CORE-Sketch: On Exact Computation of Median Absolute Deviation with Limited Space

Haoquan Guan
BNRist, Tsinghua University
ghq22@mails.tsinghua.edu.cn

Ziling Chen
BNRist, Tsinghua University
chen-zl22@mails.tsinghua.edu.cn

Shaoxu Song
BNRist, Tsinghua University
sxsong@tsinghua.edu.cn

ABSTRACT

Median absolute deviation (MAD), the median of the absolute deviations from the median, has been found useful in various applications such as outlier detection. Together with median, MAD is more robust to abnormal data than mean and standard deviation (SD). Unfortunately, existing methods return only approximate MAD that may be far from the exact one, and thus mislead the downstream applications. Computing exact MAD is costly, however, especially in space, by storing the entire dataset in memory. In this paper, we propose COnstruction-REfinement Sketch (CORE-Sketch) for computing exact MAD. The idea is to construct some sketch within limited space, and gradually refine the sketch to find the MAD element, i.e., the element with distance to the median exactly equal to MAD. Mergeability and convergence of the method is analyzed, ensuring the correctness of the proposal and enabling parallel computation. Extensive experiments demonstrate that CORE-Sketch achieves significantly less space occupation compared to the aforesaid baseline of No-Sketch, and has time and space costs relatively comparable to the DD-Sketch method for approximate MAD.

PVLDB Reference Format:
Haoquan Guan, Ziling Chen, and Shaoxu Song. CORE-Sketch: On Exact Computation of Median Absolute Deviation with Limited Space. PVLDB, 16(11): 2832 - 2844, 2023.
doi:10.14778/3611479.3611491

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/thssdb/core-sketch.

1 INTRODUCTION

Median absolute deviation (MAD), derived from the median of a dataset, is a statistic that can well describe the degree of data volatility [7]. Like the relationship between standard deviation (SD) and mean, MAD is the median of the absolute difference between the elements of the dataset and the median of the dataset, describing the fluctuation degree of the elements in the dataset around the median [19]. MAD has been widely used in various applications such as outlier detection [11, 12, 15], rare query event detection [2], error detection [13], database intrusion detection [24], hardware assertion evaluation [3] and so on, owing to its robustness to abnormal data.

Proceedings of the VLDB Endowment, Vol. 16, No. 11 ISSN 2150-8097.
doi:10.14778/3611479.3611491

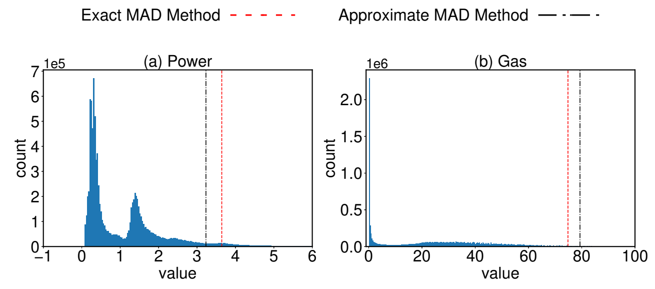


Figure 1: Outlier detection with exact and approximate MAD

1.1 Motivation

While computing exact MAD is costly, efficient estimation of approximate MAD [4] has been studied using DD-Sketch [16] for estimating median. Unfortunately, approximate MAD could be very far from the exact one, and thus mislead downstream applications.

Example 1.1. Figure 1 presents the outlier detection application of MAD over data distribution. Similar to the traditional $\mu \pm k\sigma$ method where μ is mean, σ is SD, and k is a coefficient like 3 [1, 23], data elements outside the bound of median $\pm k$MAD are regarded as outliers [4]. It is known that the mean and SD are not robust enough compared to the median and MAD [18]. We show how approximate MAD affects the performance of outlier detection compared to the exact one. The error bounds ϵ used in Figures 1 (a) and (b) are both 1e-2 under bucket limit M=500 for DD-Sketch [16].

For dataset Power in Figure 1 (a), the approximate MAD is smaller than the exact MAD. As a result, 511,717 points are regarded as outliers using approximate MAD, while only 355,461 points are indeed true outliers calculated by the exact MAD. Under this situation, the precision of outlier detection is merely 56%.

For dataset Gas in Figure 1 (b), the approximate MAD is larger than the exact MAD. Hence, approximate MAD detects only 72,873 points, among 135,241 true outliers calculated by the exact MAD. In this case, the recall of outlier detection is barely 54%.

1.2 Challenges

To compute exact MAD, a straightforward method is to load the entire dataset $\mathcal{D}$ in memory and apply the algorithm such as Quickselect [8] to calculate the median λ of $\mathcal{D}$. Then, the exact MAD is computed by calling the algorithm again to find the median of absolute deviations to the median λ. The No-Sketch method assumes that it is possible to fit the entire dataset $\mathcal{D}$ in memory, as illustrated in Figure 2 (a). Though its runtime may be faster as shown in Figures 11 and 13 of experiments in Section 5, the memory requirements of loading the entire dataset are impractical for real-world

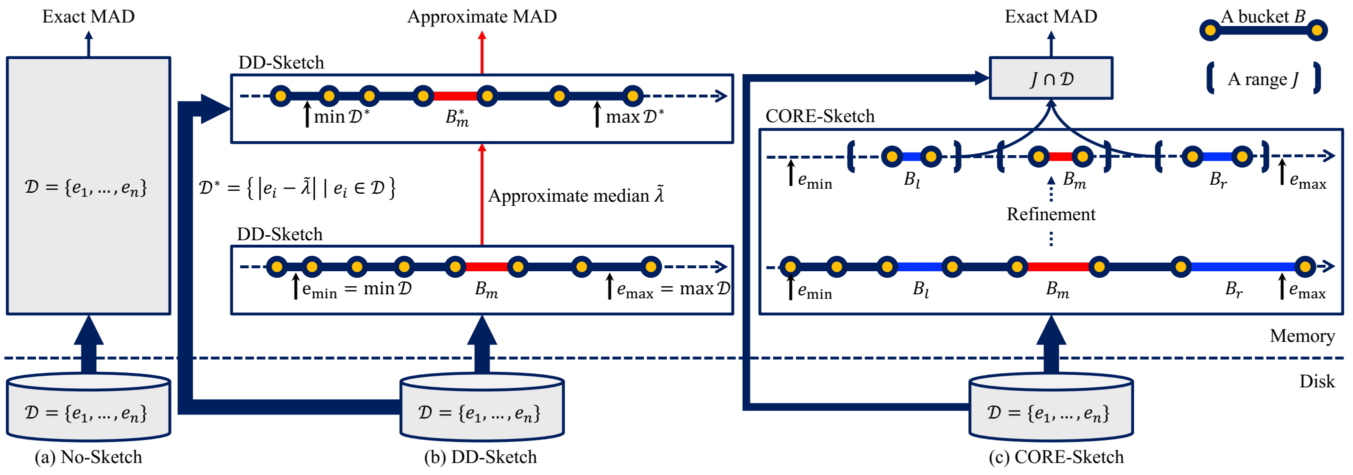

Exact MAD
$\mathcal{D} = \{e_1, \ldots, e_n\}$
$\mathcal{D} = \{e_1, \ldots, e_n\}$
(a) No-Sketch
Approximate MAD
DD-Sketch
min $\mathcal{D}^*$
B_m^*
max $\mathcal{D}^*$
$\mathcal{D}^* = \{ |e_i - \tilde{\lambda}| \mid e_i \in \mathcal{D} \}$
Approximate median $\tilde{\lambda}$
DD-Sketch
$e_{\min} = \min \mathcal{D}$
B_m
$e_{\max} = \max \mathcal{D}$
$\mathcal{D} = \{e_1, \ldots, e_n\}$
(b) DD-Sketch
Exact MAD
$J \cap \mathcal{D}$
A bucket B
A range J
CORE-Sketch
$e_{\min}$
B_l
B_m
B_r
$e_{\max}$
Refinement
$e_{\min}$
B_l
B_m
B_r
$e_{\max}$
Memory
Disk
$\mathcal{D} = \{e_1, \ldots, e_n\}$
(c) CORE-Sketch

Figure 2: An overview of different MAD query approaches

use. Instead, the computation is often with limited space in practice. For example, MCUs or FPGAs in end-devices for IoT applications have only hundreds of KBs or a few MBs memory [14, 17]. The default memory budget for a function in Apache IoTDB is 30 MB and can be up to 1.5% of physical memory [9]. The space is even more limited when multiple queries are processed in parallel (see experiments in Figure 20 in Section 5.3). The memory requirements of No-Sketch algorithm make it impractical for real-world use.

Rather than raw data elements, the approximate MAD approach [4] stores only statistical information in memory, by DD-Sketch presented in Figure 2 (b). Each bucket B counts the number of data elements in $\mathcal{D}$ that are in the range of bucket boundaries. It identifies the median bucket B_m, whose boundaries serve as the lower and upper bounds of the exact median λ, and thus can estimate approximate median $\tilde{\lambda}$ but cannot obtain the exact one. Similar to the aforesaid exact computation, in the second round, another DD-Sketch for $\mathcal{D}^*$, the absolute deviations to $\tilde{\lambda}$, is constructed. Again, by identifying the median bucket B_m^*, the approximate median of $\mathcal{D}^*$ is estimated, i.e., approximate MAD. Since the sketch stores only statistical information, it cannot obtain the exact MAD.

1.3 Intuition

In this paper, we propose a novel sketch structure, COnstruction-REfinement Sketch (CORE-Sketch), for computing exact MAD. In addition to the median bucket B_m, it maintains two more buckets B_l and B_r, whose boundaries together with B_m serve as the lower and upper bounds of MAD.

While MAD is a statistic of the dataset $\mathcal{D}$, we denote the data element $\eta \in \mathcal{D}$ the *MAD element*, which has $|\eta - \lambda| = \text{MAD}$, as in Definition 1. To compute the MAD statistic, it is thus to find the median λ and the MAD element η in $\mathcal{D}$. Rather than computing medians twice, we directly identify the ranges J of data where the median λ and the MAD element η must belong. As illustrated in Figure 2 (c), the exact MAD is thus computed by loading only the corresponding data elements in the ranges J, i.e., a subset $\mathcal{D}'$ of $\mathcal{D}$.

Note that the ranges J are much narrower than that of the entire dataset $\mathcal{D}$, $[e_{\min} = \min \mathcal{D}, e_{\max} = \max \mathcal{D}]$, while still covering the median bucket B_m and the MAD buckets B_l and B_r. Intuitively, we propose to refine the buckets for the data in the ranges J. The refined sketch with smaller median and MAD buckets, B_m, B_l and B_r, can further narrow down the ranges J of data elements to load.

1.4 Contributions

Our major contributions in this paper are summarized as follows.

