# Approximating Median Absolute Deviation with Bounded Error

Zhiwei Chen, Shaoxu Song*
Tsinghua University
{czw18,sxsong}@tsinghua.edu.cn

Ziheng Wei, Jingyun Fang, Jiang Long
Data Governance Innovation Lab, HUAWEI Cloud BU
{ziheng.wei,fangjingyun,longjiang4}@huawei.com

## ABSTRACT

The median absolute deviation (MAD) is a statistic measuring the variability of a set of quantitative elements. It is known to be more robust to outliers than the standard deviation (SD), and thereby widely used in outlier detection. Computing the exact MAD however is costly, e.g., by calling an algorithm of finding median twice, with space cost $O(n)$ over $n$ elements in a set. In this paper, we propose the first fully mergeable approximate MAD algorithm, OP-MAD, with one-pass scan of the data. Remarkably, by calling the proposed algorithm at most twice, namely TP-MAD, it guarantees to return an $(\epsilon, 1)$-accurate MAD, i.e., the error relative to the exact MAD is bounded by the desired $\epsilon$ or 1. The space complexity is reduced to $O(m)$ while the time complexity is $O(n + m \log m)$, where $m$ is the size of the sketch used to compress data, related to the desired error bound $\epsilon$. To get a more accurate MAD, i.e., with smaller $\epsilon$, the sketch size $m$ will be larger, a trade-off between effectiveness and efficiency. In practice, we often have the sketch size $m \ll n$, leading to constant space cost $O(1)$ and linear time cost $O(n)$. The extensive experiments over various datasets demonstrate the superiority of our solution, e.g., 160000× less memory and 18× faster than the aforesaid exact method in datasets *pareto* and *norm*. Finally, we further implement and evaluate the parallelizable TP-MAD in Apache Spark, and the fully mergeable OP-MAD in Structured Streaming.

**PVLDB Reference Format:**
Zhiwei Chen, Shaoxu Song and Ziheng Wei, Jingyun Fang, Jiang Long. Approximating Median Absolute Deviation with Bounded Error. PVLDB, 14(11): 2114-2126, 2021.
doi:10.14778/3476249.3476266

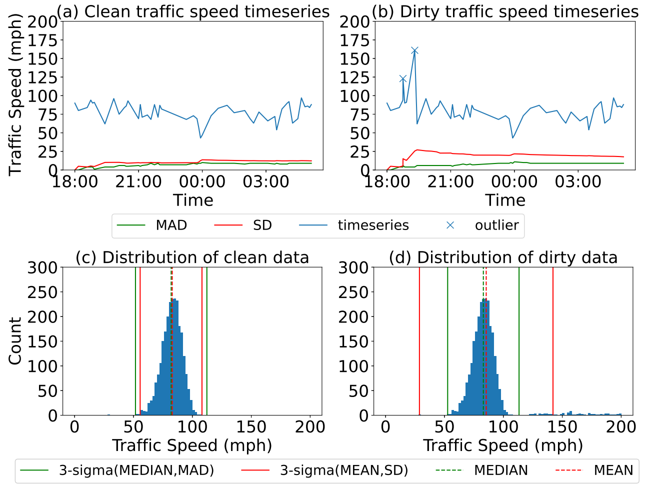

**Figure 1: Robustness of median absolute deviation (MAD) to outliers. The outliers (introduced by errors) in the time series in (b) significantly distract the SD statistics, while the MAD statistics are not affected by such outliers, close to those in the clean data in (a). When using the 3-sigma rule for outlier detection, the ranges determined by MEAN and SD over the dirty data in (d) deviate largely from the truths in (c), compared to the robust ones by MEDIAN and MAD.**

## 1 INTRODUCTION

Like the standard deviation (SD), the median absolute deviation (MAD) measures the variability of a quantitative dataset. It is defined as the median of the absolute deviations from the dataset's median. For instance, for a dataset $\mathbb{D} = \{1, 3, 3, 5, 5, 6, 9, 9, 10\}$, we have $MEDIAN = 5$. The absolute deviations from the MEDIAN 5 are $\{4, 2, 2, 0, 0, 1, 4, 4, 5\}$. The MAD of $\mathbb{D}$ is thus 2, the MEDIAN of the absolute deviations.

MAD is practically important for various applications, since it is a statistic more robust to outliers and long tails than the standard deviation (SD). For example, Cao et al. [9] use Cauchy distributions [16] with MEDIAN and MAD of historical query latency data as parameters to detect rare or abnormal query events in a database. They also apply this method to detect anomalous machines, proxy anomaly and network anomaly. Malinowski et al. [19] use MAD as a parameter to determine the number of principal factors for a data matrix in outlier detection. Wu et al. [25] and Adekeye et al. [4] calculate the limits of control charts with MAD as the scale parameter for statistical process control. Shawiesh et al. [3] and Athari et al. [7] use MAD as the estimation of variability to determine confidence intervals for locations of skewed distributions. Moreover, MAD is applied to image preprocessing, e.g., estimating noises [17] in gray-scale and colored images using MAD to select the threshold. As a popular method to preprocess high-throughput gene expression microarray data, the hit selection method uses MAD as the parameter of the z score (a hit selection threshold) to improve the overall selection rates [10]. Software engineers use MAD as the threshold to classify the defective and clean classes of datasets from monitoring systems, as an unsupervised method for software defect prediction [20].

