = \mathsf{GkMedianAlg}_\gamma(H_u)$;    $r_u = \mathbf{v}_{\mathcal{S}_u}(H_u)$;
9  else
10     Let $v$ and $z$ be the children of $u$;
11     $X = \mathcal{S}_v \times \mathcal{S}_z$;    $r = r_v + r_z$;
12     $(\mathcal{S}_u, r_u) \leftarrow \mathsf{RelClusteringFast}(\boldsymbol{q}, \mathbf{D}, \mathbf{A}_u, X, (1+\varepsilon)\gamma\sqrt{2}, r, \varepsilon)$;
13 return $(\mathcal{S}_u, r_u)$;
```

Theorem 3.11. *Let* $\mathbf{D}$ *be a database instance with* N *tuples,* $\boldsymbol{q}$ *be an acyclic join query over a set of attributes* $\mathbf{A}$*, and* $\mathbf{A}_u \subseteq \mathbf{A}$*. Given a set* $X \subset \mathbb{R}^d$*, a constant* α *such that* $\boldsymbol{v}_X(\boldsymbol{q}_u(\mathbf{D})) \leq \alpha \boldsymbol{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$*, and a constant parameter* $\varepsilon \in (0,1)$*, there exists an algorithm that computes a set* $\mathcal{S} \subset \mathbb{R}^d$ *of* k *points and a number* r_u *in* $O(|X| N \log N + |X|^2 \log^3(N) \min\{\log^{d_u}(|X|), |X| \log N\} + T_\gamma^{\mathsf{med}}(|X| \log N))$ *such that* $\boldsymbol{v}_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (1+\varepsilon)\gamma \boldsymbol{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$*, with probability at least* $1 - \frac{1}{N^{O(1)}}$*. There also exists an algorithm that computes a set* $\mathcal{S} \subseteq \boldsymbol{q}_u(\mathbf{D})$ *of* k *points and a number* r_u *with the same running time, such that* $\boldsymbol{v}_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (2+\varepsilon)\gamma \boldsymbol{v}_{\mathsf{OPT}_{disc}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$*, with probability at least* $1 - \frac{1}{N^{O(1)}}$*.*

4 EFFICIENT ALGORITHMS

We use the results from Section 3 to describe a complete algorithm for the relational k-median clustering. In the previous section, we saw how we can get a $(1+\varepsilon)\gamma$-approximation algorithm if a set X along with a number r such that $\mathbf{v}_X(\boldsymbol{q}(\mathbf{D})) \leq r \leq O(1)\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}(\mathbf{D}))}\boldsymbol{q}(\mathbf{D})$ are given. In this section, we efficiently compute the set X and the value r, and present the complete algorithms for the relational k-median clustering.

4.1 Acyclic queries

We construct a balanced binary tree $\mathcal{T}$ such that the j-th leaf node stores the j-th attribute $A_j \in \mathbf{A}$ (the order is arbitrary). For a node u of $\mathcal{T}$, let $\mathcal{T}_u$ be the subtree of $\mathcal{T}$ rooted at u. Let $\mathbf{A}_u$ be the set of attributes stored in the leaf nodes of $\mathcal{T}_u$. For every node u, our algorithm computes a set $\mathcal{S}_u$ of cardinality k and a real positive number r_u such that

$$\mathbf{v}_{\mathcal{S}_u}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (1+\varepsilon)\gamma \mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})), \tag{5}$$

for the relational k-median clustering. For the discrete relational k-median clustering, our algorithm computes a set $\mathcal{S}_u$ of cardinality k and a real positive number r_u such that

$$\mathbf{v}_{\mathcal{S}_u}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (2+\varepsilon)\gamma \mathbf{v}_{\mathsf{OPT}_{\mathsf{disc}}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})), \tag{6}$$

The definition of $\boldsymbol{q}_u(\mathbf{D})$ is the same as in the previous section: $\boldsymbol{q}_u(\mathbf{D}) = \bar{\pi}_{\mathbf{A}_u}(\boldsymbol{q}(\mathbf{D}))$. Similarly, we use the notation $d_u = |\mathbf{A}_u|$. For a finite set $Y \subset \mathbb{R}^d$ and a point $t \in \mathbb{R}^d$ let $\mathcal{N}(Y, t) =$

$\arg\min_{y\in Y}\phi(t,y)$, be the nearest neighbor of t in Y. Recall that for any set $Y\subset\mathbb{R}^{d_u}$, $\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D}))=\sum_{t\in\boldsymbol{q}(\mathbf{D})}\phi(\pi_{\mathbf{A}_u}(t),Y)=\sum_{t\in\boldsymbol{q}(\mathbf{D})}\sqrt{\sum_{A_j\in\mathbf{A}_u}(\pi_{A_j}(t)-\pi_{A_j}(\mathcal{N}(Y,\pi_{\mathbf{A}_u}(t))))^2}$.

The algorithm we propose works bottom-up on tree $\mathcal{T}$. If u has children v,z we use the precomputed $\mathcal{S}_v,\mathcal{S}_z$ and r_v,r_z to compute $\mathcal{S}_u$ and r_u.

Algorithm. First, we run the Yannakakis algorithm [56] on $\boldsymbol{q}(\mathbf{D})$ and we keep only the non-dangling tuples in $\mathbf{D}$, i.e., tuples in $\mathbf{D}$ that belong to at least one join result $\boldsymbol{q}(\mathbf{D})$. For any node u of $\mathcal{T}$, in Algorithm 3 we show the pseudocode for the geometric version of relational k-median clustering, computing $\mathcal{S}_u$ and r_u.

Let u be a leaf node where $\mathbf{A}_u$ contains one attribute $A_u\in\mathbf{A}$. This is the only case that we try to implicitly construct $\boldsymbol{q}_u(\mathbf{D})$ as a set of $O(N)$ points in $\mathbb{R}^1$ with cardinalities (weights). More specifically, we construct the weighted set of points in $\mathbb{R}^1$, $H_u=\pi_{A_u}(\boldsymbol{q}(\mathbf{D}))$ such that $p\in H_u$ has weight $w(p)=|\{t\in\boldsymbol{q}(\mathbf{D})\mid\pi_{A_u}(t)=p\}|$. Notice that H_u is a set and not a multi-set. We can compute H_u and the weight function $w(\cdot)$ as follows. Let R_j be an arbitrary relation from $\mathbf{R}$ that contains the attribute A_u. We use the counting version of Yannakakis algorithm to count the number of times that a tuple belongs in $\boldsymbol{q}(\mathbf{D})$. More specifically, we construct the join tree for $\boldsymbol{q}$ and choose R_j as its root. Using Yannakakis algorithm, for every tuple h in the root R_j we compute $c(h)=|\{t\in\boldsymbol{q}(\mathbf{D})\mid\pi_{\mathbf{A}_j}(t)=h\}|$. By grouping together tuples from R_j with the same value on attribute A_u, we compute H_u and the weight function $w(\cdot)$. More specifically, we set $H_u=\pi_{A_u}(R_j)$ and for each $p\in H_u$, we set $w(p)=\sum_{h\in R_j,\pi_{A_u}(h)=p}c(h)$. Finally, we run the standard weighted k-median $\mathsf{GkMedianAlg}_\gamma$ (or discrete k-median $\mathsf{DkMedianAlg}_\gamma$) algorithm on the weighted set H_u and we get a set of k centers $\mathcal{S}_u$. We also compute $r_u=\mathbf{v}_{\mathcal{S}_u}(H_u)$.

Next, assume that u is an inner node of $\mathcal{T}$ with two children v,z. The algorithm sets $r=r_v+r_z$ and $X=\mathcal{S}_v\times\mathcal{S}_z$. Then, we run the algorithm from Theorem 3.11 (or Theorem 3.5) using X and r as input and compute $\mathcal{S}_u$ and r_u. More specifically, for the relational k-median clustering we call $\mathsf{RelClusteringFast}(\boldsymbol{q},\mathbf{D},\mathbf{A}_u,X,(1+\varepsilon)\gamma\sqrt{2},r,\varepsilon)$, while for the discrete relational k-median clustering we call $\mathsf{RelClusteringFast}(\boldsymbol{q},\mathbf{D},\mathbf{A}_u,X,2(2+\varepsilon)\gamma\sqrt{2},r,\varepsilon)$. Let ρ be the root of $\mathcal{T}$. We return the set $\mathcal{S}=\mathcal{S}_\rho$.