(1) We devise a sketch structure CORE-Sketch in Section 2. The bucket boundary in the sketch is specially designed for refinement.
(2) We present the algorithm of computing exact MAD using CORE-Sketch in Section 3. First, we investigate the bounds of median and MAD referring to the median bucket B_m and the MAD buckets B_l and B_r, in Propositions 1 and 2. It leads to the ranges J which bound the median λ and the MAD element η, in Propositions 3 and 4. Once the sketch is refined for J, we further prove the bounds of median and MAD over the narrowed buckets, in Propositions 5 and 6. Likewise, the ranges J can be further narrowed down, referring to the ranges determined by the median bucket B_m and the MAD buckets B_l and B_r, as well as the ranges in the previous iteration, in Proposition 7. Finally, the exact MAD is computed by loading the data elements in the ranges of J, referring to Proposition 8.
(3) We prove mergeability of CORE-Sketch, in Proposition 9, which enables parallel computation of exact MAD. Convergence of the iterative algorithm is also analyzed, in Proposition 11.
(4) We conduct extensive experiments over real and synthetic data. Our CORE-Sketch approach shows up to 6 order-of-magnitude improvement in space cost compared to the No-Sketch baseline. The time and space costs of CORE-Sketch are relatively comparable to the method returning only approximate MAD, such as DD-Sketch.

2 PRELIMINARY

In this section, we first formalize the problem of exact MAD query with limited sketch space in Section 2.1. The design of CORE-Sketch is then presented in Section 2.2.

2.1 Problem Statement

To begin with, we first introduce the definition of the median absolute deviation, and formalize the exact MAD query problem.

Definition 1 (Median Absolute Deviation). *For a dataset $\mathcal{D} = \{e_1, e_2, \cdots, e_N\}$, which can be a multiset, its median absolute deviation (MAD) is defined as the median of the absolute differences between the elements and the median of $\mathcal{D}$*

$$\mathrm{MAD}(\mathcal{D}) = \mathrm{MEDIAN}(\{|e - \mathrm{MEDIAN}(\mathcal{D})| \mid e \in \mathcal{D}\}) = |\lambda - \eta|,$$

where $\lambda \in \mathcal{D}$ is the median of $\mathcal{D}$, and $\eta \in \mathcal{D}$ is the MAD element.

To compute exact MAD, it is to find median λ and MAD element η. In practice, a sketch with a number of buckets on statistics is often employed to determine the bounds of λ and η, and thus estimate approximate MAD [4]. In this study, we propose to design a sketch for finding the exact MAD element with limited buckets.

Problem 1 (Exact MAD Computation with Limited Space). *With a limited number of buckets in the sketch, the problem is to compute the exact MAD of a given dataset $\mathcal{D}$.*

To simplify the calculation, for an input dataset $\mathcal{D}$ with minimum value $e_{\min}$, all the elements are subtracted by $e_{\min} - 1$ to make the minimum value 1. Thus, without loss of generality, we suppose $e_{\min} = 1$ in the rest of our paper.

2.2 Sketch Design

To find the MAD element, a sketch is expected to be refined, till the data elements in certain buckets can be stored in memory for computation under certain space budget. Moreover, to ease parallel computation, mergeability is also desired. Both requirements motivate us to design a sketch structure with bucket boundaries from a fixed domain.

Definition 2 (Bucket Boundary Domain). *The bucket boundary domain, denoted as $\mathcal{Z}$, is defined as $\mathcal{Z} = \left\{\zeta \mid \zeta = 2^{2^{-x}},\ x \in \mathbb{N}\right\}$.*

For example, for $x = 0$, we have $\zeta = 2^{2^{-x}} = 2 \in \mathcal{Z}$. If $x = 1$, then we have $\zeta = 2^{2^{-x}} = \sqrt{2} \in \mathcal{Z}$.

Sketches with uniform bin sizes, i.e., equal-width buckets, perform well on datasets with uniform distribution. The buckets keep information equally for all parts of data. For skewed input distributions, buckets of exponential sizes are employed [16] with more buckets concentrated around $e_{\min}$ than $e_{\max}$. It naturally fits the data with more values around $e_{\min}$. For the skewed distribution with more values around $e_{\max}$, a simple idea is to reverse the order [6], e.g., by $e' = e_{\max} - e + 1$ for each $e \in \mathcal{D}$. However, if the distribution is extremely skewed, too many data are concentrated around $e_{\min}$ and thus may need more iterations to refine as shown in Figures 15 (a) and (c). For the data concentrated in the middle, it is not the best case for buckets of exponential sizes (as well as equal-width buckets).

The equal-depth buckets give the most even partition of the dataset, which maximizes the utility of the space. However, the boundaries of the equal-depth buckets can be any values without a fixed domain as in Definition 2. It brings difficulty in refinement, as the bucket boundaries could be completely different before and after refinement, and needs to re-compute entirely. Moreover, the buckets cannot be merged for parallel computing, again owing to the arbitrary bucket boundaries. In contrast, our sketch with bucket boundaries always from the fixed domain $\mathcal{Z}$ in Definition 2 is convenient for refinement and mergeability.

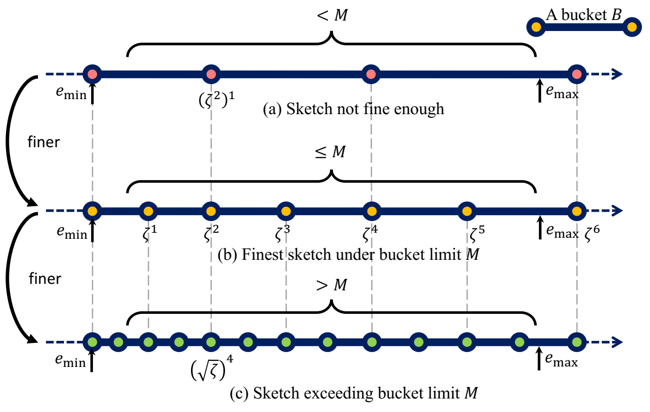


Figure 3: Determine the finest base ζ with bucket limit M

Definition 3 (CORE-Sketch). *A CORE-Sketch $\mathbb{S}(\zeta)$ with base $\zeta \in \mathcal{Z}$ is defined as $\mathbb{S}(\zeta) = \{|B_i| \mid i = 1, 2, \ldots\}$ where each $|B_i|$ counts the number of data elements in $\left[\zeta^{i-1}, \zeta^i\right)$.*

Note that a bucket with $|B_i| = 0$ does not need to be stored in the sketch. In practice, we store a map of buckets to support insertion, and a sorted array of buckets to support median and MAD range calculation. On average, a single bucket occupies 48 Bytes. Thus, for a space budget of 1MB, the bucket limit M can be estimated as $\frac{1024 \times 1024}{48} \approx 20000$.

3 MAD COMPUTATION

We present the technical details of MAD computation, from CORE-Sketch construction to refinement. The pseudo-code of exact MAD computation with limited sketch size is presented in Algorithm 1 in Section 3.8.

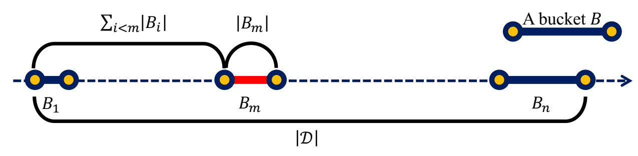
3.1 Sketch Construction

Given a dataset $\mathcal{D}$ with maximum value $e_{\max}$ and minimum value $e_{\min}$ and the bucket limit M as the input, we should first determine the base ζ for CORE-Sketch $\mathbb{S}(\zeta)$. For $\zeta \in \mathcal{Z}$, it satisfies that the sketch can cover the range $[e_{\min}, e_{\max}]$, and should be as fine as possible under the given bucket limit $|\mathbb{S}(\zeta)| \le M$.

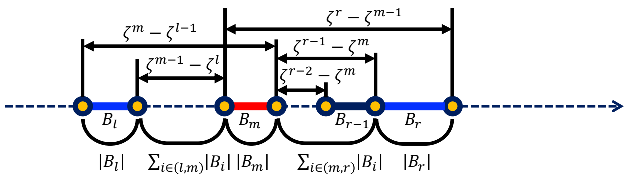
Definition 4 (Finest ζ for Dataset $\mathcal{D}$). *Given dataset $\mathcal{D}$, the finest ζ for CORE-Sketch with bucket limit M is*

$$\zeta = \min_{\zeta \in \mathcal{Z}} \left\{\zeta \mid \left\lceil \log_\zeta e_{\max} \right\rceil - \left\lceil \log_\zeta e_{\min} \right\rceil + 1 \le M\right\}. \tag{1}$$

Figure 3 illustrates how to determine the finest sketch without exceeding the bucket limit M. As shown, the constructed sketch should have buckets covering the entire range $[e_{\min}, e_{\max}]$ of the dataset $\mathcal{D}$. For any ζ, according to the definition of buckets, $e_{\min}$ drops in the bucket $B_{\lceil \log_\zeta e_{\min} \rceil}$, while $e_{\max}$ is in $B_{\lceil \log_\zeta e_{\max} \rceil}$. The number of buckets in between should not exceed M. We consider $\zeta = 2^{2^{-x}} \in \mathcal{Z}$ in the domain, starting from $x = 1$ and gradually



(a) Median bucket



(b) MAD bucket

Figure 4: Median and MAD buckets for deriving bounds

increasing $x \in \mathbb{N}$. It is to find the largest x, i.e., the minimum ζ, which still satisfies the limit M in Formula 1. The sketch in Figure 3 (a) can be further refined for more buckets, whereas the one in Figure 3 (c) is too fine having the number of buckets exceeding M. Figure 3 (b) is the finest sketch that satisfies the bucket limit.

3.2 Buckets for MAD Bound

To compute the exact MAD, it is necessary to find out the bound of the median and MAD of the dataset $\mathcal{D}$. Thus, we identify one median bucket and two MAD buckets in CORE-Sketch to give a bound for the median and MAD [4]. In the following, given the constructed CORE-Sketch $\mathbb{S}(\zeta)$ according to Section 3.1, we first give the bound of median using the median bucket in $\mathbb{S}(\zeta)$. Then, we bound MAD based on the MAD buckets in CORE-Sketch $\mathbb{S}(\zeta)$.

3.2.1 Median Bucket. Given dataset $\mathcal{D}$, we want to figure out the range of its median from CORE-Sketch $\mathbb{S}(\zeta)$. Thus, we give the definition of the median bucket in order to bound the median.

Proposition 1 (Median Bucket). *For dataset $\mathcal{D}$ and CORE-Sketch $\mathbb{S}(\zeta)$, the median bucket B_m can bound the median of $\mathcal{D}$ with its boundaries $\zeta^{m-1} \leq$ MEDIAN $< \zeta^m$, where the index m is determined by $\sum_{i<m}|B_i| < \frac{1}{2}N \leq \sum_{i\leq m}|B_i|$.*

Figure 4 (a) shows how to identify the median bucket B_m in the sketch. We aggregate the counts of buckets starting from B_1 (where $e_{\min}$ locates). When the sum first reaches $\frac{1}{2}N$, the last added bucket is indeed B_m, covering the median λ with rank $\frac{1}{2}N$.