*Example 1.1.* Figure 1 illustrates the superiority of MAD in outlier detection compared to SD. The time series in Figure 1(a) presents (a segment of) the traffic speeds in the Twin Cities Metro area in Minnesota in NAB [6]. Outliers, e.g., introduced by data errors,

*Corresponding author

Proceedings of the VLDB Endowment, Vol. 14, No. 11 ISSN 2150-8097.
doi:10.14778/3476249.3476266

significantly distract the SD statistics but not MAD, as shown in Figure 1(b). The K-sigma rule [18] with $K = 3$ detects elements outside the ranges of solid lines in the distribution as outliers, in Figure 1(d). Such ranges of solid lines can either be determined by MEAN and SD (in red), or by MEDIAN and MAD (in green). As shown in Figure 1(d), the ranges determined by MEDIAN and MAD are more robust and closer to those over the clean data in Figure 1(c). The outlier detection by MEDIAN and MAD is thus more accurate [6]. For instance, the outlier at time 19:00 with value 125, outside the range of green lines, will be detected by MEDIAN and MAD, but not by MEAN and SD in red lines. ■

Unfortunately, computing the exact MAD over a set of $n$ elements is costly, since it is defined on the median. Unlike SD, which can be efficiently calculated in constant space, the MAD computation needs to find the median of the absolute deviations from the dataset's median, i.e., finding median twice. Such an exact MAD algorithm has the space cost $O(n)$ and its time cost depends heavily on data size, which is not affordable for large scale data. Therefore, approximate MAD is often considered.

*Example 1.2.* To demonstrate the sufficiency and superiority of approximate MAD compared with exact MAD, we run the outlier detection application in Example 1.1 over a larger data size with up to $10^7$ elements. Figure 2 compares the results of approximate MAD (by Algorithm 2 in Section 3.2) and exact MAD (by Algorithm 3 in the full version technical report [1]).

(1) Comparable effectiveness. Figure 2(a) with $K = 7$ shows that the F1-score of outlier detection with approximate MAD is almost the same as that using exact MAD. That is, it is sufficient to use approximate MAD in this application.

(2) Lower time cost. Figure 2(b) presents the corresponding time costs of computing approximate MAD and exact MAD in a batch mode. While their effectiveness of outlier detection is almost the same as shown in Figure 2(a), the time cost of computing approximate MAD is only about 1/5 of exact MAD over $10^7$ elements.

(3) Lower space cost. Not only is the time cost saved, as illustrated in Figure 2(c), the corresponding space cost of computing approximate MAD is significantly lower, almost constant. Given the almost the same effectiveness but much more efficient time and space costs, it is sufficient to use approximate MAD in practice and the costly exact MAD is not necessary.

(4) Streaming computation. As presented in Section 3.2, the algorithm for computing approximate MAD can be naturally adapted to incremental computation for streaming data. In contrast, the exact MAD needs to scan the entire data, since the set of absolute deviations needs to be reconstructed given a changed median. Therefore, as illustrated in Figure 2(d), the time cost of computing approximate MAD is constant in the streaming mode, while that of exact MAD increases heavily. It demonstrates again the necessity of studying approximate MAD.

Similarly, the superiority of approximate MAD compared to exact MAD is also observed in another application of Shewhart control charts in Figure 11 in Section 5.4. ■

To find approximate MAD, a natural idea is to find the *approximate* median of the absolute deviations from the dataset's *approximate* median, by calling twice the existing methods of approximating median [15, 21]. With two median approximations, the error

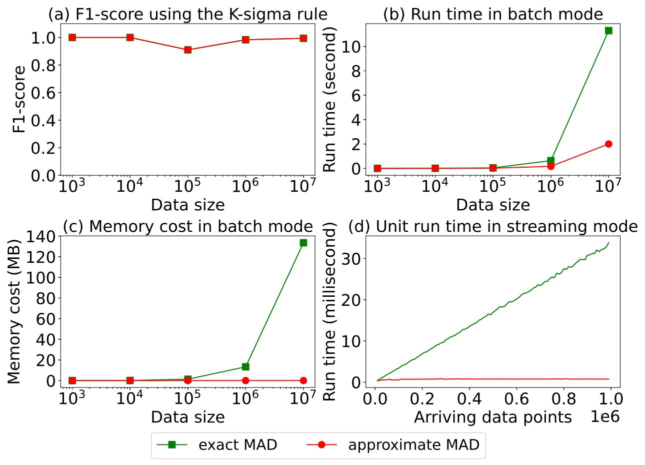

**Figure 2: Superiority of approximate MAD over exact MAD. (a) Approximate MAD and exact MAD show almost the same F1-score of outlier detection under various data sizes. The time costs in (b) and space costs in (c) of computing approximate MAD are significantly lower than those of exact MAD, in a batch mode. (d) The approximate MAD can also be incrementally computed in a streaming mode.**

introduced by such a simple approximation relative to the exact MAD is theoretically unknown and practically ineffective (see a detailed discussion in Section 6.1).

To the best of our knowledge, this is the first study on approximating MAD with bounded error. Our contributions include

- an approximate MAD algorithm with explicit error bound, which is fully mergeable and needs only 1-pass scan of data;
- a solution by calling the proposed algorithm twice, which is parallelizable and guarantees to find an $(\epsilon,1)$-accurate MAD;
- a pruning strategy to skip buckets in the sketch of the second scan, significantly reducing the space cost;
- a distributed implementation in Apache Spark and a streaming implementation in Structured Streaming, both outperforming the simple MAD approximation using the built-in *approxQuantile* for the approximate median.

Following is the structure of our paper. In Section 2, we first give the formal definition of MAD and the measure of approximate MAD in accuracy. In Section 3, we present the approximate MAD algorithm and analyze its error bound as well as complexity. In Section 4, we illustrate the solution for approximate MAD with desired error guarantees, and the bucket pruning strategy. In Section 5, we compare our solution with several baselines in experiments, showing the superiority of our proposal in accuracy and resource occupation. Finally, in Section 6 and Section 7, we discuss the related work and the conclusion.

Table 1 lists the frequently used notations.

## 2 FRAMEWORK

In this section, we review the MAD problem together with an exact computation algorithm, and introduce approximate MAD as well as the framework of our approximation solution.