Correctness. As we explained above, for the leaf nodes, the algorithm is simple. Let u be an intermediate node and let v and z be the two child nodes of u. Assuming that $\mathcal{S}_v,r_v$ and $\mathcal{S}_z,r_z$ satisfy the Equation (5) (resp. Equation (6) for the discrete relational k-median), we show that $\mathcal{S}_u$ and r_u satisfy Equation (5) (resp. Equation (6) for the discrete relational k-median). If we prove that $\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D}))\leq r_u\leq\alpha\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, then the correctness follows from Theorem 3.11. The full proof of the next lemma can be found in Appendix D.

Lemma 4.1. *If* $\mathsf{GkMedianAlg}_\gamma$ *is used, then* $\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D}))\leq r_u\leq\alpha\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, *for* $\alpha=(1+\varepsilon)\gamma\sqrt{2}$. *If* $\mathsf{DkMedianAlg}_\gamma$ *is used, then* $\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D}))\leq r_u\leq\alpha\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, *for* $\alpha=2(2+\varepsilon)\gamma\sqrt{2}$.

Proof. We focus on the case where $\mathsf{GkMedianAlg}_\gamma$ is used. Let $O_v=\pi_{\mathbf{A}_v}(\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D})))$, and $O_z=\pi_{\mathbf{A}_z}(\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D})))$. We define $O=O_v\times O_z$. Notice that $\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))\subseteq O$ so $\mathbf{v}_O(\boldsymbol{q}_u(\mathbf{D}))\leq\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$. We have,

$$\begin{aligned}\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D}))&=\sum_{t\in\boldsymbol{q}(\mathbf{D})}||\pi_{\mathbf{A}_u}(t)-\mathcal{N}(X,\pi_{\mathbf{A}_u}(t))||\\&=\sum_{t\in\boldsymbol{q}(\mathbf{D})}\sqrt{||\pi_{\mathbf{A}_v}(t)-\pi_{\mathbf{A}_v}(\mathcal{N}(X,\pi_{\mathbf{A}_u}(t)))||^2+||\pi_{\mathbf{A}_z}(t)-\pi_{\mathbf{A}_z}(\mathcal{N}(X,\pi_{\mathbf{A}_u}(t)))||^2}\\&\leq\sum_{t\in\boldsymbol{q}(\mathbf{D})}||\pi_{\mathbf{A}_v}(t)-\mathcal{N}(\mathcal{S}_v,\pi_{\mathbf{A}_v}(t))||+\sum_{t\in\boldsymbol{q}(\mathbf{D})}||\pi_{\mathbf{A}_z}(t)-\mathcal{N}(\mathcal{S}_z,\pi_{\mathbf{A}_z}(t))||\leq r_v+r_z\end{aligned}$$

$$\leq (1+\varepsilon)\gamma\left(\sum_{t\in \boldsymbol{q}(\mathbf{D})}||\pi_{\mathrm{A}_v}(t)-\mathcal{N}(\mathsf{OPT}(\boldsymbol{q}_v(\mathbf{D})),\pi_{\mathrm{A}_v}(t))||+\sum_{t\in \boldsymbol{q}(\mathbf{D})}||\pi_{\mathrm{A}_z}(t)-\mathcal{N}(\mathsf{OPT}(\boldsymbol{q}_z(\mathbf{D})),\pi_{\mathrm{A}_z}(t))||\right)$$

$$\leq (1+\varepsilon)\gamma\sqrt{2}\sum_{t\in \boldsymbol{q}(\mathbf{D})}\sqrt{||\pi_{\mathrm{A}_v}(t)-\pi_{\mathrm{A}_v}(\mathcal{N}(O,\pi_{\mathrm{A}_u}(t)))||^2+||\pi_{\mathrm{A}_z}(t)-\pi_{\mathrm{A}_z}(\mathcal{N}(O,\pi_{\mathrm{A}_u}(t)))||^2}$$

$$=(1+\varepsilon)\gamma\sqrt{2}\sum_{t\in \boldsymbol{q}(\mathbf{D})}||\pi_{\mathrm{A}_u}(t)-\mathcal{N}(O,\pi_{\mathrm{A}_u}(t))||\leq (1+\varepsilon)\gamma\sqrt{2}\cdot \mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

Similarly, if $\mathsf{DkMedianAlg}_\gamma$ is used, we have $\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D}))\leq r_u\leq 2(2+\varepsilon)\gamma\sqrt{2}\cdot\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$. □

Putting everything together, we conclude with the next theorem.

THEOREM 4.2. *Given an acyclic join query $\boldsymbol{q}$, a database instance $\mathbf{D}$, a parameter k, and a constant parameter $\varepsilon\in(0,1)$, there exists an algorithm that computes a set $\mathcal{S}\subset\mathbb{R}^d$ of k points such that $\boldsymbol{v}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))\leq(1+\varepsilon)\gamma\boldsymbol{v}_{\mathsf{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$, with probability at least $1-\frac{1}{N^{O(1)}}$. The running time of the algorithm is $O(Nk^2\log(N)+k^4\log^3(N)\min\{\log^d(k),k^2\log N\}+T_\gamma^{\mathsf{med}}(k^2\log N))$. Furthermore, there exists an algorithm that computes a set $\mathcal{S}'\subseteq\boldsymbol{q}(\mathbf{D})$ of k points in the same time such that $\boldsymbol{v}_{\mathcal{S}'}(\boldsymbol{q}(\mathbf{D}))\leq(2+\varepsilon)\gamma\boldsymbol{v}_{\mathsf{OPT}_{disc}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$, with probability at least $1-\frac{1}{N^{O(1)}}$.*

We extend the result of Theorem 4.3 for the relational k-means clustering in Appendix A. Using Theorem 3.5 instead of Theorem 3.11 we can get a deterministic algorithm for the relational k-median clustering problem with the same approximation guarantees that runs in $O(k^{2d+2}N\log^{d+2}N+T_\gamma^{\mathsf{med}}(k^2\log N))$ time.

4.2 Cyclic queries

We use the notion of fractional hypertree width [29] and use a standard procedure to extend our algorithms to every (cyclic) join query $\boldsymbol{q}$. For a cyclic join query $\boldsymbol{q}$, we convert it to an equivalent acyclic query such that each relation is the result of a (possibly cyclic) join query with fractional edge cover at most $\mathsf{fhw}(\boldsymbol{q})$. We evaluate the (possibly cyclic) queries to derive the new relations and then apply the algorithm from Section 4.1 on the new acyclic query. Since it is a typical method in database theory, we give the details in Appendix E.

THEOREM 4.3. *Given a join query $\boldsymbol{q}$, a database instance $\mathbf{D}$, a parameter k, and a constant parameter $\varepsilon\in(0,1)$, there exists an algorithm that computes a set $\mathcal{S}\subset\mathbb{R}^d$ of k points such that $\boldsymbol{v}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))\leq(1+\varepsilon)\gamma\boldsymbol{v}_{\mathsf{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$, with probability at least $1-\frac{1}{N^{O(1)}}$. The running time of the algorithm is $O(N^{\mathsf{fhw}}k^2\log(N)+k^4\log^3(N)\min\{\log^d(k),k^2\log N\}+T_\gamma^{\mathsf{med}}(k^2\log N))$. Furthermore, there exists an algorithm that computes a set $\mathcal{S}'\subseteq\boldsymbol{q}(\mathbf{D})$ of k points in the same time such that $\boldsymbol{v}_{\mathcal{S}'}(\boldsymbol{q}(\mathbf{D}))\leq(2+\varepsilon)\gamma\boldsymbol{v}_{\mathsf{OPT}_{disc}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$, with probability at least $1-\frac{1}{N^{O(1)}}$. The same results hold for the relational k-means clustering problem.*

5 CONCLUSION

In this paper we propose improved approximation algorithms for the relational k-median and k-means clustering. There are multiple interesting open problems derived from this work. It is interesting to check whether our geometric algorithms can be extended other clustering objective functions such as the sum of radii clustering. It is also interesting to use the ideas proposed in this paper to design clustering algorithms on the results of more complex queries such as conjunctive queries with inequalities or conjunctive queries with negation. Finally, someone can study relational clustering under different distance functions such as the Hamming, Jaccard or the cosine distance.