3.2.2 MAD Bucket. We show below, given dataset $\mathcal{D}$ and median bucket B_m of CORE-Sketch $\mathcal{S}(\zeta)$, there exists some buckets in $\mathcal{S}(\zeta)$ that can bound the MAD with boundaries.

Proposition 2 (MAD Bucket). *For dataset $\mathcal{D}$ and CORE-Sketch $\mathbb{S}(\zeta)$, the MAD buckets B_l and B_r can bound MAD by their left and right boundaries together with the median bucket B_m*

$$\min\{\zeta^{m-1}-\zeta^l,\zeta^{r-1}-\zeta^m\} < \text{MAD} < \max\{\zeta^m-\zeta^{l-1},\zeta^r-\zeta^{m-1}\},$$

where the indexes l and r are determined by

$$\begin{aligned}\sum_{i\in(l,m)}|B_i| + \sum_{i\in(m,r)}|B_i| + |B_m| < \frac{1}{2}N \\ \leq \min\{|B_l|,|B_r|\} + \sum_{i\in(l,m)}|B_i| + \sum_{i\in(m,r)}|B_i| + |B_m|\end{aligned} \tag{2}$$

and one of the following two conditions

$$\begin{aligned}\zeta^{m-1} - \zeta^{l+1} \leq \zeta^{r-1} - \zeta^m \leq \zeta^{m-1} - \zeta^l, \\ \zeta^{r-2} - \zeta^m \leq \zeta^{m-1} - \zeta^l \leq \zeta^{r-1} - \zeta^m.\end{aligned} \tag{3}$$

Intuitively, Proposition 2 figures out the MAD buckets B_l and B_r, based on the median bucket B_m. As illustrated in Figure 4 (b), MAD should be greater than the distance between the right and left boundaries of B_l and B_m, i.e., $\zeta^{m-1} - \zeta^l$, or $\zeta^{r-1} - \zeta^m$ of buckets B_m and B_r. Meanwhile, MAD should be less than the distances $\zeta^m - \zeta^{l-1}$ or $\zeta^r - \zeta^{m-1}$ of B_l, B_m, B_r boundaries.

To have the aforesaid bounds on MAD, referring to the symmetric property of MAD w.r.t. the median, the MAD buckets B_l and B_r are expected to be almost equally distant from B_m. For example, in Figure 4 (b), the distance between the right boundary of B_l and the left boundary of B_m, $\zeta^{m-1} - \zeta^l$, is close to that between B_m and B_r, i.e., no less than $\zeta^{r-2} - \zeta^m$ and no greater than $\zeta^{r-1} - \zeta^m$. In other words, it satisfies the second condition of Formula 3, $\zeta^{r-2} - \zeta^m \leq \zeta^{m-1} - \zeta^l \leq \zeta^{r-1} - \zeta^m$.

In addition to distance, the total number of elements in between the right boundary of B_l and the left boundary of B_r is $\sum_{i\in(l,m)}|B_i| + \sum_{i\in(m,r)}|B_i| + |B_m|$, as shown in Figure 4 (b). To bound MAD, the count should not exceed $\frac{N}{2}$ as in Formula 2. Moreover, together with bucket B_l or B_r, the total number of elements should be no less than $\frac{N}{2}$. Hence, the MAD buckets B_l and B_r are determined by Formulas 2 and 3.

3.3 Ranges of MAD Element η

Given the median bucket B_m and MAD buckets B_l and B_r in CORE-Sketch $\mathbb{S}(\zeta)$, we can figure out the ranges of median λ and MAD element η accordingly. The idea is to determine the range of η referring to the aforesaid bounds of λ and MAD $= |\lambda - \eta|$.

3.3.1 Median Range. The median λ of the dataset $\mathcal{D}$ is naturally bounded by the median bucket B_m, referring to Proposition 1.

Proposition 3 (Median Range). *For dataset $\mathcal{D}$ and CORE-Sketch $\mathbb{S}(\zeta)$, the median range J_m, having boundaries ζ^{m-1} and ζ^m same as the median bucket B_m, bounds the median $\lambda \in J_m = \left[\zeta^{m-1},\zeta^m\right)$.*

3.3.2 MAD Element Range. For the MAD element η of dataset $\mathcal{D}$, we recall the definition of MAD given in Definition 1, MAD$(\mathcal{D}) = |\lambda - \eta|$. As the bounds of λ and $|\lambda - \eta|$ are figured out by the left and right boundaries of B_m, B_l and B_r in Propositions 1 and 2, the range of η can be derived accordingly.

Proposition 4 (MAD Element Range). *For dataset $\mathcal{D}$ and CORE-Sketch $\mathbb{S}(\zeta)$, let $d_{\min}$ and $d_{\max}$ be the minimum and maximum distances of median bucket B_m, MAD buckets B_l and B_r,*

$$d_{\min} = \min\{\max\{-\rho,\zeta^{m-1}-\zeta^l\},\max\{-\rho,\zeta^{r-1}-\zeta^m\}\}$$
$$d_{\max} = \max\{\zeta^m-\zeta^{l-1},\zeta^r-\zeta^{m-1}\}$$

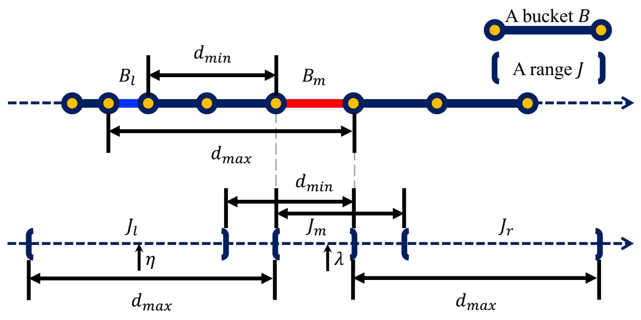


Figure 5: Ranges of median and MAD elements

where ρ is an infinitesimal. Then the MAD element η of $\mathcal{D}$ can be bounded by the ranges J_l and J_r, i.e., $\eta \in J_l \cup J_r$, where

$$J_l = \left(\zeta^{m-1} - d_{\max}, \zeta^m - d_{\min}\right)$$
$$J_r = \left(\zeta^{m-1} + d_{\min}, \zeta^m + d_{\max}\right).$$

By defining $J = J_l \cup J_m \cup J_r$, we can narrow down the ranges of median λ and MAD element η from $\mathcal{D}$ to $J \cap \mathcal{D}$. For example, in Figure 5, given $d_{\min}$ and $d_{\max}$ determined in Figure 4 (b), the median λ and MAD element η must locate in ranges J_m, J_l and J_r.

3.4 Sketch Refinement

Since J specifies the ranges of median λ and MAD element η, the CORE-Sketch $\mathbb{S}(\zeta)$ can be further refined with a smaller ζ, within the fixed bucket limit M.

Let $J^{\downarrow}$ and $J^{\uparrow}$ denote the left and right boundaries of the ranges J, respectively. It is to determine the finest $\zeta \in \mathcal{Z}$ that can cover the ranges J and still satisfies the bucket limit $|\mathbb{S}(\zeta)| \leq M$.

Definition 5 (Finest ζ for Ranges J). *Given the ranges J of median and MAD elements, the finest ζ with bucket limit M is*

$$\zeta = \min_{\zeta \in \mathcal{Z}} \left\{ \zeta \mid \sum_{p=m,l,r} \left(\left\lceil \log_\zeta J_p^{\uparrow} \right\rceil - \left\lceil \log_\zeta J_p^{\downarrow} \right\rceil + 1 \right) \leq M \right\}. \quad (4)$$

Figure 6 shows how to refine CORE-Sketch by calculating the finest ζ for the ranges J. Different from Figure 3 for the entire $\mathcal{D}$, the buckets only need to cover the ranges J. For example, for J_l, the buckets with indexes $\left\lceil \log_\zeta J_l^{\downarrow} \right\rceil$ and $\left\lceil \log_\zeta J_l^{\uparrow} \right\rceil$ cover the left and right boundaries $J_l^{\downarrow}$ and $J_l^{\uparrow}$, respectively.

3.5 Buckets for MAD Bound in Refined Sketch

Given the refined CORE-Sketch $\mathbb{S}(\zeta)$ with smaller buckets, the corresponding median bucket B_m and MAD buckets B_l and B_r should be re-identified. To identify the median and MAD buckets for the target $\mathcal{D}$, we should record the number of elements that are no longer considered in the refined sketch for ranges $J = J_l \cup J_m \cup J_r$. Thereby, we define four subsets for the elements in $\mathcal{D} \setminus J$.

$$I_1 = \mathcal{D} \cap [e_{\min}, J_l^{\downarrow}], \quad I_2 = \mathcal{D} \cap [J_l^{\uparrow}, J_m^{\downarrow})$$
$$I_3 = \mathcal{D} \cap [J_m^{\uparrow}, J_r^{\downarrow}], \quad I_4 = \mathcal{D} \cap [J_r^{\uparrow}, e_{\max}]$$

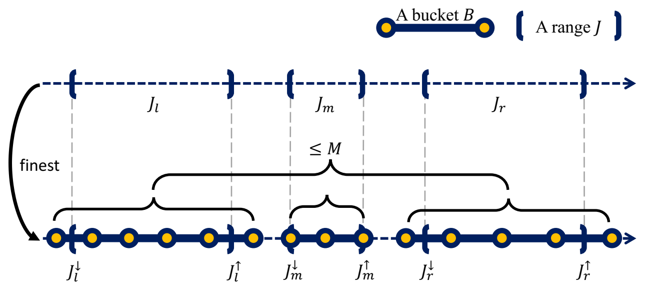
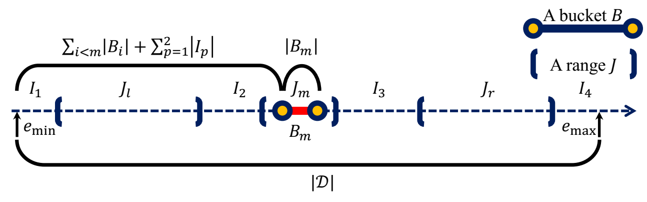
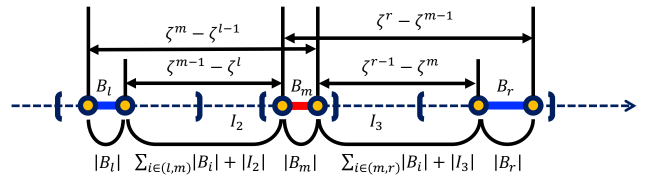


Figure 6: Determine the finest base ζ for the data ranges J



(a) Median bucket over refined sketch



(b) MAD bucket over refined sketch

Figure 7: Median and MAD buckets in the refined sketch

Dataset $\mathcal{D}$ can thus be divided into seven non-intersecting parts, $\mathcal{D} = \left(\bigcup_{p=1}^{4} I_p\right) \cup \left(\bigcup_{p=m,l,r} (J_p \cap \mathcal{D})\right)$.