Table 1: Notations

| Symbol | Description |
|---|---|
| $\hat{MAD}$ | approximate MAD |
| $\epsilon$ | the desired relative error of $\hat{MAD}$ |
| $\epsilon'$ | the returned relative error of $\hat{MAD}$ |
| $\alpha$ | the parameter of APPROX-MAD, representing the targeted relative error of $\hat{MAD}$ |
| $\beta$ | a coefficient that helps our solution return an $\epsilon$-accurate MAD in the second scan |
| $\mathbb{D}$ | a univariate set of $n$ quantitative elements, having $\mathbb{D} = \{v_i \| 1 \le i \le n\}$ |
| $\mathbb{S}$ | a sketch composed of at most $m$ buckets with the cardinality $\gamma = \frac{1+\alpha}{1-\alpha}$ |
| $\mathbb{S}^+$, $\mathbb{S}^-$ | sketches for positive and negative elements |
| $B_i$ | the bucket with the index $i$ in $\mathbb{S}$ |
| $B^0$ | the bucket with a counter for zero elements |
| $\mathbf{L}(B_i)$ | the lower boundary of $B_i$, i.e., $\mathbf{L}(B_i^+) = \gamma^{i-1}$, $\mathbf{L}(B_i^-) = -\gamma^i$ and $\mathbf{L}(B^0) = 0$ |
| $\mathbf{U}(B_i)$ | the upper boundary of $B_i$, i.e., $\mathbf{U}(B_i^+) = \gamma^i$, $\mathbf{U}(B_i^-) = -\gamma^{i-1}$ and $\mathbf{U}(B^0) = 0$ |
| $b_i$ | the number of elements in $B_i$ |
| $B_p$ | the bucket covering $MEDIAN$ |
| $B_q$ | the bucket used to estimate the range of MAD along with $B_p$, determined by Algorithm 1 |

## 2.1 Exact Median Absolute Deviation

Formally, MAD is defined as Definition 1.

Definition 1. *The median absolute deviation (MAD) of a univariate set, $\mathbb{D} = \{v_1, v_2, v_3, \dots, v_n\}$, is the median of $|v_i - MEDIAN|$, for $1 \le i \le n$, where $MEDIAN$ is the median of $\mathbb{D}$.*

According to the definition, the straightforward and traditional method to query MAD is to sort $\mathbb{D}$ to get exact MEDIAN and sort the dataset $\mathbb{D}' = \{|v_i - MEDIAN| | 1 \le i \le n\}$ to get its exact median, which is MAD of $\mathbb{D}$. (See Algorithm 3 in the full version technical report [1].)

As the exact algorithm requires space to store the entire dataset, its space complexity is $O(n)$, which is not affordable for large data. Also, its time cost depends heavily on the data size. Thus, we introduce approximation to address the MAD calculation.

## 2.2 Approximate Median Absolute Deviation

Following the same line of approximate quantiles [21], we study the relative error of approximate MAD. An $\epsilon$-accurate approximate MAD is defined as Definition 2.

Definition 2. *An $\epsilon$-accurate approximate median absolute deviation, $\hat{MAD}$, of the dataset $\mathbb{D}$ is bounded by*

$$|\hat{MAD} - MAD| \le \epsilon MAD$$

*where $MAD$ is the exact median absolute deviation of $\mathbb{D}$.*

Next, we introduce our approximate MAD solution with relative error bound guarantees. Most importantly, our solution does not require storing all data into memory. Instead, its space occupation can be bounded by a relatively small constant for data from most distributions. Moreover, the solution supports parallelization, i.e., a large dataset can be processed with the same accuracy by combining the results over several partitions. It is thus implemented and evaluated in Apache Spark in Section 5.3.3, which can be further applied in a clusterix-like database management system [22].

Figure 3 illustrates the framework of our solution that calculates an $\epsilon$-accurate approximate MAD. Its core is APPROX-MAD, a one-pass algorithm we propose to compute approximate MAD along with its error bound. The solution works as follows.

(1) Execute APPROX-MAD with the desired error bound, $\epsilon$, as its parameter. It would return an approximate MAD, $MAD_1$, and its error bound $\epsilon'$ which may exceed $\epsilon$.
(2) If $\epsilon' \le \epsilon$, $MAD_1$ is absolutely an $\epsilon$-accurate approximate MAD and thus the final result, i.e., branch (a) in Figure 3.
(3) Otherwise, it would determine whether an $\epsilon$-accurate result is available for this dataset with an auto-calculated coefficient $\beta$ according to Proposition 4.1. When the data is highly concentrated, i.e., $\beta \le 0$ or $\beta \ge 1$, the elements in the dataset are close with each other, in other words, close to the $MEDIAN$, having $\frac{MAD}{|MEDIAN|} \approx 0$. The extreme case is that all the elements in the dataset have the same (constant) value, i.e., $MAD = 0$. Intuitively, in such cases, we may directly assign the approximate MAD as $\hat{MAD} = 0$. Remarkably, referring to Definition 2, this simple approximation indeed returns a 1-accurate MAD. Therefore, for $\beta \le 0$ or $\beta \ge 1$, it outputs $\langle 0, 1 \rangle$, denoting $\hat{MAD} = 0$ with error bound 1, in branch (c) in Figure 3.
(4) In other cases, the second scan would be posed using APPROX-MAD with the updated parameter $\beta\epsilon$. It returns an $\epsilon$-accurate $MAD_2$ as the final result, i.e., branch (b) in Figure 3.

As analyzed in Section 3.4, the time and space complexities of Algorithm 2 APPROX-MAD by one-pass through the data are $O(n + m \log m)$ and $O(m)$, respectively, where $n$ is the number of elements and $m$ is the size of the sketch used to compress data. As aforesaid, by calling APPROX-MAD at most twice in Figure 3, our solution guarantees to return an $(\epsilon, 1)$-accurate MAD, with time cost $O(n + m \log m)$ and space cost $O(m)$.

## 3 FIRST SCAN

In this section, we present the algorithm, APPROX-MAD, for approximate MAD of a univariate dataset $\mathbb{D}$ with one-pass scan. It has a parameter $\alpha$ representing its targeted error bound. For the first scan, we set $\alpha = \epsilon$, the desired error bound. Two values would be output by APPROX-MAD, an approximate MAD and its corresponding error bound $\epsilon'$. Note that the approximate MAD returned in the first scan is $\epsilon'$-accurate, rather than $\epsilon$-accurate (which will be achieved by the second scan in Section 4).