REFERENCES

[1] https://db-engines.com/en/ranking_categories.

[2] Kaggle machine learning and data science survey. https://www.kaggle.com/datasets/kaggle/kaggle-survey-2018, 2018. [Online; accessed 14-May-2024].

[3] Mahmoud Abo-Khamis, Sungjin Im, Benjamin Moseley, Kirk Pruhs, and Alireza Samadian. A relational gradient descent algorithm for support vector machine training. In *Symposium on Algorithmic Principles of Computer Systems (APOCS)*, pages 100–113. SIAM, 2021.

[4] Mahmoud Abo Khamis, Hung Q Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. In-database learning with sparse tensors. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 325–340, 2018.

[5] Pankaj K Agarwal, Jeff Erickson, et al. Geometric range searching and its relatives. *Contemporary Mathematics*, 223:1–56, 1999.

[6] Pankaj K Agarwal, Aryan Esmailpour, Xiao Hu, Stavros Sintos, and Jun Yang. Computing a well-representative summary of conjunctive query results. *Proceedings of the ACM on Management of Data*, 2(5):1–27, 2025.

[7] Pankaj K Agarwal and Micha Sharir. Arrangements and their applications. In *Handbook of computational geometry*, pages 49–119. Elsevier, 2000.

[8] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 15–28. Springer, 2009.

[9] Grant Anderson and Bernhard Pfahringer. Clustering relational data based on randomized propositionalization. In *Inductive Logic Programming: 17th International Conference, ILP 2007, Corvallis, OR, USA, June 19-21, 2007, Revised Selected Papers 17*, pages 39–48. Springer, 2008.

[10] Marcelo Arenas, Timo Camillo Merkl, Reinhard Pichler, and Cristian Riveros. Towards tractability of the diversity of query answers: Ultrametrics to the rescue. *arXiv preprint arXiv:2408.01657*, 2024.

[11] David Arthur and Sergei Vassilvitskii. k-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

[12] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 21–29, 2001.

[13] Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. *SIAM Journal on Computing*, 42(4):1737–1767, 2013.

[14] Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1119–1127, 2018.

[15] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means+. *Proceedings of the VLDB Endowment*, 5(7), 2012.

[16] Vladimir Braverman, Gereon Frahling, Harry Lang, Christian Sohler, and Lin F Yang. Clustering high dimensional dynamic data streams. In *International Conference on Machine Learning*, pages 576–585. PMLR, 2017.

[17] Ronald L Breiger, Scott A Boorman, and Phipps Arabie. An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. *Journal of mathematical psychology*, 12(3):328–383, 1975.

[18] Nofar Carmeli, Nikolaos Tziavelis, Wolfgang Gatterbauer, Benny Kimelfeld, and Mirek Riedewald. Tractable orders for direct access to ranked answers of conjunctive queries. *ACM Transactions on Database Systems*, 48(1):1–45, 2023.

[19] Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 1–10, 1999.

[20] Jiaxiang Chen, Qingyuan Yang, Ruomin Huang, and Hu Ding. Coresets for relational data and the applications. *Advances in Neural Information Processing Systems*, 35:434–448, 2022.

[21] Zhaoyue Cheng and Nick Koudas. Nonlinear models over normalized data. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1574–1577. IEEE, 2019.

[22] Zhaoyue Cheng, Nick Koudas, Zhe Zhang, and Xiaohui Yu. Efficient construction of nonlinear models over normalized data. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1140–1151. IEEE, 2021.

[23] Ryan Curtin, Benjamin Moseley, Hung Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. Rk-means: Fast clustering for relational data. In *International Conference on Artificial Intelligence and Statistics*, pages 2742–2752. PMLR, 2020.

[24] Shaleen Deep, Xiao Hu, and Paraschos Koutris. Ranked enumeration of join queries with projections. *Proceedings of the VLDB Endowment*, 15(5):1024–1037, 2022.

[25] Shaleen Deep and Paraschos Koutris. Ranked enumeration of conjunctive query results. In *24th International Conference on Database Theory*, 2021.

[26] Alina Ene, Sungjin Im, and Benjamin Moseley. Fast clustering using mapreduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 681–689, 2011.

[27] Aryan Esmailpour and Stavros Sintos. Improved approximation algorithms for relational clustering. *arXiv preprint arXiv:2409.18498*, 2024.

[28] Hichem Frigui, Cheul Hwang, and Frank Chung-Hoon Rhee. Clustering and aggregation of relational data with applications to image database categorization. *Pattern Recognition*, 40(11):3053–3068, 2007.

[29] G. Gottlob, G. Greco, and F. Scarcello. Treewidth and hypertree width. *Tractability: Practical Approaches to Hard Problems*, 1, 2014.

[30] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE transactions on knowledge and data engineering*, 15(3):515–528, 2003.

[31] Dan Halperin and Micha Sharir. Arrangements. In *Handbook of discrete and computational geometry*, pages 723–762. Chapman and Hall/CRC, 2017.

[32] Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 126–134, 2005.

[33] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300, 2004.

[34] Timothy C Havens. *Clustering in relational data and ontologies*. University of Missouri-Columbia, 2010.

[35] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 10–18, 2002.

[36] Mahmoud Abo Khamis, Ryan R Curtin, Benjamin Moseley, Hung Q Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. Functional aggregate queries with additive inequalities. *ACM Transactions on Database Systems (TODS)*, 45(4):1–41, 2020.

[37] Mahmoud Abo Khamis, Hung Q Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. Ac/dc: in-database learning thunderstruck. In *Proceedings of the second workshop on data management for end-to-end machine learning*, pages 1–10, 2018.

[38] Mathias Kirsten and Stefan Wrobel. Relational distance-based clustering. In *International Conference on Inductive Logic Programming*, pages 261–270. Springer, 1998.

[39] Arun Kumar, Jeffrey Naughton, and Jignesh M Patel. Learning generalized linear models over normalized data. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1969–1984, 2015.

[40] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *proceedings of the forty-fifth annual ACM symposium on theory of computing*, pages 901–910, 2013.

[41] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[42] Bo Long, Zhongfei Zhang, and S Yu Philip. *Relational data clustering: models, algorithms, and applications*. CRC Press, 2010.

[43] Timo Camillo Merkl, Reinhard Pichler, and Sebastian Skritek. Diversity of answers to conjunctive queries. In *26th International Conference on Database Theory (ICDT 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

[44] Benjamin Moseley, Kirk Pruhs, Alireza Samadian, and Yuyan Wang. Relational algorithms for k-means clustering. In *48th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 198, pages 97:1–97:21, 2021.

[45] Jennifer Neville, Micah Adler, and David Jensen. Clustering relational data using attribute and link information. In *Proceedings of the text mining and link analysis workshop, 18th international joint conference on artificial intelligence*, pages 9–15. San Francisco, CA: Morgan Kaufmann Publishers, 2003.

[46] Hung Q Ngo, Ely Porat, Christopher Ré, and Atri Rudra. Worst-case optimal join algorithms. *Journal of the ACM (JACM)*, 65(3):1–40, 2018.

[47] Hung Q Ngo, Christopher Ré, and Atri Rudra. Skew strikes back: New developments in the theory of join algorithms. *Acm Sigmod Record*, 42(4):5–16, 2014.