3.5.1 Median Bucket in the Refined Sketch.

For dataset $\mathcal{D}$, to figure out the range of the median over the refined CORE-Sketch $\mathbb{S}(\zeta)$, we give the median bucket in the refined sketch.

Proposition 5 (Median Bucket in Refined Sketch). *For dataset $\mathcal{D}$ and the refined CORE-Sketch $\mathbb{S}(\zeta)$, the median bucket B_m bounds the median $\zeta^{m-1} \leq$ MEDIAN $< \zeta^m$, where the index m is determined by $\sum_{i<m} |B_i| + \sum_{p=1}^{2} |I_p| < \frac{1}{2}N \leq \sum_{i \leq m} |B_i| + \sum_{p=1}^{2} |I_p|$.*

Figure 7 (a) shows how to identify the median bucket B_m in the sketch. Instead of aggregating from $|B_1|$ in Figure 4 (a), we only need to aggregate the counts of buckets in the ranges J and $|I_1| + |I_2|$. Again, when the sum first reaches $\frac{1}{2}N$, the last added bucket (in J_m) is indeed B_m, covering the median λ with rank $\frac{1}{2}N$.

3.5.2 MAD Bucket in the Refined Sketch.

Given dataset $\mathcal{D}$ and median bucket B_m in CORE-Sketch $\mathbb{S}(\zeta)$, we identify the buckets that can bound MAD with their boundaries.

Proposition 6 (MAD Bucket in Refined Sketch). *For dataset $\mathcal{D}$ and the refined CORE-Sketch $\mathbb{S}(\zeta)$, the MAD buckets B_l and B_r can bound MAD with their left and right boundaries together with*

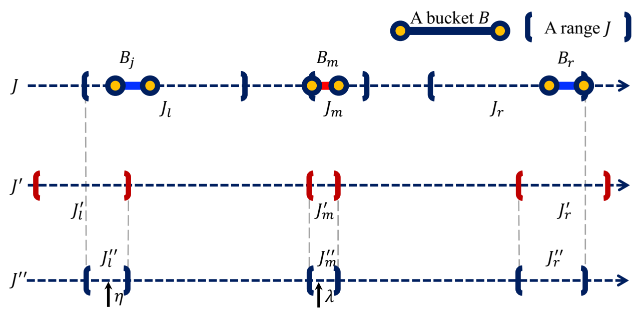


Figure 8: Shrink ranges of median λ and MAD element η

median bucket B_m in the refined sketch

$$\min\{\zeta^{m-1}-\zeta^l,\zeta^{r-1}-\zeta^m\} < \text{MAD} < \max\{\zeta^m-\zeta^{l-1},\zeta^r-\zeta^{m-1}\},$$

where the indexes l and r are determined by

$$\sum_{i\in(l,m)}|B_i| + \sum_{i\in(m,r)}|B_i| + |B_m| + \sum_{p=2}^{3}|I_p| < \frac{1}{2}N$$
$$\leq \min\{|B_l|,|B_r|\} + \sum_{i\in(l,m)}|B_i| + \sum_{i\in(m,r)}|B_i| + |B_m| + \sum_{p=2}^{3}|I_p| \tag{5}$$

and one of the following two conditions

$$\zeta^{m-1}-\zeta^{l+1} \leq \zeta^{r-1}-\zeta^m \leq \zeta^{m-1}-\zeta^l,$$
$$\zeta^{r-2}-\zeta^m \leq \zeta^{m-1}-\zeta^l \leq \zeta^{r-1}-\zeta^m. \tag{6}$$

As shown in Figure 7 (b), in the refined sketch, Proposition 6 figures out the MAD buckets B_l and B_r, based on the median bucket B_m. The MAD bound is the same as in Proposition 2. Moreover, referring to the symmetric property of MAD w.r.t. the median, the conditions in Formula 6 are also the same as in Formula 2.

Note that only the buckets in the ranges of J_l, J_m, J_r will be refined and maintained, as illustrated in Figure 7 (b). Thereby, the total number of elements lying from the right boundary of B_l to the left boundary of B_r is $\sum_{i\in(l,m)}|B_i|+\sum_{i\in(m,r)}|B_i|+|B_m|+|I_2|+|I_3|$, where $|I_2|$ and $|I_3|$ are counted in the previous iterations. Intuitively, to bound MAD, it should not exceed $\frac{N}{2}$ as in Formula 5. Likewise, together with the bucket B_l or B_r, the total number of elements exceeds $\frac{N}{2}$. Finally, the MAD buckets B_l and B_r in the refined sketch can be obtained by Formulas 5 and 6.

3.6 Ranges of MAD Element in Refined Sketch

Following the same line of Section 3.3, we can determine the new ranges J', referring to the boundaries of MAD buckets B_l and B_r together with median bucket B_m in the sketch refined for the previous ranges J. Intuitively, since both ranges J and J' specify the ranges where the median λ and the MAD element η must belong, the ranges can be further shrunk by their intersections.

Proposition 7 (MAD Element Range in Refined Sketch). *For the ranges J in sketch $\mathbb{S}(\zeta)$, and J' in the sketch $\mathbb{S}(\zeta')$ refined from $\mathbb{S}(\zeta)$, the shrunk ranges $J''=J''_l\cup J''_m\cup J''_r=(J_l\cap J'_l)\cup(J_m\cap J'_m)\cup(J_r\cap J'_r)$ still bound the median λ and MAD element η in $\mathbb{S}(\zeta')$.*

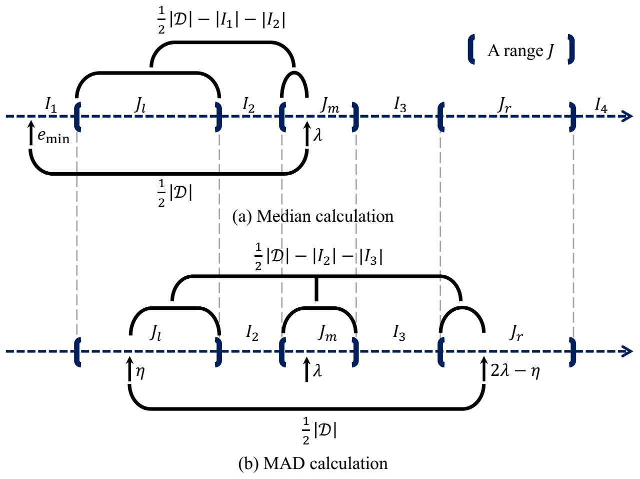


Figure 9: MAD calculation with ranges J of λ and η

Figure 8 gives an example of the shrunk ranges. Given buckets B_m, B_l and B_r in $\mathbb{S}(\zeta)$ over ranges J, the new ranges J' can be calculated for the sketch $\mathbb{S}(\zeta')$ refined from $\mathbb{S}(\zeta)$. As shown, $J''=J\cap J'$ can still bound median λ and MAD element η, referring to the property of ranges. We replace J' with the further shrunk J''.

3.7 MAD Calculation

Once the CORE-Sketch is sufficiently refined such that data elements in ranges J can fit in memory, e.g., recycling the space of CORE-Sketch, we can find exactly the median λ and the MAD element η by ranking elements in $J\cap\mathcal{D}$. Again, efficient algorithms such as Quickselect [8] can be used to determine the exact rank in the small subset $J\cap\mathcal{D}$ in memory.

Proposition 8 (MAD Calculation). *Given dataset $\mathcal{D}$ with median range J_m and MAD element ranges $J_l\cup J_r$, the exact MAD can be calculated with ranges $J=J_l\cup J_m\cup J_r$ as follows.*

(1) The exact median of $\mathcal{D}$, i.e., λ, is the element in $J\cap\mathcal{D}$ with rank $\frac{1}{2}N-|I_1|-|I_2|$.
(2) The exact MAD of $\mathcal{D}$ is the element in $\{|e-\lambda| \mid e\in J\cap\mathcal{D}\}$ with rank $\frac{1}{2}N-|I_2|-|I_3|$.

Figure 9 illustrates the process of MAD calculation. Without loss of generality, we suppose $\eta<\lambda$. As shown in Figure 9 (a), the median λ, with rank $\frac{1}{2}N$ in $\mathcal{D}$, should have rank $\frac{1}{2}N-|I_1|-|I_2|$ in $J\cap\mathcal{D}$. Likewise, in Figure 9 (b), the MAD element η should be the one with rank $\frac{1}{2}N$ in the dataset $\{|e-\lambda| \mid e\in\mathcal{D}\}$. As shown, the number of elements between η and $2\lambda-\eta$ should be exactly $\frac{1}{2}N$. Or equivalently, there should be $\frac{1}{2}N-|I_2|-|I_3|$ elements between η and $2\lambda-\eta$ in $J\cap\mathcal{D}$.

3.8 Iterative Algorithm

Finally we present the algorithm for exact MAD with limited space. It takes dataset $\mathcal{D}$ and bucket limit M as the input, gradually refines the CORE-Sketch to find the MAD element η using the aforesaid techniques, and outputs the exact MAD.

Algorithm 1 presents the iterative computation. Lines 1-2 construct the sketch by scanning the data in $\mathcal{D}$ in Section 3.1. Lines 3 searches for the median bucket B_m and MAD buckets B_l and B_r as stated in Section 3.2. Line 4 calculates the ranges J of median λ and MAD element η in Section 3.3. Line 6-7 refines the sketch by querying the data elements $J \cap \mathcal{D}$ in disk as stated in Section 3.4. Line 8 searches for the median and MAD buckets over the refined sketch as stated in Section 3.5. Line 9 calculates and shrinks the ranges J of λ and η in Section 3.3 and 3.6. Line 11 stores $J \cap \mathcal{D}$ in memory and calculates the exact MAD as stated in Section 3.7.