The algorithm has two stages. The first sketch stage is to scan elements, allocate them into multiple buckets, and keep their count in each bucket. In the next stage, it calculates MAD based on the counts and boundaries of buckets. We introduce the sketch structure as preliminary in Section 3.1 and Algorithm 1 in Section 3.2,

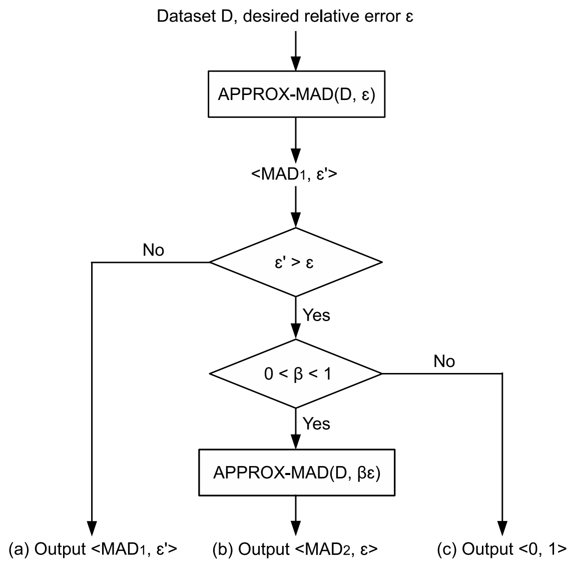

**Figure 3: The framework to get an $(\epsilon, 1)$-accurate MAD, where $\epsilon'$ is the output error bound of $MAD_1$ in the first scan, and $\beta$ is used in the input error bound of the second scan, determined automatically according to Proposition 4.1. When the data is highly concentrated, i.e., $\beta \le 0$ or $\beta \ge 1$, it directly outputs 0 which is a 1-accurate MAD.**

for the two stages respectively. Putting them together, we get the complete algorithm. Its error bound and complexity are discussed in Section 3.3 and Section 3.4.

## 3.1 Sketch Structure

Since storing the entire dataset in memory is intolerable for big data as discussed in Section 2.1, we consider using a sketch structure from which we can infer data distribution at a relatively low cost of space. We employ the sketch structure in DDSketch [21], which is fully mergeable and used for quantile computation with relative error bound guarantees. Its basic idea is bucketing and estimating elements using buckets' boundaries. We note that this structure is potentially applicable for MAD computation. With similar ideas, MAD can be estimated by two buckets' distance as it is computed by the distance between an element and MEDIAN. Besides, its simplicity helps us guarantee high efficiency in time and space (nevertheless, we propose an optimization strategy to further improve the efficiency for MAD computation in Section 4.2). Now let us introduce the sketch as preliminary. The original sketch [21] is only for positive elements as defined below.

Definition 3. *With the cardinality $\gamma = (1+\alpha)/(1-\alpha)$ and a limit m on the number of buckets, where $\alpha$ is the targeted relative error bound, a sketch for positive elements is defined as*

$$\mathbb{S}^+ = \{B^+_{i_1}, B^+_{i_2}, B^+_{i_3}, \dots\},$$

*where $|B^+_i| = b^+_i$ counts $v$ with $\gamma^{i-1} < v \le \gamma^i$, and $|\mathbb{S}^+| \le m$.*

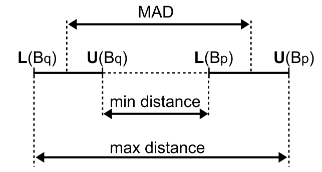

**Figure 4: The exact MAD is bounded by the minimum and the maximum distances between buckets $B_p$ and $B_q$ referring to Lemma 3.3.**

If $|\mathbb{S}^+| = m + 1$ or larger, the bucket with the lowest index, i.e., $B^+_{i_1}$, is merged into its neighbor $B^+_{i_2}$ with $b^+_{i_2} = b^+_{i_1} + b^+_{i_2}$, in order to keep at most $m$ buckets.

For negative elements, a second sketch would be kept whose indices are calculated on the absolute values. Its merging process starts from the highest indices. To distinguish two sketches, we denote them as $\mathbb{S}^+$ and $\mathbb{S}^-$. For zero elements, a separate bucket $B^0$ is kept to count them. We use $\mathbb{S}$ to denote their combination for the entire dataset. Here is an example to illustrate how it works.

*Example 3.1.* We consider a sketch $\mathbb{S}$ with $\alpha = 0.01$, making $\gamma = 1.02$, and $m = 5$. It scans $\mathbb{D} = \{1, 3, 3, 5, 5, 6, 9, 9, 10\}$. For $v_1 = 1$ and $v_2 = 3$, they fall into $B^+_0$ ($\lceil \log_\gamma(1) \rceil = 0$) and $B^+_{55}$ ($\lceil \log_\gamma(3) \rceil = 55$). Thus, we increment $b^+_0$ and $b^+_{55}$. In this way, we have $\mathbb{S} = \{B^+_0(1), B^+_{55}(2), B^+_{81}(2), B^+_{90}(1), B^+_{110}(2), B^+_{116}(1)\}$ where $b^+_i$ is within parentheses. Because $|\mathbb{S}|$ exceeds 5, the bucket with the lowest index, i.e., $B^+_0$, is merged into $B^+_{55}$, updating $b^+_{55} = 3$. Finally, we have $\mathbb{S} = \{B^+_{55}(3), B^+_{81}(2), B^+_{90}(1), B^+_{110}(2), B^+_{116}(1)\}$. ■

We denote the scanning process that inserts elements as INSERT. As a parameter to balance the capability and efficiency of $\mathbb{S}$, $m$ has been comprehensively discussed in [21]. Therefore, we will not include $m$ in our following algorithms for convenience but only discuss it when analyzing complexity in Section 3.4.

## 3.2 APPROX-MAD

In this section, we introduce the approximate MAD algorithm. Recall that we keep the number of elements within each bucket $B$ in a sketch $\mathbb{S}$. The idea is to find two buckets in $\mathbb{S}$, denoted as $B_p$ and $B_q$, whose minimum and maximum distance can bound the exact MAD, as shown in Figure 4. Then we calculate approximate MAD using their lower and upper boundaries, denoted as $\mathbf{L}(B)$ and $\mathbf{U}(B)$.