[48] Steffen Rendle. Scaling factorization machines to relational data. *Proceedings of the VLDB Endowment*, 6(5):337–348, 2013.

[49] Report. Relational database management system market: Industry analysis and forecast 2021-2027: By type, deployment, end users, and region, 2022.

[50] Maximilian Schleich, Dan Olteanu, Mahmoud Abo-Khamis, Hung Q Ngo, and XuanLong Nguyen. Learning models over relational data: A brief tutorial. In *Scalable Uncertainty Management: 13th International Conference, SUM 2019, Compiègne, France, December 16–18, 2019, Proceedings 13*, pages 423–432. Springer, 2019.

[51] Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. Learning linear regression models over factorized joins. In *Proceedings of the 2016 International Conference on Management of Data*, pages 3–18, 2016.

[52] Yihan Sun and Guy E Blelloch. Parallel range, segment and rectangle queries with augmented maps. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 159–173. SIAM, 2019.

[53] Nikolaos Tziavelis, Deepak Ajwani, Wolfgang Gatterbauer, Mirek Riedewald, and Xiaofeng Yang. Optimal algorithms for ranked enumeration of answers to full conjunctive queries. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 13, page 1582. NIH Public Access, 2020.

[54] Nikolaos Tziavelis, Nofar Carmeli, Wolfgang Gatterbauer, Benny Kimelfeld, and Mirek Riedewald. Efficient computation of quantiles over joins. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 303–315, 2023.

[55] Keyu Yang, Yunjun Gao, Lei Liang, Bin Yao, Shiting Wen, and Gang Chen. Towards factorized svm with gaussian kernels over normalized data. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1453–1464. IEEE, 2020.

[56] Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, volume 81, pages 82–94, 1981.

[57] Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. Random sampling over joins revisited. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1525–1539, 2018.

A EFFICIENT ALGORITHMS FOR RELATIONAL k-MEANS CLUSTERING

In this section, we focus on the relational k-means clustering. The algorithm is similar with the algorithm described in section 3. However, there are changes in the analysis of the algorithms that need to be addressed. For simplicity, we focus on the (geometric) relational k-means clustering. All the results are straightforwardly extended to the discrete version using $\mathsf{DkMeansAlg}_\gamma$ instead of $\mathsf{GkMeansAlg}_\gamma$ and using the fact that $\mu_{\mathrm{OPT}(Y)}(Y) \leq \mu_{\mathrm{OPT}_{\mathrm{disc}}(Y)}(Y) \leq 4\mu_{\mathrm{OPT}(Y)}(Y)$, for any set of tuples Y, equivalently to the relational k-median clustering problem in Sections 3, 4. Due to space constraints, all missing proofs can be found in the full version of the paper [27].

We keep the main notation as it is in section 3. We change the definition of the set X to be a set of points in $\mathbb{R}^{d_u}$ such that $\mu_X(\boldsymbol{q}_u(\mathbf{D})) \leq \alpha \cdot \mu_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, where $\alpha > 1$ is a constant. Notice that $\mu_X(\boldsymbol{q}_u(\mathbf{D})) = \sum_{t\in\boldsymbol{q}(\mathbf{D})} \phi^2(\pi_{\mathrm{A}_u}(t), X)$. We also assume that r is a real number such that $\mu_X(\boldsymbol{q}_u(\mathbf{D})) \leq r \leq \alpha \cdot \mu_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$. In fact, we can also assume that $\frac{r}{1+\varepsilon} \leq \mu_X(\boldsymbol{q}_u(\mathbf{D}))$. Note that again we do not assume anything about the size of X, and in the next section, we show that we can always consider $|X| = O(k^2)$. Again, we assume that at first, the set X and the number r are given as input, and later we describe the algorithm to efficiently construct X and r. Similarly to the relational k-median clustering, we propose two algorithms, one slower deterministic and one faster randomized. The running times of the algorithms are the same as the running times of the algorithms for the relational k-median clustering.

A.1 Deterministic algorithm

We set $\Phi = \sqrt{\frac{r}{\alpha n}}$ to be a lower bound estimate of the average mean radius $\sqrt{\frac{\mu_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))}{n}}$, and keep all other parameters as described for the k-median coreset in section 3.1. Then we construct the same exponential grids and follow the exact same algorithms to get the coreset C. Then, we run the standard algorithm for the weighted k-means problem on C, $\mathsf{GkMeansAlg}_\gamma(C)$, to get a set of k centers $\mathcal{S}$. We return the set of centers $\mathcal{S}$. Furthermore, we set and return $r_u = \frac{1}{1-\varepsilon}\mu_{\mathcal{S}}(C)$.

Correctness. We conduct a correctness analysis by proving analogous versions of the lemmas established for the k-median clustering, but this time for the k-means clustering.

For any tuple $t \in \boldsymbol{q}_u(\mathbf{D})$, let $x_i \in X$ be the center that is closest to t. We have that $\phi(t, x_i) \leq \alpha n \Phi$. So, the following lemma is correct by the same argument as in lemma 3.1.

Lemma A.1. *Every tuple $t \in \boldsymbol{q}_u(\mathbf{D})$ is assigned to one point in C. Furthermore, the number of tuples in $\boldsymbol{q}_u(\mathbf{D})$ that are assigned to a point $s \in C$ is $w(s)$.*

Any point $t \in \boldsymbol{q}_u(\mathbf{D})$ is assigned to one point in C, so we can define $x_{i(t)}$ and $\sigma(t)$ as before.

Lemma A.2. *C is a k-means ε-coreset for $\boldsymbol{q}_u(\mathbf{D})$.*

Proof. Let Y be an arbitrary set of k points in $\mathbb{R}^{d_u}$. The error is defined as

$$\begin{aligned}\mathcal{E} = |\mu_Y(\boldsymbol{q}(\mathbf{D})) - \mu_Y(C)| &\leq \sum_{t\in\boldsymbol{q}_u(\mathbf{D})} |\phi^2(t, Y) - \phi^2(\sigma(t), Y)| \\ &= \sum_{t\in\boldsymbol{q}_u(\mathbf{D})} |(\phi(t, Y) - \phi(\sigma(t), Y))(\phi(t, Y) + \phi(\sigma(t), Y))|.\end{aligned}$$

By the triangle inequality, $\phi(t, Y) \leq \phi(\sigma(t), Y) + \phi(t, \sigma(t))$ and $\phi(\sigma(t), Y) \leq \phi(t, Y) + \phi(t, \sigma(t))$. Hence, $|\phi(t, Y) - \phi(\sigma(t), Y)| \leq \phi(t, \sigma(t))$. We have,

$$\mathcal{E} \leq \sum_{t\in\boldsymbol{q}_u(\mathbf{D})} |(\phi(t, Y) - \phi(\sigma(t), Y))(\phi(t, Y) + \phi(\sigma(t), Y))| \leq \sum_{t\in\boldsymbol{q}_u(\mathbf{D})} \phi(t, \sigma(t))(2\phi(t, Y) + \phi(t, \sigma(t))).$$

We divide the points of $\boldsymbol{q}_u(\mathbf{D})$ into three cases. Let $P_1 = \{t \in \boldsymbol{q}_u(\mathbf{D}) | \phi(t, x_{i(t)}) \leq \Phi \wedge \phi(t, Y) \leq \Phi\}$, $P_2 = \{t \in \boldsymbol{q}_u(\mathbf{D}) \setminus P_1 | \phi(t, Y) \leq \phi(t, x_{i(t)})\}$, and $P_3 = \{t \in \boldsymbol{q}_u(\mathbf{D}) \setminus P_1 | \phi(t, x_{i(t)}) < \phi(t, Y)\}$. Note that we do not actually construct these sets, instead we only analyze the error induced by these three types of points separately. It is straightforward to see that $P_1 \cup P_2 \cup P_3 = \boldsymbol{q}_u(\mathbf{D})$ and $P_1 \cap P_2 = \emptyset$, $P_1 \cap P_3 = \emptyset$, $P_2 \cap P_3 = \emptyset$.