Algorithm 1: MAD-CORE-Sketch($\mathcal{D}$, M)

```
   Input: Dataset $\mathcal{D}$ and bucket limit $M$
   Output: MAD of dataset $\mathcal{D}$
1  $\zeta \leftarrow$ Finest-Base($[e_{\min}, e_{\max}]$, $M$);
2  $\mathbb{S}(\zeta) \leftarrow$ Construct-Sketch($\mathcal{D}$);
3  $B_m, B_l, B_r \leftarrow$ Median-MAD-Buckets($\mathbb{S}(\zeta)$);
4  $J_m, J_l, J_r \leftarrow$ Median-MAD-Element-Ranges($B_m, B_l, B_r$);
5  while Data in $J = J_l \cup J_m \cup J_r$ are too large in memory do
6      $\zeta \leftarrow$ Refined-Base($J$, $M$);
7      $\mathbb{S}(\zeta) \leftarrow$ Refine-Sketch($\mathcal{D} \cap J$);
8      $B_m, B_l, B_r \leftarrow$ Refine-Buckets($\mathbb{S}(\zeta)$);
9      $J_m, J_l, J_r \leftarrow$ Shrink-Ranges($B_m, B_l, B_r$);
10 MAD $\leftarrow$ MAD-Calculation($\mathcal{D} \cap J$);
11 return MAD;
```

4 SKETCH ANALYSIS

In this section, we present the theoretical analysis of CORE-Sketch. Mergeability is illustrated in Section 4.1 to ensure the feasibility of parallel computing. Convergence is proved in Section 4.2 to guarantee the correctness of the iterative algorithm.

4.1 Mergeability

In order to run the exact MAD computation algorithm in parallel, it is expected to construct CORE-Sketches in different partitions of the dataset, and merge them for finding the MAD element. According to Section 2.2, the boundaries of buckets in any CORE-Sketch are defined with ζ from a fixed domain and thus possibly aligned. Such buckets from different sketches with aligned boundaries can thus be directly merged without accessing the raw dataset.

Proposition 9 (Mergeability of CORE-Sketch). *For CORE-Sketch $\mathbb{S}(\zeta)$ on dataset $\mathcal{D}$ and $\mathbb{S}'(\zeta')$ on dataset $\mathcal{D}'$ with $\zeta \geq \zeta'$, the sketch $\mathbb{S}''(\zeta)$ on the merged dataset $\mathcal{D} \cup \mathcal{D}'$ always has $|B''_i| = |B_i| + \sum_{j \in \left((i-1)\log_{\zeta'}\zeta, i\log_{\zeta'}\zeta\right]} |B'_j|$.*

Figure 10 gives an example of merging. For $\zeta = (\zeta')^2$, the sketch $\mathbb{S}''(\zeta)$ on the merged dataset $\mathcal{D} \cup \mathcal{D}'$ also has base ζ. For each B''_i in $\mathbb{S}''(\zeta)$, its count has $|B''_i| = |B_i| + |B'_{2i-1}| + |B'_{2i}|$.

4.2 Convergence

Algorithm 1 proposes to gradually refine the CORE-Sketch and shrink ranges J for finding the MAD element. Convergence of the iterative computation is thus concerned, i.e., whether the sketch

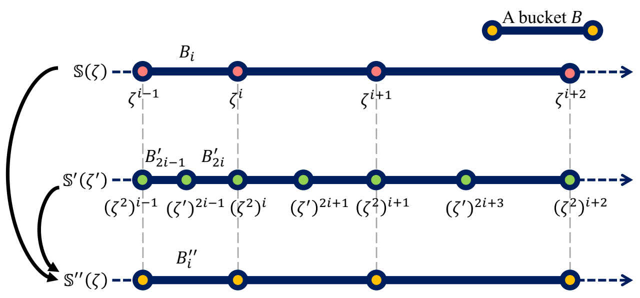


Figure 10: Merge $\mathbb{S}(\zeta)$ on dataset $\mathcal{D}$ and $\mathbb{S}'(\zeta')$ on dataset $\mathcal{D}'$

and the ranges can be reduced in each iteration. We show how ζ is reduced in each CORE-Sketch refinement in Lemma 10, and thus leads to shrunk ranges J in each iteration in Proposition 11.

Lemma 10 (Success of Refinement). *Given a sufficiently large $M \geq 2l + 2\lambda + 2\max\{\eta, \lambda - \eta\} + 8$, where l is the index of the MAD bucket B_l, in each iteration, we can always have $\zeta' < \zeta$, where ζ' is the base of CORE-Sketch $\mathbb{S}(\zeta')$ refined from the previous $\mathbb{S}(\zeta)$.*

Note that the performance of sketch with exponential-size buckets is related to the data distribution. For example, the bucket limit of DD-Sketch is determined by the CDF of the dataset [16]. Hence, it is not surprising that the bucket limit M guaranteeing the convergence depends on the true median value λ, a feature of data distribution. Specifically, as shown in Figure 3, if $\zeta \in \mathcal{Z}$ becomes smaller by refinement, the number of buckets in a particular range doubles. For a larger median λ, the length of the median bucket B_m, denoted as $d_m = \zeta^m - \zeta^{m-1}$, will be larger as well due to the exponentially increasing bucket size. Consequently, $d_{\max} - (d_{\min} - d_m)$ in Figure 5 is larger as well, resulting in wider ranges J_l and J_r. The aforesaid doubled buckets for refining the wider ranges J_l and J_r may exceed the bucket limit M, i.e., failed refinement. In other words, the bucket limit M should be enlarged for a larger median λ, in order to ensure successful refinement and thus convergence.

Proposition 11 (Convergence of Ranges). *Given a sufficiently large $M \geq l + \lambda + \max\{\eta, \lambda - \eta\} + 8$, where l is the index of the MAD bucket B_l, in each iteration, we can always have the new ranges $J' \subset J$ shrunk from the previous ranges J.*

Since the ranges J of median λ and MAD element η shrink in each iteration, the data elements in $J \cap \mathcal{D}$ will finally become small enough to store in memory, and thus find λ and η in Section 3.7. That is, the iterative computation converges.

4.3 Complexity Analysis

The space cost of CORE-Sketch is $O(M)$, referring to the bucket limit M. The time complexity is $O\left(f(C)(N + M\log M)\right)$, as analyzed in Proposition 12, where $f(C)$ in Formula 7 serves as the upper bound on the number of refinements. As shown, the larger the proportion C is, i.e., more data in $\mathcal{D}$ to fit in memory, the fewer the iterations are needed. The experiments also show that with the same M, a larger data size N needs more iterations of refinements in Figure 15 (a) and (b), and consequently higher time cost in Figure 11. On the other hand, given a larger bucket limit M, fewer iterations

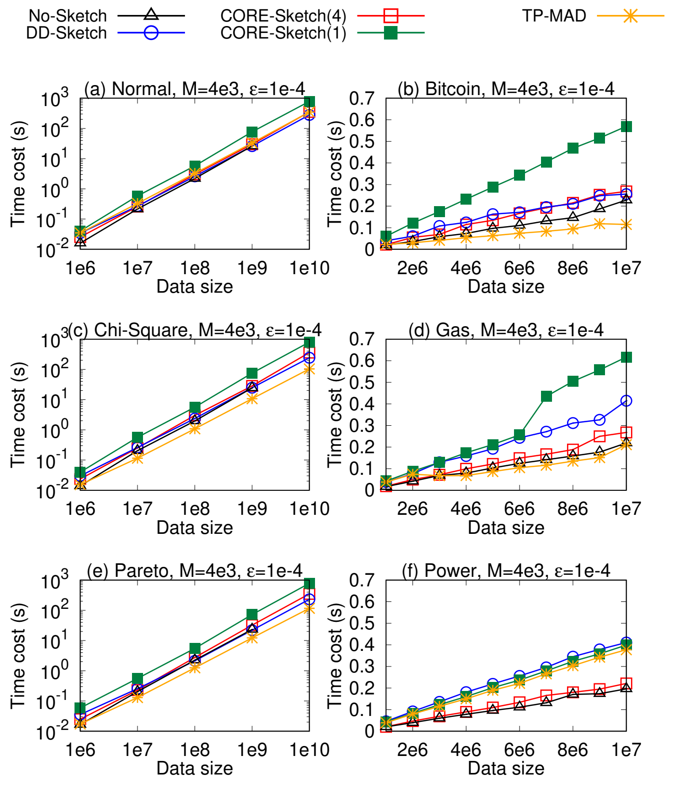


Figure 11: Time cost over data size N

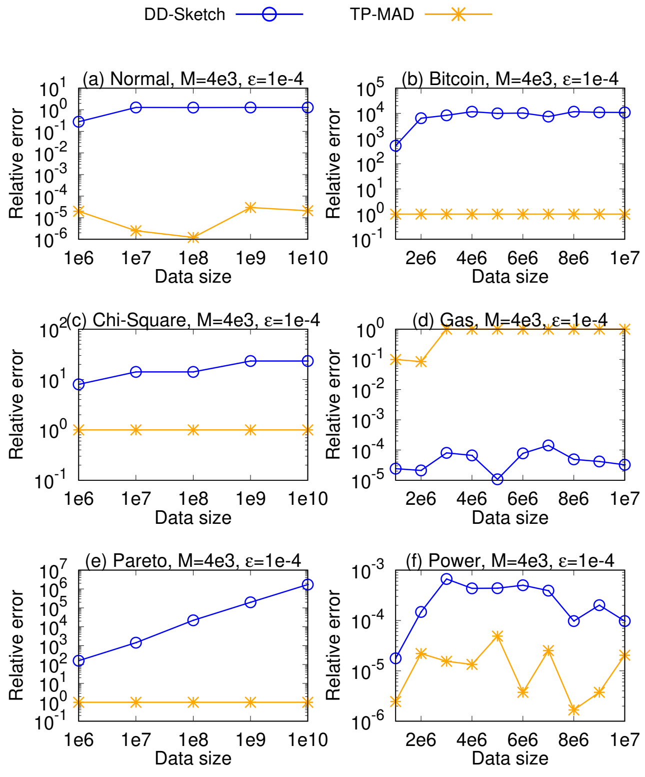


Figure 12: Relative error of approximate MAD over data size

of refinements are needed, as shown in Figure 15 (c) and (d). The corresponding time cost decreases as well in Figure 13.

Proposition 12 (Time Complexity of CORE-Sketch). *The time complexity of CORE-Sketch is $O\left(f(C)(N + M\log M)\right)$, having the upper bound on the number of refinements*

$$f(C) = -\log\log\max\left\{\zeta \mid \max_{i=1,\cdots,M-1}\left(F(\zeta^{i+1}) - F(\zeta^{i-2})\right) \le C\right\}, \tag{7}$$

where N is the data size, M is the bucket limit, C is the percentage of data that can fit in memory and F is the CDF of the dataset $\mathcal{D}$.

5 EXPERIMENTS

We implement CORE-Sketch and other baselines in Java. The experiment code and data can be found in [5] for reproducing. The experiments are conducted on a machine with 8-core 2.3 GHz CPU and 64 GB memory. We employ 3 synthetic datasets and 3 real-world datasets with various distributions as introduced in the full version technical report [5].

5.1 Comparison with Baselines

We compare CORE-Sketch with baselines introduced in Section 6, No-Sketch, DD-Sketch and TP-MAD, in Figures 11, 12, 13 and 14.