Note that for buckets $B^+_i \in \mathbb{S}^+$, $\mathbf{L}(B^+_i)$ and $\mathbf{U}(B^+_i)$ are $\gamma^{i-1}$ and $\gamma^i$. For buckets $B^-_i \in \mathbb{S}^-$, $\mathbf{L}(B^-_i)$ and $\mathbf{U}(B^-_i)$ are $-\gamma^i$ and $-\gamma^{i-1}$. And boundaries of $B^0$ are both 0. We use Algorithm 1, to locate the two buckets, $B_p$ and $B_q$.

The algorithm runs as follows.

(1) We first union $\mathbb{S}^-$, $B^0$ and $\mathbb{S}^+$ to a general sketch $\mathbb{S}$ (Line 1).
(2) Then we find the bucket $B_p$ *holding* MEDIAN using Algorithm 4 in the full version technical report [1], which is the progress of DDSketch to find 0.5-quantile [21] (Line 2).
(3) Regarding $B_p$ as the center of $\mathbb{S}$, we divide $\mathbb{S}$ into two parts, the left sketch $\mathbb{S}_l$ and the right sketch $\mathbb{S}_r$ (Lines 5-6).

**Algorithm 1:** FIND-MAD-BUCKETS ($\mathbb{S}^+, \mathbb{S}^-, B^0, n$)

```
Input: $\mathbb{S}^+$ for positive elements, $\mathbb{S}^-$ for positive elements,
       $B^0$ counting zero elements, and the data size $n$
Output: two buckets, $B_p$ and $B_q$
1  $\mathbb{S} \leftarrow \mathbb{S}^- \cup \{B^0\} \cup \mathbb{S}^+$;
2  $B_p \leftarrow$ FIND-MEDIAN-BUCKET($\mathbb{S}, n$);
3  $B_q \leftarrow B_p$;
4  $count_{mad} \leftarrow b_p$;
5  $\mathbb{S}_l \leftarrow \{B_i | B_i \in \mathbb{S} \land \mathbf{L}(B_i) < \mathbf{L}(B_p)\}$;
6  $\mathbb{S}_r \leftarrow \{B_i | B_i \in \mathbb{S} \land \mathbf{L}(B_i) > \mathbf{L}(B_p)\}$;
7  $B_l \leftarrow \arg\max_{B \in \mathbb{S}_l} \mathbf{L}(B)$;
8  $B_r \leftarrow \arg\min_{B \in \mathbb{S}_r} \mathbf{L}(B)$;
9  while $count_{mad} < \lfloor \frac{n}{2} \rfloor$ do
10     if $\mathbf{L}(B_p) - \mathbf{U}(B_l) < \mathbf{L}(B_r) - \mathbf{U}(B_p)$ then
11         $B_q \leftarrow B_l$;
12         $\mathbb{S}_l \leftarrow \mathbb{S}_l - B_l$;
13         $B_l \leftarrow \arg\max_{B \in \mathbb{S}_l} \mathbf{L}(B)$;
14     else
15         $B_q \leftarrow B_r$;
16         $\mathbb{S}_r \leftarrow \mathbb{S}_r - B_r$;
17         $B_r \leftarrow \arg\min_{B \in \mathbb{S}_r} \mathbf{L}(B)$;
18     $count_{mad} \leftarrow count_{mad} + b_q$;
19 Find $B_{q'}$ covering $\mathbf{L}(B_p) + \mathbf{U}(B_p) - \mathbf{U}(B_q)$;
20 if $\mathbf{U}(B_{q'}) \geq \mathbf{L}(B_p) + \mathbf{U}(B_p) - \mathbf{L}(B_q)$ then
21     $B_q \leftarrow B_{q'}$;
22 return $B_p$, $B_q$;
```

(4) We use $count_{mad}$ to denote the number of elements nearby $MEDIAN$ and start searching from buckets which are the closest to $B_p$ in $\mathbb{S}_l$ and $\mathbb{S}_r$, denoted as $B_l$ and $B_r$ (Lines 7-8).
(5) In each iteration, from $B_l$ and $B_r$ we choose the bucket with smaller minimum distance and update $count_{mad}$ by adding the bucket' counter (Lines 9-18). Once $count_{mad}$ exceeds $\lfloor \frac{n}{2} \rfloor$, the searching terminates.
(6) There may exist a bucket $B_{q'}$ on the other side of $B_p$ that is searched before $B_q$ but has larger maximum distance from $B_p$ than $B_q$. In this case, it should be $B_{q'}$ that makes $count_{mad}$ exceed $\lfloor \frac{n}{2} \rfloor$ rather than $B_q$. We check this case in Line 19-21.

With $B_p$ and $B_q$ returned by Algorithm 1, we calculate approximate MAD in two cases. If $B_p$ and $B_q$ are from the same sketch ($\mathbb{S}^+$ or $\mathbb{S}^-$), we set

$$\hat{MAD} = 2\frac{(\mathbf{U}(B_p) - \mathbf{L}(B_q))(\mathbf{L}(B_p) - \mathbf{U}(B_q))}{(\gamma + 1)|\mathbf{L}(B_p) - \mathbf{L}(B_q)|}. \quad (1)$$

Otherwise, we set

$$\hat{MAD} = 2\frac{\max(|\mathbf{U}(B_p) - \mathbf{L}(B_q)|, |\mathbf{L}(B_p) - \mathbf{U}(B_q)|)}{\gamma + 1}. \quad (2)$$

We will discuss why we can bound the error of $\hat{MAD}$ with these equations in Section 3.3.2. Note that in Equation 1, we assume $p \neq q$. If $p = q$, it has $MAD \leq \gamma^p - \gamma^{p-1} < \frac{2\alpha}{1-\alpha}|MEDIAN|$, where $\alpha$ is set to a value close to 0. For such a situation, we believe the data is too concentrated, inferring $\frac{MAD}{|MEDIAN|} \approx 0$, so that 0 would be the approximation. The entire procedure of APPROX-MAD is given in Algorithm 2.