For a point $t \in P_1$, since $\phi(t, x_{i(t)}) \leq \Phi$, by the construction of the exponential grid, we have that $\phi(t, \sigma(t)) \leq \mathrm{diam}(\square_t) \leq \frac{\varepsilon}{10\alpha}\Phi$, hence,

$$\sum_{t \in P_1} \phi(t, \sigma(t))(2\phi(t, Y) + \phi(t, \sigma(t)) \leq \sum_{t \in P_1} \frac{\varepsilon}{10\alpha}\Phi(2\Phi + \frac{\varepsilon}{10\alpha}\Phi) \leq \frac{3\varepsilon}{10\alpha} n\Phi^2 \leq \frac{3\varepsilon}{10}\mu_{OPT(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

By the definition of the set P_2, we have $\phi(t, x_{i(t)}) > \Phi$, hence, as shown in the proof of lemma 3.2, $\phi(t, \sigma(t)) \leq \frac{\varepsilon}{10\alpha}\phi(t, x_{i(t)})$, and $\phi(t, x_{i(t)}) \leq (1 + \frac{4\varepsilon}{10\alpha})\phi(t, X)$. Therefore,

$$\sum_{t \in P_2} \phi(t, \sigma(t))(2\phi(t, Y) + \phi(t, \sigma(t)) \leq \sum_{t \in P_2} \frac{\varepsilon}{10\alpha}\phi(t, x_{i(t)})(2\phi(t, x_{i(t)}) + \frac{\varepsilon}{10\alpha}\phi(t, x_{i(t)}))$$

$$\leq \frac{3\varepsilon}{10\alpha}\sum_{t \in P_2} \phi^2(t, x_{i(t)}) \leq \frac{3\varepsilon}{10\alpha}\sum_{t \in P_2}(1 + \frac{4\varepsilon}{10\alpha})\phi^2(t, X) \leq \frac{12\varepsilon}{10\alpha}\mu_X(\boldsymbol{q}_u(\mathbf{D})) \leq \frac{12\varepsilon}{10}\mu_{OPT(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

For the last case, where $t \in P_3$, we have $\phi(t, \sigma(t)) \leq \frac{\varepsilon}{10\alpha}\phi(t, Y)$. It holds because: If $\phi(t, x_{i(t)}) \leq \Phi$, then $\phi(t, \sigma(t)) \leq \frac{\varepsilon}{10\alpha}\Phi \leq \frac{\varepsilon}{10\alpha}\phi(t, Y)$. If $\phi(t, x_{i(t)}) > \Phi$ then $\phi(t, \sigma(t)) \leq \frac{\varepsilon}{10\alpha}\phi(t, x_{i(t)}) \leq \frac{\varepsilon}{10\alpha}\phi(t, Y)$. Hence, we have,

$$\sum_{t \in P_3} \phi(t, \sigma(t))(2\phi(t, Y) + \phi(t, \sigma(t)) \leq \sum_{t \in P_3} \frac{\varepsilon}{10\alpha}\phi(t, Y)(2\phi(t, Y) + \frac{\varepsilon}{10\alpha}\phi(t, Y)) \leq \frac{3\varepsilon}{10\alpha}\sum_{t \in P_3}\phi^2(t, Y)$$

$$\leq \frac{3\varepsilon}{10\alpha}\mu_Y(\boldsymbol{q}_u(\mathbf{D})) \leq \frac{3\varepsilon}{10}\mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

Finally, we bound the error,

$$\mathcal{E} \leq \sum_{t \in \boldsymbol{q}_u(\mathbf{D})} \phi(t, \sigma(t))(2\phi(t, Y) + \phi(t, \sigma(t)) \leq \sum_{t \in P_1} \phi(t, \sigma(t))(2\phi(t, Y) + \phi(t, \sigma(t))$$

$$+ \sum_{t \in P_2} \phi(t, \sigma(t))(2\phi(t, Y) + \phi(t, \sigma(t)) + \sum_{t \in P_3} \phi(t, \sigma(t))(2\phi(t, Y) + \phi(t, \sigma(t))$$

$$\leq \frac{3\varepsilon}{10}\mu_{OPT(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})) + \frac{12\varepsilon}{10}\mu_{OPT(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})) + \frac{3\varepsilon}{10}\mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})) \leq \frac{18\varepsilon}{10}\mu_Y(\boldsymbol{q}_u(\mathbf{D})).$$

Thus, if we set $\varepsilon \leftarrow \varepsilon/18$, the result follows. □

With the same argument as in lemma 3.3, we can prove the following lemma.

Lemma A.3. $\mu_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (1 + \varepsilon)\gamma\mu_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$.

As the algorithm for constructing the k-means coreset is almost the same as the one for the k-median's coreset, and the only difference is in the analysis, the running time is the same and we can conclude with the following theorem.

Theorem A.4. *Let* $\mathbf{D}$ *be a database instance with* N *tuples,* $\boldsymbol{q}$ *be an acyclic join query over a set of attributes* $\mathbf{A}$ *and* $\mathbf{A}_u \subseteq \mathbf{A}$. *Given a set* $X \subset \mathbb{R}^d$, *a constant* α *such that* $\mu_X(\boldsymbol{q}_u(\mathbf{D})) \leq \alpha\mu_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, *and a constant parameter* $\varepsilon \in (0, 1)$, *there exists an algorithm that computes a set* $\mathcal{S} \subset \mathbb{R}^d$ *of* k *points and a number* r_u *in* $O(|X|^{d_u+1}N\log^{d_u+2}N + T_Y^{\mathrm{mean}}(|X|\log N))$ *time such that* $\mu_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (1 + \varepsilon)\gamma\mu_{\mathrm{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$. *There also exists an algorithm that computes a set* $\mathcal{S} \subseteq \boldsymbol{q}_u(\mathbf{D})$ *of* k *points and a number* r_u *with the same running time, such that* $\mu_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (4 + \varepsilon)\gamma\mu_{\mathrm{OPT}_{disc}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$.

A.2 Randomized Algorithm

The algorithm for the k-means problem is exactly the same as the one we described in section 3.2 for the k-median problem. Hence, in this section, we use the same notation and prove the additional lemmas required to show that the algorithm works also for k-means clustering. Note that this algorithm builds the k-means version of the exponential grid as described in A.1, and then continues with the randomized counting algorithm described in section 3.2.

Correctness. Notice that Lemmas 3.6 and 3.7 are independent of the objective function of the problem and we can directly assume they are true without proving them again in this section. The proof of the next lemma follows from the same steps as in Lemma 3.8.

Lemma A.5. *Let Y be an arbitrary set of k points in $\mathbb{R}^{d_u}$. It holds that $\mu_Y(\bar{P}_u) \leq \varepsilon\mu_Y(P_u)+\varepsilon\mu_Y(\boldsymbol{q}(\mathbf{D}))$ and $\mu_Y(\boldsymbol{q}_u(\mathbf{D})) \leq (1+\varepsilon)\mu_Y(P_u)$ with probability at least $1-\frac{1}{N^{O(1)}}$.*

Similarly, the proof of the next lemma is analogous to that of Lemma 3.9, and is shown in [27].

Lemma A.6. *C is a k-means ε-coreset of P_u with probability at least $1-\frac{1}{N^{O(1)}}$.*

Finally, we show that $\mu_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D}))$ is a good approximation of $\mu_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D})}(\boldsymbol{q}_u(\mathbf{D}))$ and that $r_u = \frac{1+\varepsilon}{(1-\varepsilon)^2}\mu_{\mathcal{S}}(C)$ is a good estimate of $\mu_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D}))$. The proof of the following lemma is analogous to that of Lemma 3.10.

Lemma A.7. *$\mu_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (1+\varepsilon)\gamma\mu_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, with probability at least $1-\frac{1}{N^{O(1)}}$.*

Notice that there is no difference between the above algorithm and the one for k-median proposed in section 3.2, and the only difference is in the correctness analysis. Therefore, the running is exactly the same, and we can directly have the following theorem.