5.1.1 Comparison on Data Size. In Figure 11, when the data size of datasets Normal, Chi-Square and Pareto reaches 1e10, the No-Sketch algorithm fails to complete. The datasets are too large to load in 64 GB memory for applications. On substantially large synthetic datasets, DD-Sketch fails to give an approximate MAD within the error bound ϵ, except (d) Gas in Figure 12. In contrast, CORE-Sketch can still compute the exact MAD under limited space. It shows the effectiveness of our proposed algorithm compared with both exact No-Sketch algorithm and DD-Sketch approximation.

The time cost of TP-MAD is low in Figure 11 (b), since it fails to return an approximate MAD within the specified error bound ϵ but simply the worst error 1 as illustrated in Figure 12 (b). Indeed, with the increase in data size, the data cannot be accurately summarized by the sketch with the limited space. The relative error of TP-MAD thus increases from the specified ϵ to the worst 1, e.g., starting from 2e6 in Figure 12 (d).

5.1.2 Comparison on Bucket Limit. For the same reason, by enlarging the bucket limit M in Figure 14, the relative error of TP-MAD drops from 1 to the specified ϵ. The corresponding time cost in Figure 13 increases for obtaining the more accurate ϵ rather than the worst 1. Indeed, to obtain the specified ϵ with bucket limit M larger than 8e3 in Figure 14 (d), the time cost of TP-MAD in Figure 13 (d) is comparable to CORE-Sketch returning the exact MAD.

5.1.3 Evaluation of Single Thread Implementation. As shown in Figure 13, CORE-Sketch(1) with single thread can show competitive

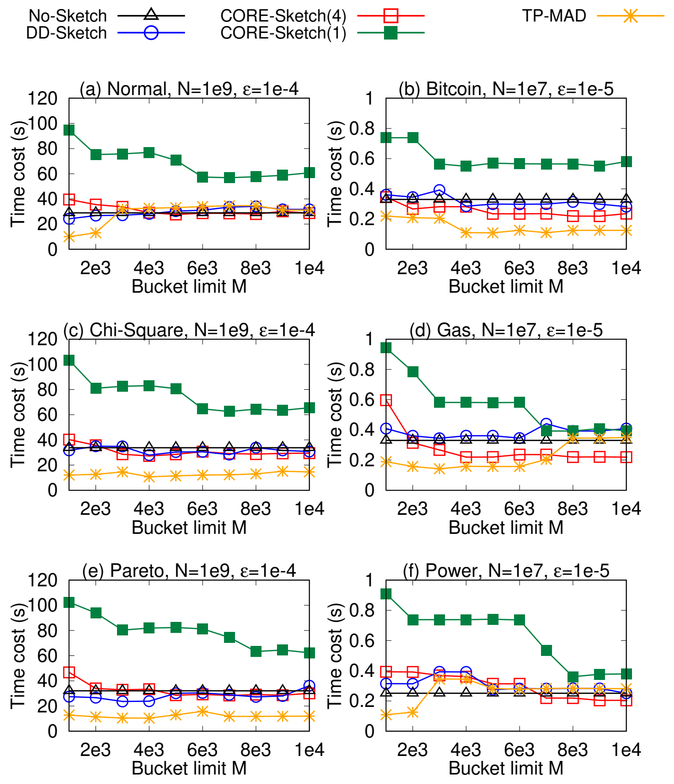


Figure 13: Time cost over bucket limit M

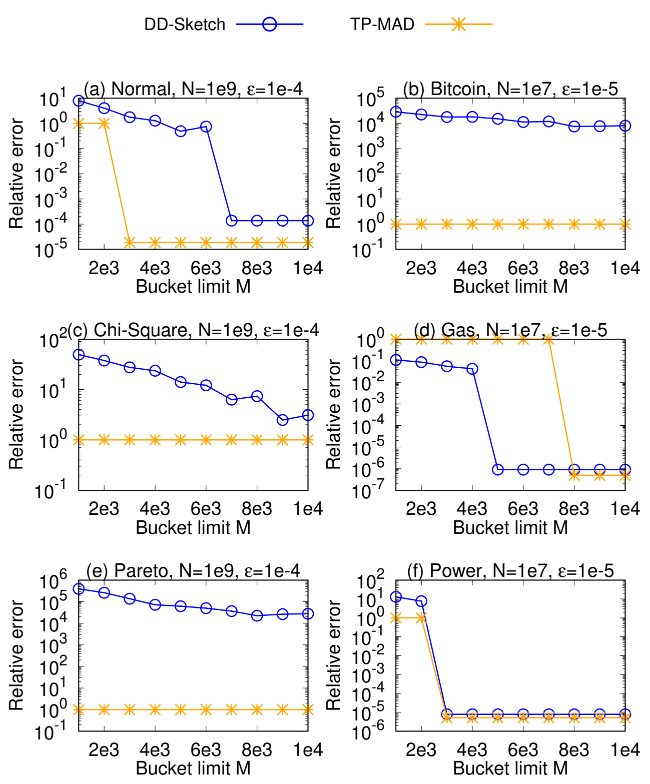


Figure 14: Relative error of approximate MAD over M

time cost compared with the baselines, when the bucket limit M is large. The reason is that a larger M leads to fewer iterations of sketch refinement, as illustrated in Figure 15 (c) and (d). For the same reason, given the same M, a larger data size means more iterations in Figure 15 (a) and (b), and higher corresponding time cost, e.g., in 1e7 of Figure 11 (d) Gas. The No-Sketch method needs up to 10^6 times larger memory (data size N / bucket limit M), i.e., a trade-off between time and space for exact MAD.

For a small bucket limit M in Figure 13, CORE-Sketch(1) is about 2 to 3 times slower than the approximate methods. The higher time cost is worthwhile, since the approximation by TP-MAD and DD-Sketch may fail to return approximate MAD within the given error bound ϵ in such a limited M. That is, as shown in Figure 14, TP-MAD and DD-Sketch give approximate MAD with rather high relative error (1 for TP-MAD and even up to 10000 for DD-Sketch in M=2e3). The results verify again the trade-off between effectiveness and efficiency by exact and approximate methods.

5.1.4 Evaluation on the Number of Iterations. As shown in Figures 15 (a) and (b), for all datasets, as the data size increases, CORE-Sketch needs to perform more rounds to complete the calculation of exact MAD. The reason is that more elements are likely to be stored in one bucket, and thus the sketch needs more refinement. The time cost of CORE-Sketch also grows with the data size due to the increase of number of rounds in Figure 11.

According to Figures 15 (c) and (d), as the buckets size increases, a fewer number of rounds are required by CORE-Sketch. The buckets

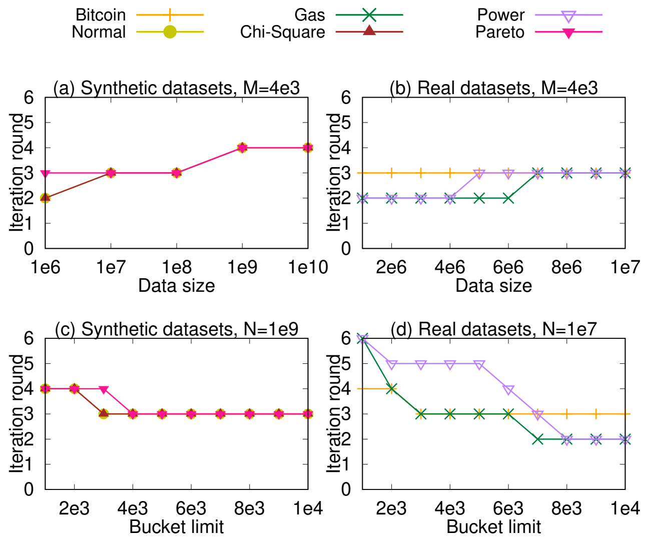


Figure 15: Refinement iteration rounds of CORE-Sketch over data size N and bucket limit M

of CORE-Sketch in each round are more refined and can store more statistical information. It also leads to the decrease in time cost of CORE-Sketch, in Figure 13, when the bucket limit grows.

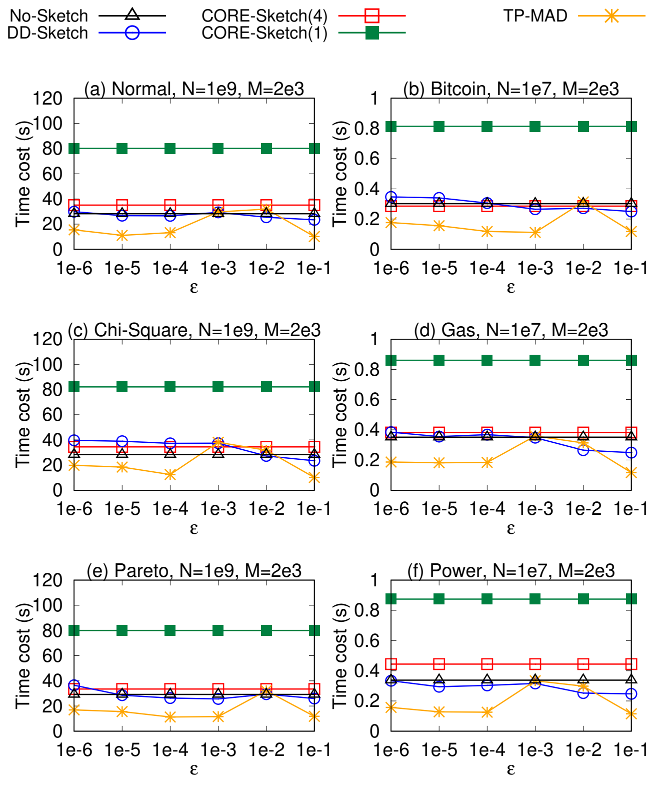


Figure 16: Time cost over various error bounds ϵ if applicable

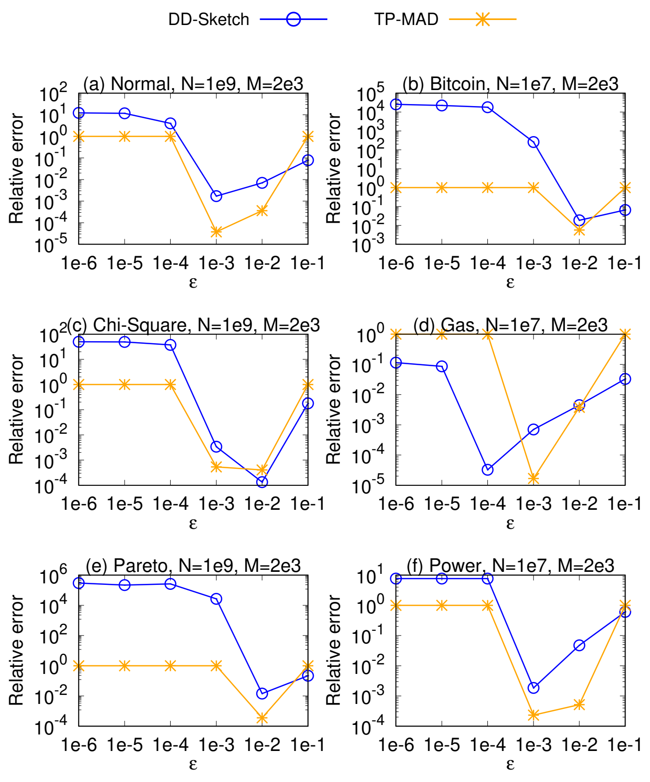


Figure 17: Relative error of approximate MAD under various error bounds ϵ

5.2 Comparison with Approximation

Compared with DD-Sketch [16] and TP-MAD [4], the contributions of our present paper are as follows. First, while DD-Sketch and TP-MAD are approximate quantile or MAD algorithms, CORE-Sketch is the first sketch to support exact MAD with comparable space.