**Algorithm 2:** APPROX-MAD ($\mathbb{D}, \alpha$)

```
Input: the input dataset $\mathbb{D}$ and the targeted error bound $\alpha$
Output: approximate $\hat{MAD}$ and its error bound $\epsilon'$
1  $n \leftarrow 0$;
2  $\gamma \leftarrow \frac{1+\alpha}{1-\alpha}$;
3  create sketches $\mathbb{S}^+$ and $\mathbb{S}^-$ with cardinality $\gamma$;
4  initialize $B^0$ with $b^0 = 0$;
5  for $v_i \in \mathbb{D}$ do
6      $n \leftarrow n + 1$;
7      if $v_i > 0$ then
8          INSERT($\mathbb{S}^+, v_i$);
9      else if $v_i < 0$ then
10         INSERT($\mathbb{S}^-, |v_i|$);
11     else
12         $b^0 \leftarrow b^0 + 1$
13 $B_p, B_q \leftarrow$ FIND-MAD-BUCKETS($\mathbb{S}^+, \mathbb{S}^-, B^0, n$);
14 if $B_p$ and $B_q$ are from the same sketch then
15     if $p \neq q$ then
16         $\hat{MAD} \leftarrow 2\frac{(\gamma^p - \gamma^{q-1})(\gamma^{p-1} - \gamma^q)}{(\gamma+1)|\gamma^{p-1} - \gamma^{q-1}|}$;
17         $\epsilon' \leftarrow \frac{\gamma^{|p-q|}+1}{\gamma^{|p-q|}-1}\alpha$;
18     else
19         $\hat{MAD} \leftarrow 0$;
20         $\epsilon' \leftarrow 1$;
21 else
22     $\hat{MAD} \leftarrow 2\frac{\max(|\mathbf{U}(B_p) - \mathbf{L}(B_q)|, |\mathbf{L}(B_p) - \mathbf{U}(B_q)|)}{\gamma+1}$;
23     $\epsilon' \leftarrow \alpha$;
24 return $\hat{MAD}$, $\epsilon'$;
```

*Example 3.2.* For $\mathbb{D} = \{1, 3, 3, 5, 5, 6, 9, 9, 10\}$ and $\alpha = 0.01$, its MEDIAN and MAD are 5 and 2. After scanning, we have the sketch $\mathbb{S} = \{B_0^+(1), B_{55}^+(2), B_{81}^+(2), B_{90}^+(1), B_{110}^+(2), B_{116}^+(1)\}$. We get $B_{81}^+$ that covers 5 as $B_p$ (step 1-2). Next, the searching for $B_q$ starts from $B_{55}^+$ and $B_{90}^+$, and stops until finding $B_{55}^+$ that covers 3 as $B_q$ (step 3-5). The condition in step 6 does not hold. Therefore, with $B_{81}^+$ and $B_{55}^+$, we estimate $\hat{MAD} = 2.0257$. ■

## 3.3 Accuracy Analysis

With Algorithm 2 for an approximate MAD, we next analyze the relative error bound of the MAD result. We first prove that the exact MAD is bounded by the minimum and the maximum distance between $B_p$ and $B_q$ referring to Lemma 3.3 in Section 3.3.1. Then, with the scope of MAD determined by $B_p$ and $B_q$, Section 3.3.2 presents the error bound of approximate MAD defined in Equation 1 and Equation 2 in Proposition 3.4 and Proposition 3.5, respectively. Finally, in Section 3.3.3, we give the worst-case error bound in Proposition 3.6 of the aforesaid results.

*3.3.1 Bounds of MAD.* First, we can bound MAD as follows.

LEMMA 3.3. *Given the output of Algorithm 1, $B_p$ and $B_q$, MAD is bounded by the minimum and the maximum distance between $B_p$ and $B_q$ as illustrated in Figure 4, i.e.,*

$$\mathbf{L}(B_p) - \mathbf{U}(B_q) \le MAD \le \mathbf{U}(B_p) - \mathbf{L}(B_q) \quad \text{if } \mathbf{L}(B_p) \ge \mathbf{L}(B_q),$$
$$\mathbf{L}(B_q) - \mathbf{U}(B_p) \le MAD \le \mathbf{U}(B_q) - \mathbf{L}(B_p) \quad \text{if } \mathbf{L}(B_p) < \mathbf{L}(B_q).$$

*(See proof in the full version technical report [1].)* ■

*3.3.2 Error Bound.* Note that $B_p$ and $B_q$ could be either from the same sketch ($\mathbb{S}^+$ or $\mathbb{S}^-$) or not, and we give different formulas for approximate MAD in Equation 1 and Equation 2, respectively. Now we discuss the error bound of MAD in these two cases, in Proposition 3.4 and Proposition 3.5.

For the first case, i.e., $B_p$ and $B_q$ are from the same sketch, the bound of MAD in Lemma 3.3 can be written as $\gamma^{max(p,q)-1} - \gamma^{min(p,q)} \le MAD \le \gamma^{max(p,q)} - \gamma^{min(p,q)-1}$, based on which we have the following proposition.

PROPOSITION 3.4. *Given $p, q$ with $p \ne q$, $\alpha > 0$ and $\gamma = \frac{1+\alpha}{1-\alpha}$, let $\hat{MAD} = 2\frac{(\gamma^p-\gamma^{q-1})(\gamma^{p-1}-\gamma^q)}{(\gamma+1)|\gamma^{p-1}-\gamma^{q-1}|}$. Then, we have*

$$|\hat{MAD} - MAD| \le \frac{\gamma^{|p-q|}+1}{\gamma^{|p-q|}-1}\alpha MAD$$

*if $\gamma^{max(p,q)-1} - \gamma^{min(p,q)} \le MAD \le \gamma^{max(p,q)} - \gamma^{min(p,q)-1}$.*
*(See proof in the full version technical report [1].)* ■

Proposition 3.4 shows that if $B_p$ and $B_q$ are from the same sketch, we estimate MAD by APPROX-MAD($\mathbb{D}, \epsilon$) with error bounded by $\epsilon' = \frac{\gamma^{|p-q|}+1}{\gamma^{|p-q|}-1}\epsilon$ only if $p \ne q$, i.e., $MAD \ge \frac{2\epsilon}{1-\epsilon}|MEDIAN|$. However, the error bound, affected by $\gamma^{|p-q|}$, exceeds $\epsilon$. Note that $|MEDIAN| \approx \gamma^p$ and $MAD \approx |\gamma^p - \gamma^q|$, and thereby $\gamma^{|p-q|} \approx 1 + \frac{MAD}{|MEDIAN|}$ or $1/(1 - \frac{MAD}{|MEDIAN|})$. With a fixed $\epsilon$, if $\frac{MAD}{|MEDIAN|}$ is smaller, i.e., data is more concentrated, APPROX-MAD would get a smaller $\gamma^{|p-q|}$ and thereby output an approximate MAD with larger $\epsilon'$.