Theorem A.8. *Let $\mathbf{D}$ be a database instance with N tuples, $\boldsymbol{q}$ be an acyclic join query over a set of attributes $\mathbf{A}$, and $\mathbf{A}_u \subseteq \mathbf{A}$. Given a set $X \subset \mathbb{R}^d$, a constant α such that $\mu_X(\boldsymbol{q}_u(\mathbf{D})) \leq \alpha\mu_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, and a constant parameter $\varepsilon \in (0,1)$, there exists an algorithm that computes a set $\mathcal{S} \subset \mathbb{R}^d$ of k points and a number r_u in $O(|X|N\log N+|X|^2\log^3(N)\min\{\log^{d_u}(|X|), |X|\log N\}+T_\gamma^{\mathsf{mean}}(|X|\log N))$ such that $\mu_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (1+\varepsilon)\gamma\mu_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, with probability at least $1-\frac{1}{N^{O(1)}}$. There also exists an algorithm that computes a set $\mathcal{S} \subseteq \boldsymbol{q}_u(\mathbf{D})$ of k points and a number r_u with the same running time, such that $\mu_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (4+\varepsilon)\gamma\mu_{\mathsf{OPT}_{disc}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, with probability at least $1-\frac{1}{N^{O(1)}}$.*

A.3 Efficient Algorithms

We use the results from the previous section to describe a complete algorithm for the relational k-means clustering. As we did in section 4.1, in this section, we describe how to efficiently construct the set X and the number r. To do so, we construct a binary tree as described in section 4.1, and use the same notation. Our goal this time is to compute for each node u, a set $\mathcal{S}_u$, and a number r_u, such that,

$$\mu_{\mathcal{S}_u}(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq (1+\varepsilon)\gamma\mu_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})). \tag{7}$$

The algorithm is identical to that of k-median clustering, with one difference: instead of applying the standard k-median algorithm to the leaf nodes, we use the standard k-means algorithm, and for the intermediate nodes, we apply the coreset-based relational k-means algorithms, as described above.

Correctness. For any set $Y \subseteq \mathbb{R}^{d_u}$, $\mu_Y(\boldsymbol{q}_u(\mathbf{D})) = \sum_{t\in\boldsymbol{q}(\mathbf{D})}\phi^2(\pi_{\mathbf{A}_u}(t), Y) = \sum_{t\in\boldsymbol{q}(\mathbf{D})}\sum_{A_j\in\mathbf{A}_u}(\pi_{A_j}(t)-\pi_{A_j}(\mathcal{N}(Y,\pi_{\mathbf{A}_u}(t)))^2$. Assuming that $\mathcal{S}_v, r_v$ and $\mathcal{S}_z, r_z$ satisfy Equation 7, we show how to compute $\mathcal{S}_u$ and r_u that satisfy Equation 7.

If we prove that X is a constant α-approximation of the k-means problem on $\boldsymbol{q}_u(\mathbf{D})$ and $\mu_X(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq \alpha\mu_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$, then the correctness follows from Theorem A.8 (or Theorem A.4). The proof of the next lemma is a simpler version of the proof in Lemma 4.1, and it follows by the same arguments. We show its proof in [27].

Lemma A.9. $\mu_X(\boldsymbol{q}_u(\mathbf{D})) \leq r_u \leq \alpha\mu_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$.

We have that $\mathcal{S} = \mathcal{S}_p$ is a $(1+\varepsilon)\gamma$ approximation of the relational k-median problem in $\boldsymbol{q}(\mathbf{D})$.

The algorithm is the same as the proposed algorithm for the k-median problem, and it has the same time complexity. Therefore we can conclude with the following theorem.

Theorem A.10. *Given an acyclic join query $\boldsymbol{q}$, a database instance $\mathbf{D}$, a parameter k, and a constant parameter $\varepsilon \in (0,1)$, there exists an algorithm that computes a set $\mathcal{S} \subset \mathbb{R}^d$ of k points such that $\mu_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D})) \leq (1+\varepsilon)\gamma\mu_{\mathsf{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$, with probability at least $1-\frac{1}{N^{O(1)}}$. The running time of the algorithm is $O(Nk^2\log(N) + k^4\log^3(N)\min\{\log^d(k), k^2\log N\} + T_\gamma^{\mathsf{mean}}(k^2\log N))$. Furthermore, there exists an algorithm that computes a set $\mathcal{S}' \subseteq \boldsymbol{q}(\mathbf{D})$ of k points in the same time such that $\mu_{\mathcal{S}'}(\boldsymbol{q}(\mathbf{D})) \leq (4+\varepsilon)\gamma\mu_{\mathsf{OPT}_{disc}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$, with probability at least $1-\frac{1}{N^{O(1)}}$.*

B MISSING PROOFS FROM SUBSECTION 3.1

Proof of Lemma 3.3. Next, we assume that $\mathsf{DkMedianAlg}_\gamma$ is used. We note that

$$\begin{aligned}\mathbf{v}_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) &\leq \frac{1}{1-\varepsilon'}\mathbf{v}_{\mathcal{S}}(C) = r_u \leq \gamma\frac{1}{1-\varepsilon'}\mathbf{v}_{\mathsf{OPT}_{\mathsf{disc}}(C)}(C) \leq 2\gamma\frac{1}{1-\varepsilon'}\mathbf{v}_{\mathsf{OPT}(C)}(C)\\ &\leq 2\frac{1+\varepsilon'}{1-\varepsilon'}\gamma\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})) \leq 2(1+4\varepsilon')\gamma\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))\\ &= 2(1+\varepsilon)\gamma\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})) \leq 2(1+\varepsilon)\gamma\mathbf{v}_{\mathsf{OPT}_{\mathsf{disc}}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).\end{aligned}$$

The result follows setting $\varepsilon \leftarrow \varepsilon/2$. □

C MISSING PROOFS FROM SUBSECTION 3.2

Proof of Lemma 3.8. We bound the second term in the sum. We define a new assignment function σ' as we did in Lemma 3.2. For a point $p \in \bar{P}_u$ we set $\sigma'(p) = p$. If a point $t \in P_u$, is not charged by any point in $\bigcup_{\square\in L}\mathcal{P}_\square^L$, then $\sigma'(t) = t$. For a point $t \in P_u$ let p_t be the point in $\bigcup_{\square\in L}\mathcal{P}_\square^L$ such that p_t charges $t = t_{j_h(p_t)}$ for a value of h, according to Lemma 3.7. If $p_t \in \bar{P}_u$ then $\sigma'(t) = p_t$. If $p_t \in P_u$, $\sigma'(t) = t$. Using the new assignment function $\sigma'(\cdot)$, the second term can be bounded as

$$\varepsilon'\sum_{p\in\bar{P}_u}\sum_{h=1}^{1/\varepsilon'}\phi(p, t_{j_h(p)}) \leq \varepsilon'\sum_{t\in P_u}\phi(t,\sigma'(t)) \leq \varepsilon'\sum_{t\in\boldsymbol{q}_u(\mathbf{D})}\phi(t,\sigma'(t)).$$