Moreover, under limited space budget, DD-Sketch and TP-MAD may fail to return the quantile or MAD within the given error bound ϵ. This is because a small ϵ requires DD-Sketch and TP-MAD to have fine-grained buckets, in order to store the statistical information around median and MAD elements. Limited space prevents DD-Sketch and TP-MAD from allocating such fine-grained buckets. In contrast, our CORE-Sketch introduces the technique of sketch refinement to handle this problem. By refining the sketch through iterations, the buckets can be fine enough to compute the ranges of the median and MAD element.

5.2.1 Computation Performance. To compare CORE-Sketch with DD-Sketch and TP-MAD, in great detail, Figures 16 and 17 present the experiments by varying the error bound ϵ of approximation. Generally, as shown in Figure 17, given a large error bound ϵ, the relative error of the returned approximate MAD will be high. In particular, when ϵ is up to 1e-1, TP-MAD may return approximate MAD with relative error 1. The reason is that it regards median and MAD element as the same value. On the other hand, for a small error bound ϵ, the sketch fails to perform in the limited space. DD-sketch returns extremely large relative error, e.g., for ϵ=1e-6 in Figures 17 (b) and (e), whereas TP-MAD has the worst error 1.

Again, the TP-MAD time cost of such failure cases is low in Figure 16. Indeed, our CORE-Sketch(4) with four threads for exact MAD has a time cost comparable to TP-MAD when its error is successfully bounded by ϵ rather than the worst 1, e.g., for ϵ=1e-2 in Figure 16. Thereby, the advantage of the proposed CORE-Sketch(4) is to return the exact MADwith time and space costs comparable to the approximate DD-Sketch and TP-MAD having bounded error ϵ.

5.2.2 Application in Outlier Detection. We conduct qualitative experiments of outlier detection by varying error bound ϵ of approximation and resource limitation M. As in Example 1.1, we compare the outliers detected by approximate MAD with the truths by exact MAD, and report F1 score of precision and recall.

Figure 18 presents the F1 score of outlier detection, using the MAD returned by the approximate baselines under various error bounds ϵ. The results are generally analogous to the corresponding relative errors in Figure 17. A larger relative error of the approximate MAD leads to a lower F1 score of outlier detection. If the approximation methods fail to return MAD within the error bound ϵ, e.g., 1e-6 to 1e-4 in Figure 17 (a), the approximate MAD is useless with outlier detection accuracy almost 0 in Figure 18 (a).

Figure 19 illustrates the outlier detection accuracy of approximate MAD returned by DD-Sketch and TP-MAD under various bucket limit M. Again, the detection F1 score is analogous to the

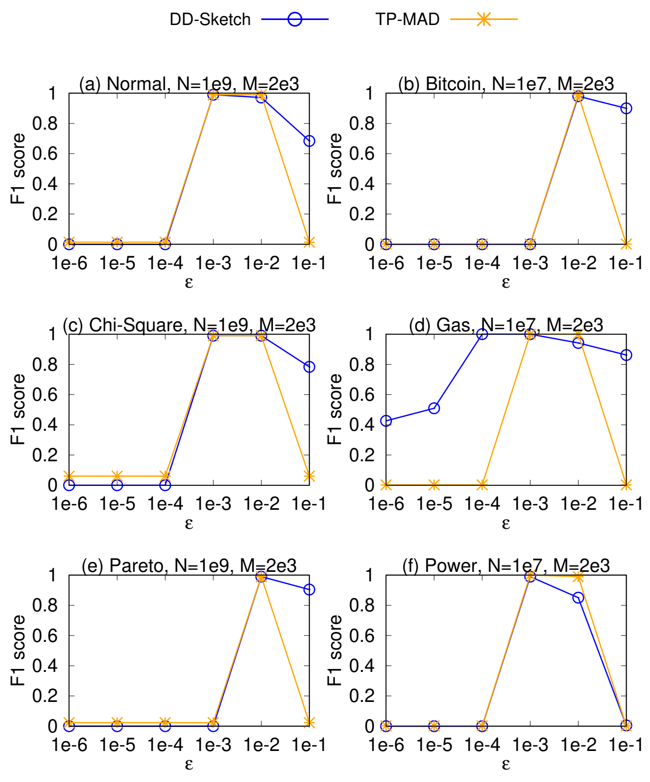


Figure 18: F1 score of outlier detection by approximate MAD under various error bounds ϵ

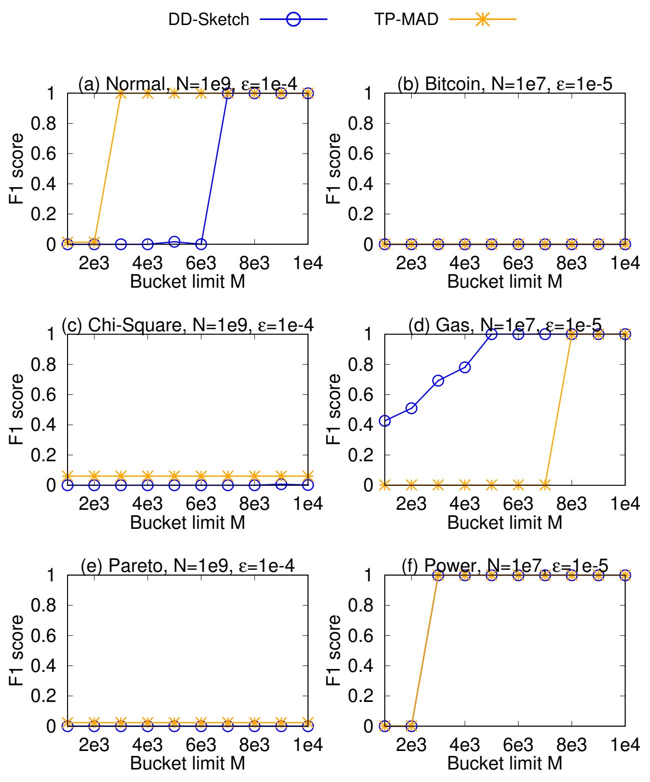


Figure 19: F1 score of outlier detection by approximate MAD under various bucket limit M

relative error in Figure 14. When the returned approximate MAD is not bounded by ϵ, e.g., for M=2e3 in Figure 14 (a), it could barely detect outliers with F1 score close to 0 in Figure 19.

The trade-off between approximation and exact computation is thus on time and space. While the approximation methods need more space to obtain MAD having low relative error in Figure 14 and consequently higher outlier detection accuracy in Figure 19, the proposed exact method CORE-Sketch needs a bit higher time cost in Figure 13. Nevertheless, CORE-Sketch(4) with four threads has time performance comparable to the approximation computation. However, the approximation methods may fail to obtain MAD with error bounded by ϵ even with M=1e4, for datasets (b) Bitcoin, (c) Chi-Square and (e) Pareto, in Figures 14 and 19.

Even worse, one may need to try different error bounds ϵ for various datasets, in order to obtain approximate MAD that is successfully bounded by ϵ, as illustrated in Figures 17 and 18. In contrast, without tuning such a parameter, the proposed method can always return the exact MADin one shot. In this sense, the proposed approach provides new insight faster.

5.3 Evaluation on System Implementation

Note that the sketch refinement technique is necessary when the memory is limited, e.g., in IoT devices or databases with heavy loads. For example, consider a trillion records of 16-byte floats. The No-Sketch method without sketch refinement needs $\frac{16\times10^{12}}{1024^4} \approx 15$ TB

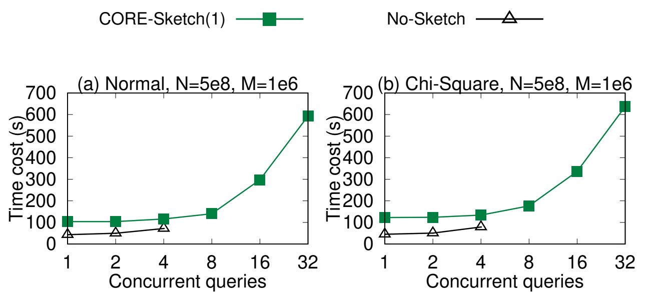


Figure 20: Concurrent MAD query processing

memory to load the data. The experiments in Figure 11 demonstrate that No-Sketch works on one billion (1e9) records with 64 GB memory for applications, but fails in ten billion (1e10) data size.

In contrast, the proposed CORE-Sketch with bucket limit needs extremely lower space cost. This major advance also enables more concurrent MAD queries in parallel. We implement CORE-Sketch(1) with single thread in Apache IoTDB [10], and compare with the No-Sketch implementation in the database. Figure 20 presents the time cost of various concurrent queries. As shown, when the number of concurrent queries reaches 8, all 64 GB memory is consumed and the No-Sketch method suffers out-of-memory (OOM) error.

The proposed CORE-Sketch with refinement technique can still perform in even 32 concurrent queries. The total memory for sketch with bucket limit $M = 1e6$ is only 48 Byte $\times 10^6 \approx 50$ MB, shared by all concurrent queries, where the size for a single bucket is 48 Bytes. Although the refinement takes some extra time cost in query processing, it is a worthwhile trade-off for the extremely lower space cost and thus supporting much more current queries.

6 RELATED WORK

6.1 MAD Computation

To show the best known upper bounds of the computational complexity of the problem, we discuss the time and space complexity of exact and approximate algorithms in Table 1.

For exact MAD, one may call QuickSelect [20] twice to compute the median and the median of deviation to median, i.e., No-Sketch with both time and space complexity $O(N)$, where N is the size of dataset $\mathcal{D}$. Note that it is often impractical to load the entire data in memory, and thus fails to perform on data size 1e10 even with 64 GB memory in Figure 11. In contrast, our CORE-Sketch has significantly lower space cost $O(M)$, but a bit higher time cost $O\left(f(C)(N + M\log M)\right)$, as analyzed in Proposition 12, where M is the maximum number of buckets, and $f(C)$ in Formula 7 serves as the upper bound on the number of refinements.