For the other case, i.e., one of $B_p$ and $B_q$ is from $\mathbb{S}^+$ and the other is from $\mathbb{S}^-$, the bound of MAD in Lemma 3.3 can be written $\gamma^{p-1} + \gamma^{q-1} \le MAD \le \gamma^p + \gamma^q$. It leads to the error bound below.

PROPOSITION 3.5. *Given $p, q$, $\alpha > 0$ and $\gamma = \frac{1+\alpha}{1-\alpha}$, let $\hat{MAD} = 2\frac{\gamma^p+\gamma^q}{\gamma+1}$. We have*

$$|\hat{MAD} - MAD| \le \alpha MAD$$

*if $\gamma^{p-1} + \gamma^{q-1} \le MAD \le \gamma^p + \gamma^q$.*
*(See proof in the full version technical report [1].)* ■

According to Proposition 3.5, APPROX-MAD($\mathbb{D}, \epsilon$) returns an $\epsilon$-accurate MAD, when one of $B_p$ and $B_q$ is from $\mathbb{S}^+$ and the other is from $\mathbb{S}^-$. Besides, if either $B_p$ or $B_q$ is $B^0$, whose boundaries are both 0, we have $\gamma^{p-1} \le MAD \le \gamma^p$ (suppose $B_q$ is $B^0$). Then the MAD computation is reducible to querying $MEDIAN$, meaning that $\hat{MAD} = \frac{2\gamma^p}{1+\gamma}$ (as defined in Equation 2) is an $\epsilon$-accurate MAD [21]. If both $B_p$ and $B_q$ are $B^0$, we have $\hat{MAD} = 0$ calculated by Equation 2, which equals the exact MAD. Combining these cases, we define $\hat{MAD}$ as Equation 2.

*3.3.3 Worst-case Accuracy.* As discussed in Section 3.2, if $MAD < \frac{2\epsilon}{1-\epsilon}|MEDIAN|$, APPROX-MAD may find $B_p$ and $B_q$ as the same bucket in the same sketch ($\mathbb{S}^+$ or $\mathbb{S}^-$), which means the data is so concentrated that APPROX-MAD would output 0 directly. Otherwise, MAD would be computed over Equation 1 or Equation 2. In the worst case, we have the following proposition for accuracy.

PROPOSITION 3.6. *APPROX-MAD($\mathbb{D}, \epsilon$) can estimate a 1-accurate approximate MAD of $\mathbb{D}$.*
*(See proof in the full version technical report [1].)* ■

### 3.4 Complexity Analysis

*3.4.1 Time Complexity.* In the scanning stage, APPROX-MAD requires one-pass scan of data. In the query stage, it needs to sort the sketch to search for $B_p$ and $B_q$. Therefore, its time complexity is $O(n + m \log m)$, where $m$ is the maximum sketch size as introduced in Definition 3.

*3.4.2 Space Complexity.* Intuitively, the space cost of the sketch is $O(m)$. Note that $m$ is only related with value range instead of data size. In the worst case that $\mathbb{D} = \{\gamma^1, \gamma^2, \ldots, \gamma^n\}$, we have $m = n$. However, it has been proven sufficiently that for most distributions (even Pareto distributions that have exponentially fatter tails), $m$ is bounded by a relatively small constant for billions of elements [21]. We will further demonstrate the superiority of our solution in memory cost by experiments in Section 5.2.3.

## 4 SECOND SCAN

Using APPROX-MAD, we can get an $\epsilon'$-accurate MAD, where $\epsilon'$ may exceed the desired error bound $\epsilon$. As discussed in Section 2, if data is extremely concentrated, 0 would be the approximation. However, for most cases, we can utilize the knowledge from the first scan to fine-tune the parameter $\alpha_2$ of APPROX-MAD($\mathbb{D}, \alpha_2$) in the second scan, i.e., by narrowing it down, to get a more accurate approximate MAD with a bounded error $\epsilon$. To distinguish the notations of the two scans, we mark those in the first scan with the subscript 1 and those in the second scan with the subscript 2, e.g., the sketch in the first or the second scan is denoted as $\mathbb{S}_1$ or $\mathbb{S}_2$.

### 4.1 Parameter Determination

Now we introduce how to set $\alpha_2$ so that the error of $\hat{MAD}_2$ is bounded by $\epsilon$. Note that we need a second scan using APPROX-MAD only when the error bound of the first scan $\epsilon'$ exceeds $\epsilon$ and does not equal 1, i.e., $B_{1p_1}$ and $B_{1q_1}$ are from the same sketch ($\mathbb{S}_1^+$ or $\mathbb{S}_1^-$) with $p_1 \ne q_1$ according to Proposition 3.4. Our discussion is based on this prerequisite. With $\epsilon_1' = \frac{\gamma_1^{|p_1-q_1|}+1}{\gamma_1^{|p_1-q_1|}-1}\epsilon$ based on Proposition 3.4, we can set $\alpha_2$ by multiplying $\epsilon$ with a coefficient $\beta$, i.e., $\alpha_2 = \beta\epsilon$, so that $\epsilon_2' = \frac{\gamma_2^{|p_2-q_2|}+1}{\gamma_2^{|p_2-q_2|}-1}\beta\epsilon \le \epsilon$. This condition holds only if $\beta \le \frac{\gamma_2^{|p_2-q_2|}-1}{\gamma_2^{|p_2-q_2|}+1}$. We thereby define $\beta = \frac{\delta-1}{\delta+1}$ with