We note that σ' is a different assignment than the assignment σ we used in Lemma 3.2, however, notice that in all cases both t and $\sigma'(t)$ belong to the same cell defined by the exponential grids as constructed and processed by the algorithm. In fact, both t and $\sigma'(t)$ belong in the same cell $\square_{i'(t)} \in \overline{V}_{i'(t)}$ around a center $x_{i'(t)} \in X$. By construction, condition (3) is satisfied for $x_{i'(t)}$ and $\square_{i'(t)}$. Hence, the same properties hold. We can distinguish between $\phi(t, x_{i'(t)}) \leq \Phi$ and $\phi(t, x_{i'(t)}) > \Phi$ as we did in Lemma 3.2 making the same arguments. Using the proof of Lemma 3.2, we get that $\varepsilon'\sum_{t\in\boldsymbol{q}_u(\mathbf{D})}\phi(t,\sigma'(t)) \leq (\varepsilon')^2\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D})) \leq \varepsilon'\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D}))$. The first inequality $\mathbf{v}_Y(\bar{P}_u) \leq \varepsilon'\mathbf{v}_Y(P_u) + \varepsilon'\mathbf{v}_Y(\boldsymbol{q}_u(\mathbf{D}))$ follows. □

Proof of Lemma 3.10. We assume that $\mathsf{DkMedianAlg}_\gamma$ is used. We have,

$$\begin{aligned}\mathbf{v}_{\mathcal{S}}(C) &\leq \gamma\mathbf{v}_{\mathsf{OPT}_{\mathsf{disc}}(C)}(C) \leq 2\gamma\mathbf{v}_{\mathsf{OPT}(C)}(C) \leq 2\gamma\mathbf{v}_{\mathsf{OPT}(P_u)}(C) \leq 2(1+9\varepsilon')\gamma\mathbf{v}_{\mathsf{OPT}(P_u)}(P_u)\\ &\leq 2(1+9\varepsilon')\gamma\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})),\ \text{hence,}\end{aligned}$$

$$\mathbf{v}_{\mathcal{S}}(\boldsymbol{q}_u(\mathbf{D})) \leq (1+4\varepsilon')\mathbf{v}_{\mathcal{S}}(P_u) \leq \frac{1+4\varepsilon'}{1-9\varepsilon'}\mathbf{v}_{\mathcal{S}}(C) = r_u \leq 2\frac{(1+4\varepsilon')(1+9\varepsilon')}{1-9\varepsilon'}\gamma\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$$
$$\leq 2(1+34\varepsilon')\gamma\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})) \leq 2(1+\varepsilon)\gamma\mathbf{v}_{\mathsf{OPT}_{\mathrm{disc}}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

The result follows setting $\varepsilon \leftarrow \varepsilon/2$. □

D MISSING PROOFS FROM SUBSECTION 4.1

Proof of Lemma 4.1. We first consider that $\mathsf{GkMedianAlg}_\gamma$ is used. Let $O_v = \pi_{\mathbf{A}_v}(\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D})))$, and $O_z = \pi_{\mathbf{A}_z}(\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D})))$. We define $O = O_v \times O_z$. Notice that $\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D})) \subseteq O$ so $\mathbf{v}_O(\boldsymbol{q}_u(\mathbf{D})) \leq \mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D}))$. We have,

$$\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D})) = \sum_{t\in\boldsymbol{q}(\mathbf{D})} ||\pi_{\mathbf{A}_u}(t) - \mathcal{N}(X, \pi_{\mathbf{A}_u}(t))||$$
$$= \sum_{t\in\boldsymbol{q}(\mathbf{D})} \sqrt{||\pi_{\mathbf{A}_v}(t) - \pi_{\mathbf{A}_v}(\mathcal{N}(X,\pi_{\mathbf{A}_u}(t)))||^2 + ||\pi_{\mathbf{A}_z}(t) - \pi_{\mathbf{A}_z}(\mathcal{N}(X,\pi_{\mathbf{A}_u}(t)))||^2}$$
$$\leq \sum_{t\in\boldsymbol{q}(\mathbf{D})} ||\pi_{\mathbf{A}_v}(t) - \pi_{\mathbf{A}_v}(\mathcal{N}(X,\pi_{\mathbf{A}_u}(t)))|| + \sum_{t\in\boldsymbol{q}(\mathbf{D})} ||\pi_{\mathbf{A}_z}(t) - \pi_{\mathbf{A}_z}(\mathcal{N}(X,\pi_{\mathbf{A}_u}(t)))||$$
$$= \sum_{t\in\boldsymbol{q}(\mathbf{D})} ||\pi_{\mathbf{A}_v}(t) - \mathcal{N}(\mathcal{S}_v,\pi_{\mathbf{A}_v}(t))|| + \sum_{t\in\boldsymbol{q}(\mathbf{D})} ||\pi_{\mathbf{A}_z}(t) - \mathcal{N}(\mathcal{S}_z,\pi_{\mathbf{A}_z}(t))|| \leq r_v + r_z$$
$$\leq (1+\varepsilon)\gamma\left(\sum_{t\in\boldsymbol{q}(\mathbf{D})}||\pi_{\mathbf{A}_v}(t)-\mathcal{N}(\mathsf{OPT}(\boldsymbol{q}_v(\mathbf{D})),\pi_{\mathbf{A}_v}(t))||+\sum_{t\in\boldsymbol{q}(\mathbf{D})}||\pi_{\mathbf{A}_z}(t) - \mathcal{N}(\mathsf{OPT}(\boldsymbol{q}_z(\mathbf{D})),\pi_{\mathbf{A}_z}(t))||\right)$$
$$\leq (1+\varepsilon)\gamma\sum_{t\in\boldsymbol{q}(\mathbf{D})} ||\pi_{\mathbf{A}_v}(t) - \mathcal{N}(O_v,\pi_{\mathbf{A}_v}(t))|| + (1+\varepsilon)\gamma\sum_{t\in\boldsymbol{q}(\mathbf{D})} ||\pi_{\mathbf{A}_z}(t) - \mathcal{N}(O_z,\pi_{\mathbf{A}_z}(t))||$$
$$= (1+\varepsilon)\gamma\sum_{t\in\boldsymbol{q}(\mathbf{D})} \left(||\pi_{\mathbf{A}_v}(t) - \pi_{\mathbf{A}_v}(\mathcal{N}(O,\pi_{\mathbf{A}_u}(t)))|| + ||\pi_{\mathbf{A}_z}(t) - \pi_{\mathbf{A}_z}(\mathcal{N}(O,\pi_{\mathbf{A}_u}(t)))||\right)$$
$$\leq (1+\varepsilon)\gamma\sqrt{2}\sum_{t\in\boldsymbol{q}(\mathbf{D})} \sqrt{||\pi_{\mathbf{A}_v}(t) - \pi_{\mathbf{A}_v}(\mathcal{N}(O,\pi_{\mathbf{A}_u}(t)))||^2 + ||\pi_{\mathbf{A}_z}(t) - \pi_{\mathbf{A}_z}(\mathcal{N}(O,\pi_{\mathbf{A}_u}(t)))||^2}$$
$$= (1+\varepsilon)\gamma\sqrt{2}\sum_{t\in\boldsymbol{q}(\mathbf{D})} ||\pi_{\mathbf{A}_u}(t) - \mathcal{N}(O,\pi_{\mathbf{A}_u}(t))|| = (1+\varepsilon)\gamma\sqrt{2}\cdot\mathbf{v}_O(\boldsymbol{q}_u(\mathbf{D}))$$
$$\leq (1+\varepsilon)\gamma\sqrt{2}\cdot\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

The first and second equalities hold by the definition of the Euclidean metric. The third inequality holds because $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, for $a, b > 0$.

We show the fourth inequality by proof by contradiction. Notice that for any $\bar{x} \in X$, $\pi_{\mathbf{A}_v}(\bar{x}) \in \mathcal{S}_v$, because $X = \mathcal{S}_v \times \mathcal{S}_z$. Let $s \in \mathcal{S}_v$ be the closest point in $\mathcal{S}_v$ from $\pi_{\mathbf{A}_v}(t)$, and let $x \in X$ be the closest point in X from $\pi_{\mathbf{A}_u}(t)$. Without loss of generality, assume that s is the unique nearest neighbor. Assume that $\pi_{\mathbf{A}_v}(\mathcal{N}(X,\pi_{\mathbf{A}_u}(t))) \neq \mathcal{N}(\mathcal{S}_v,\pi_{\mathbf{A}_v}(t)) \Leftrightarrow \pi_{\mathbf{A}_v}(x) \neq s$. In fact, assume that $\pi_{\mathbf{A}_v}(x) = s' \in \mathcal{S}_v$ such that $s' \neq s$, and $\pi_{\mathbf{A}_z}(x) = \bar{s} \in \mathcal{S}_z$. Let $x' = s \times \bar{s} \in X$. We have,

$$||x - \pi_{\mathbf{A}_u}(t)|| = \sqrt{\sum_{A_j\in\mathbf{A}_u}(\pi_{A_j}(x) - \pi_{A_j}(t))^2} = \sqrt{\sum_{A_j\in\mathbf{A}_v}(\pi_{A_j}(x) - \pi_{A_j}(t))^2 + \sum_{A_j\in\mathbf{A}_z}(\pi_{A_j}(x) - \pi_{A_j}(t))^2}$$
$$> \sqrt{\sum_{A_j\in\mathbf{A}_v}(\pi_{A_j}(s) - \pi_{A_j}(t))^2 + \sum_{A_j\in\mathbf{A}_z}(\pi_{A_j}(\bar{s}) - \pi_{A_j}(t))^2} = \sqrt{\sum_{A_j\in\mathbf{A}_u}(\pi_{A_j}(x') - \pi_{A_j}(t))^2}$$
$$= ||x' - \pi_{\mathbf{A}_u}(t)||,$$

which is a contradiction because x is the closest point in X from $\pi_{\mathbf{A}_u}(t)$.