For approximate MAD, the exact contributions of the previous works [4, 16] are discussed as follows. DDSketch [4] gives a data structure to query the quantile under relative error ϵ with a success rate δ. The space complexity of DD-Sketch is $O(M)$, and the time complexity is $O(N + M\log M)$. Note that using DD-Sketch twice to compute MAD cannot give an explicit error bound for MAD. Thereby, its relative error could be extremely large in practice, e.g., in Figures 12 and 14, (b) and (e).

TP-MAD [16] extends the DD-Sketch algorithm to support approximate MAD query. It can return an $(\epsilon, 1)$-accurate MAD, i.e., the relative error is at most ϵ or 1. The space complexity of TP-MAD remains $O(M)$, while guaranteeing the $O(N + M\log M)$ time complexity. Unfortunately, TP-MAD does not support exact MAD with parameter $\epsilon = 0$, as stated in [16]. This is because the algorithm cannot construct buckets under $\epsilon = 0$, as the boundary of the buckets is computed according to $\frac{1+\epsilon}{1-\epsilon}$. The boundary of the buckets will all be 1 if we set $\epsilon = 0$, i.e., the buckets can only store elements lying in $(1, 1] = \emptyset$. In short, without the sketch refinement proposed in this study, the TP-MAD method [4] returns approximate MAD and cannot be made exact.

6.2 Database Implementation

Sketches are constructed to accelerate the in-database statistical computing. For example, DD-Sketch [16] introduces a sketch with logarithmic-sized bins, i.e., exponentially increasing widths to compute the approximate quantile of a dataset. In this study, we employ such exponential ranges in CORE-Sketch to better handle the skewed data. Likewise, KLL± [25] is designed to handle dynamic datasets with frequent updates by a combination of adaptive sampling and merging techniques. It is promising to also extend the proposed CORE-Sketch to handle frequent updates in the future.

Moreover, MAD query functions are also supported in database products. For example, the function `anomalydetection.mad()` in InfluxDB is used for MAD calculation and anomaly detection [21]. Similarly, the MAD function can be called in SQL statements in Apache IoTDB [22], e.g., `select mad(s0) from root.d0`.

Remarkably, the proposed CORE-Sketch with extremely low space cost enables more MAD queries processed in parallel. Figure 20 in Section 5.3 compares CORE-Sketch with the No-Sketch implementation in Apache IoTDB [10]. As shown, the No-Sketch method fails in more than 4 concurrent queries owing to OOM errors, while CORE-Sketch can successfully handle 32 concurrent queries.

Table 1: Complexity analysis of MAD algorithms

Algorithm	Result	Time	Space
No-Sketch	Exact	$O(N)$	$O(N)$
CORE-Sketch	Exact	$O\left(f(C)(N + M\log M)\right)$	$O(M)$
DD-Sketch	Approximate	$O(N + M\log M)$	$O(M)$
TP-MAD	Approximate	$O(N + M\log M)$	$O(M)$

7 CONCLUSIONS

In this paper, we propose a novel structure of CORE-Sketch for computing exact MAD with limited space. Instead of estimating approximate MAD by statistics in the sketch, the refinement scheme of sketch gradually narrows down the ranges of the median and MAD element. Once the ranges of data elements are small enough to be stored in memory, the exact MAD is computed. Mergeability of CORE-Sketch is illustrated, which enables parallel computation of exact MAD. Moreover, convergence of the iterative algorithm with sketch refinement is also analyzed. The extensive experiments over real and synthetic data demonstrate that our CORE-Sketch shows up to 3 order-of-magnitude improvement in space cost compared to the No-Sketch baseline. Remarkably, the time and space costs of CORE-Sketch are relatively comparable to the DD-Sketch method for estimating approximate MAD.

In CORE-Sketch, three buckets, one for median and two others for MAD, are refined at the same time to share the same scan of full data in each iteration. It is interesting to refine the median and MAD buckets separately, in order to further reduce the memory space. With a shuffle to partition the data according to buckets, one may refine the median bucket first by scanning only the corresponding data partition. Then, the MAD buckets are refined, again by scanning only the data partitions corresponding to two MAD buckets. While the buckets maintained at the same time are reduced, a substantial reduction in memory consumption, such separated refinements incur an extra cost of data shuffle in the disk, and more iterations of refinement for median and MAD buckets, respectively. It is promising to further study the trade-off in performance.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (62232005, 62072265, 62021002), the National Key Research and Development Plan (2021YFB3300500), Beijing National Research Center for Information Science and Technology (BNR2022RC01011), and Alibaba Group through Alibaba Innovative Research (AIR) Program. Shaoxu Song (https://sxsong.github.io/) is the corresponding author.

REFERENCES

[1] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein (Eds.). ACM, 93–104. https://doi.org/10.1145/342009.335388

[2] Wei Cao, Yusong Gao, Bingchen Lin, Xiaojie Feng, Yu Xie, Xiao Lou, and Peng Wang. 2018. TcpRT: Instrument and Diagnostic Analysis System for Service Quality of Cloud Databases at Massive Scale in Real-time. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 615–627. https://doi.org/10.1145/3183713.3190659

[3] Huina Chao, Huawei Li, Xiaoyu Song, Tiancheng Wang, and Xiaowei Li. 2020. Evaluating and Constraining Hardware Assertions with Absent Scenarios. *J. Comput. Sci. Technol.* 35, 5 (2020), 1198–1216. https://doi.org/10.1007/s11390-020-9708-x

[4] Zhiwei Chen, Shaoxu Song, Ziheng Wei, Jingyun Fang, and Jiang Long. 2021. Approximating Median Absolute Deviation with Bounded Error. *Proc. VLDB Endow.* 14, 11 (2021), 2114–2126. https://doi.org/10.14778/3476249.3476266

[5] Source code and full version technical report for CORE-Sketch. 2023. https://github.com/thssdb/core-sketch.

[6] Graham Cormode, Zohar S. Karnin, Edo Liberty, Justin Thaler, and Pavel Veselý. 2021. Relative Error Streaming Quantiles. In *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo (Eds.). ACM, 96–108. https://doi.org/10.1145/3452021.3458323

[7] Frank R. Hampel. 1974. The Influence Curve and its Role in Robust Estimation. *J. Amer. Statist. Assoc.* 69, 346 (1974), 383–393. https://doi.org/10.1080/01621459.1974.10482962

[8] C. A. R. Hoare. 1961. Algorithm 65: find. *Commun. ACM* 4, 7 (1961), 321–322. https://doi.org/10.1145/366622.366647

[9] Configuration in Apache IoTDB. 2023. https://iotdb.apache.org/UserGuide/Master/Reference/Common-Config-Manual.html.

[10] Implementation in Apache IoTDB. 2023. https://github.com/apache/iotdb/tree/research/core-sketch.

[11] Ashish Kamra, Evimaria Terzi, and Elisa Bertino. 2008. Detecting anomalous access patterns in relational databases. *VLDB J.* 17, 5 (2008), 1063–1077. https://doi.org/10.1007/s00778-007-0051-4

[12] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. 2015. Generic and Scalable Framework for Automated Time-series Anomaly Detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams (Eds.). ACM, 1939–1947. https://doi.org/10.1145/2783258.2788611

[13] Kim-Hung Le and Paolo Papotti. 2020. User-driven Error Detection for Time Series with Events. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020.* IEEE, 745–757. https://doi.org/10.1109/ICDE48307.2020.00070

[14] Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. 2020. MCUNet: Tiny Deep Learning on IoT Devices. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/86c51678350f656dcc7f490a43946ee5-Abstract.html

[15] Minghua Ma, Zheng Yin, Shenglin Zhang, Sheng Wang, Christopher Zheng, Xinhao Jiang, Hanwen Hu, Cheng Luo, Yilin Li, Nengjun Qiu, Feifei Li, Changcheng Chen, and Dan Pei. 2020. Diagnosing Root Causes of Intermittent Slow Queries in Large-Scale Cloud Databases. *Proc. VLDB Endow.* 13, 8 (2020), 1176–1189. https://doi.org/10.14778/3389133.3389136

[16] Charles Masson, Jee E. Rim, and Homin K. Lee. 2019. DDSketch: A Fast and Fully-Mergeable Quantile Sketch with Relative-Error Guarantees. *Proc. VLDB Endow.* 12, 12 (2019), 2195–2205. https://doi.org/10.14778/3352063.3352135

[17] René Müller and Jens Teubner. 2009. FPGA: what's in it for a database?. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, Ugur Çetintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul (Eds.). ACM, 999–1004. https://doi.org/10.1145/1559845.1559965

[18] John Paparrizos, Paul Boniol, Themis Palpanas, Ruey Tsay, Aaron J. Elmore, and Michael J. Franklin. 2022. Volume Under the Surface: A New Accuracy Evaluation Measure for Time-Series Anomaly Detection. *Proc. VLDB Endow.* 15, 11 (2022), 2774–2787. https://www.vldb.org/pvldb/vol15/p2774-paparrizos.pdf

[19] T. Pham-Gia and T.L. Hung. 2001. The mean and median absolute deviations. *Mathematical and Computer Modelling* 34, 7 (2001), 921–936. https://doi.org/10.1016/S0895-7177(01)00109-1

[20] Helmut Prodinger. 1995. Multiple Quickselect - Hoare's Find Algorithm for Several Elements. *Inf. Process. Lett.* 56, 3 (1995), 123–129. https://doi.org/10.1016/0020-0190(95)00150-B

[21] MAD query in InfluxDB. 2023. https://docs.influxdata.com/flux/v0.x/stdlib/contrib/anaisdg/anomalydetection/mad.

[22] MAD query in IoTDB. 2023. https://iotdb.apache.org/UserGuide/Master/Operators-Functions/Data-Profiling.html#mad.

[23] Durgesh Samariya and Jiangang Ma. 2022. A New Dimensionality-Unbiased Score for Efficient and Effective Outlying Aspect Mining. *Data Sci. Eng.* 7, 2 (2022), 120–135. https://doi.org/10.1007/s41019-022-00185-5

[24] Ricardo Jorge Santos, Jorge Bernardino, and Marco Vieira. 2014. Approaches and Challenges in Database Intrusion Detection. *SIGMOD Rec.* 43, 3 (2014), 36–47. https://doi.org/10.1145/2694428.2694435

[25] Fuheng Zhao, Sujaya Maiyya, Ryan Weiner, Divy Agrawal, and Amr El Abbadi. 2021. KLL±: Approximate Quantile Sketches over Dynamic Datasets. *Proc. VLDB Endow.* 14, 7 (2021), 1215–1227. https://doi.org/10.14778/3450980.3450990