The fifth inequality holds because r_v, r_z satisfy Equation (5). The sixth inequality holds because of the definition of $\mathcal{S}_v$ and $\mathcal{S}_z$, i.e., $\mathbf{v}_{\mathcal{S}_v}(\boldsymbol{q}_v(\mathbf{D})) \leq (1+\varepsilon)\gamma\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_v(\mathbf{D}))}(\boldsymbol{q}_v(\mathbf{D}))$ (similarly for z). The seventh inequality holds because $\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_v(\mathbf{D}))}(\boldsymbol{q}_v(\mathbf{D})) \leq \mathbf{v}_{O_v}(\boldsymbol{q}_v(\mathbf{D}))$ (similarly for z). The eighth equality holds because it is equivalent to the fourth inequality. The ninth inequality holds because $a+b \leq \sqrt{2}\sqrt{a^2+b^2}$ for $a, b > 0$. The tenth equality holds because of the definition of the Euclidean metric. The eleventh equality holds by definition of $\mathbf{v}_O(\boldsymbol{q}_u(\mathbf{D}))$. The last inequality holds because $\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D})) \subseteq O$. Overall, $\alpha = (1+\varepsilon)\gamma\sqrt{2}$ which is a constant. Furthermore, notice that $r = r_v + r_z$ so the result follows. Moreover, $r_v \leq (1+\varepsilon)\sum_{t\in\boldsymbol{q}(\mathbf{D})} ||\pi_{\mathbf{A}_v}(t) - \mathcal{N}(\mathcal{S}_v, \pi_{\mathbf{A}_v}(t))||$ so $r \leq (1+\varepsilon)\sqrt{2}\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D}))$.

Similarly, we show the analysis assuming $D\mathsf{GkMedianAlg}_\gamma$ is used. We have,
$\mathbf{v}_X(\boldsymbol{q}_u(\mathbf{D})) \leq \ldots \leq r_v + r_z$

$$\leq (2+\varepsilon)\gamma\left(\sum_{t\in\boldsymbol{q}(\mathbf{D})}\left(||\pi_{\mathbf{A}_v}(t) - \mathcal{N}(\mathsf{OPT}_{\mathsf{disc}}(\boldsymbol{q}_v(\mathbf{D})), \pi_{\mathbf{A}_v}(t))|| + ||\pi_{\mathbf{A}_z}(t) - \mathcal{N}(\mathsf{OPT}_{\mathsf{disc}}(\boldsymbol{q}_z(\mathbf{D})), \pi_{\mathbf{A}_z}(t))||\right)\right)$$

$$\leq 2(2+\varepsilon)\gamma\left(\sum_{t\in\boldsymbol{q}(\mathbf{D})}\left(||\pi_{\mathbf{A}_v}(t) - \mathcal{N}(\mathsf{OPT}(\boldsymbol{q}_v(\mathbf{D})), \pi_{\mathbf{A}_v}(t))|| + ||\pi_{\mathbf{A}_z}(t) - \mathcal{N}(\mathsf{OPT}(\boldsymbol{q}_z(\mathbf{D})), \pi_{\mathbf{A}_z}(t))||\right)\right)$$

$$\leq \ldots \leq 2(2+\varepsilon)\gamma\sqrt{2}\mathbf{v}_{\mathsf{OPT}(\boldsymbol{q}_u(\mathbf{D}))}(\boldsymbol{q}_u(\mathbf{D})).$$

□

E EXTENSION TO CYCLIC QUERIES

A fractional edge cover of join query $\boldsymbol{q}$ is a point $x = \{x_R \mid R \in \mathbf{R}\} \in \mathbb{R}^m$ such that for any attribute $A \in \mathbf{A}$, $\sum_{R\in\mathbf{R}_A} x_R \geq 1$, where $\mathbf{R}_A$ are all the relations in $\mathbf{A}$ that contain the attribute A. As proved in [13], the maximum output size of a join query $\boldsymbol{q}$ is $O(N^{\|x\|_1})$. Since the above bound holds for any fractional edge cover, we define $\rho = \rho(\boldsymbol{q})$ to be the fractional cover with the smallest ℓ_1-norm, i.e., $\rho(\boldsymbol{q})$ is the value of the objective function of the optimal solution of linear programming (LP): $\min \sum_{R\in\mathbf{R}} x_R$, s.t. $\forall R \in \mathbf{R} : x_R \geq 0$ and $\forall A \in \mathbf{A} : \sum_{R\in\mathbf{A}_R} x_R \geq 1$.

Next, we give the definition of the Generalized Hypertree Decomposition (GHD). A GHD of $\boldsymbol{q}$ is a pair $(\mathcal{T}, \lambda)$, where $\mathcal{T}$ is a tree as an ordered set of nodes and $\lambda : \mathcal{T} \longrightarrow 2^{\mathbf{A}}$ is a labeling function which associates to each vertex $u \in \mathcal{T}$ a subset of attributes in $\mathbf{A}$, called λ_u, such that the following conditions are satisfied: i) (coverage) For each $R \in \mathbf{R}$, there is a node $u \in \mathcal{T}$ such that $\mathbf{A}_R \subseteq \lambda_u$, where $\mathbf{A}_R$ is the set of attributes contained in R; ii) (connectivity) For each $A \in \mathbf{A}$, the set of nodes $\{u \in \mathcal{T} : A \in \lambda_u\}$ forms a connected subtree of $\mathcal{T}$.

Given a join query $\boldsymbol{q}$, one of its GHD $(\mathcal{T}, \lambda)$ and a node $u \in \mathcal{T}$, the width of u is defined as the optimal fractional edge covering number of its derived hypergraph $(\lambda_u, \mathcal{E}_u)$, where $\mathcal{E}_u = \{\mathbf{A}_R \cap \lambda_u : R \in \mathbf{R}\}$. Given a join query and a GHD $(\mathcal{T}, \lambda)$, the width of $(\mathcal{T}, \lambda)$ is defined as the maximum width over all nodes in $\mathcal{T}$. Then, the fractional hypertree width of a join query follows: The fractional hypertree width of a join query $\boldsymbol{q}$, denoted as $\mathsf{fhw}(\boldsymbol{q})$, is $\mathsf{fhw}(\boldsymbol{q}) = \min_{(\mathcal{T},\lambda)} \max_{u\in\mathcal{T}} \rho(\lambda_u, \mathcal{E}_u)$, i.e., the minimum width over all GHDs.

Overall, $O(N^{\mathsf{fhw}})$ is an upper bound on the number of join results materialized for each node in $\mathcal{T}$. It is also the time complexity to compute the join results for each node in $\mathcal{T}$ [13]. Hence, we converted our original cyclic query into an acyclic join query (with join tree $\mathcal{T}$) where each relation has $O(N^{\mathsf{fhw}})$ tuples. We execute all our algorithms to the new acyclic join query replacing the N factor with the N^{fhw} factor in the running time.

Received May 2024; accepted August 2024