$$\delta = \begin{cases} \min(\gamma_1^{|p_1-q_1|-2} - \frac{1}{\gamma_1} + \frac{1}{\gamma_1^3}, \frac{1}{\gamma_1^3-\gamma_1^{|p_1-q_1|+1}+\gamma_1^2}) & \text{if } p_1 < q_1 \\ \min(\frac{1}{\gamma_1^2} + \frac{1}{\gamma_1^3} - \frac{1}{\gamma_1^{|p_1-q_1|+1}}, \frac{1}{\gamma_1^{2-|p_1-q_1|}+\gamma_1^3-\gamma_1}) & \text{if } p_1 > q_1 \end{cases}$$

which leads to the desired error bound $\epsilon$.

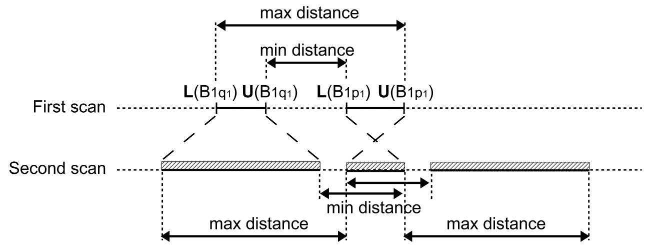

**Figure 5: The bucket pruning strategy for the second scan when $\mathbf{L}(B_{1p_1}) > \mathbf{L}(B_{1q_1})$. $B_{1p_1}$ and $B_{1q_1}$ are buckets in the first scan, where MEDIAN is covered by $B_{1p_1}$ and MAD is bounded by the minimum and the maximum distance between $B_{1p_1}$ and $B_{1q_1}$. According to Proposition 4.2, during the second scan, only buckets within the shadow range need maintenance, while other buckets can be pruned.**

Proposition 4.1. *If the determined coefficient $\beta$ has $0 < \beta < 1$, the second scan APPROX-MAD($\mathbb{D}, \beta\epsilon$) guarantees to return an $\epsilon$-accurate MAD.*
*(See proof in the full version technical report [1].)* ∎

Note that Proposition 4.1 is based on the condition that $0 < \beta < 1$, i.e., $\delta > 1$. Because $\delta$ is monotonically increasing with $|p_1 - q_1|$ and $|p_1 - q_1|$ is positively correlated with $\frac{MAD}{|MEDIAN|}$, the condition holds only if the data is not extremely concentrated. In such cases, the more concentrated the data is, the lower $\beta$ would be, leading a finer-grained sketch in the second scan, i.e., more buckets for holding all elements and thus higher costs of time and space. If the condition does not hold, our solution would estimate MAD as 0, which is 1-accurate according to Definition 2, as shown in Figure 3 in Section 2.2. Finally, together with Proposition 4.1 for $0 < \beta < 1$, the framework guarantees to return an $(\epsilon, 1)$-accurate MAD.

## 4.2 Bucket Pruning

With $0 < \beta < 1$, we have $\alpha_2 = \beta\epsilon < \alpha_1 = \epsilon$ and thus $\gamma_2 < \gamma_1$. It means more buckets would be allocated during the second scan to hold elements used in APPROX-MAD. To prevent the sketch size, i.e., the number of buckets, from growing substantially, we propose to prune unnecessary buckets. Intuitively, with $B_{1p_1}$ and $B_{1q_1}$ according to the first scan, we have determined the rough scope of MEDIAN and MAD, which implies the potential locations of $B_{2p_2}$ and $B_{2q_2}$. Therefore, we can prune unnecessary buckets that cannot be selected as $B_{2p_2}$ and $B_{2q_2}$ in the second scan. We enable the pruning by changing elements in unnecessary buckets to make them fall in their closest necessary buckets. In this way, their count is maintained while unnecessary buckets are omitted.

For $\mathbf{L}(B_{1p_1}) > \mathbf{L}(B_{1q_1})$ as shown in Figure 5, MEDIAN is bounded by $\mathbf{L}(B_{1p_1})$ and $\mathbf{U}(B_{1p_1})$, and MAD is bounded by $\mathbf{L}(B_{1p_1}) - \mathbf{U}(B_{1q_1})$ and $\mathbf{U}(B_{1p_1}) - \mathbf{L}(B_{1q_1})$. For elements $v$ with $v < \hat{v} = \mathbf{L}(B_{1p_1}) - (\mathbf{U}(B_{1p_1}) - \mathbf{L}(B_{1q_1})) < MEDIAN - MAD$, they can be regarded as $\hat{v}$ without influencing the searching process of Algorithm 1. Thereby, buckets covering the range $(-\infty, \hat{v}]$, i.e., buckets $B_{2i}$ with $\mathbf{U}(B_{2i}) \leq \hat{v}$, can be pruned. Following the same idea, we have the proposition below for necessary buckets in the second scan.

Proposition 4.2. *With $B_{1p_1}$ and $B_{1q_1}$ according to Lemma 3.3 from the first scan, a bucket $B_{2i}$ of the sketch in the second scan is necessary only if its lower- and upper- boundaries, $\mathbf{L}(B_{2i})$ and $\mathbf{U}(B_{2i})$, satisfy one of the following conditions.*

$$\begin{cases} \mathbf{L}(B_{2i}) < \mathbf{U}(B_{1p_1}) \\ \mathbf{U}(B_{2i}) > \mathbf{L}(B_{1p_1}) \end{cases}$$

$$\begin{cases} \mathbf{L}(B_{2i}) < 2\mathbf{U}(B_{1p_1}) - \mathbf{L}(B_{1q_1}) \\ \mathbf{U}(B_{2i}) > 2\mathbf{L}(B_{1p_1}) - \mathbf{U}(B_{1q_1}) \end{cases}$$

$$\begin{cases} \mathbf{L}(B_{2i}) < \mathbf{U}(B_{1p_1}) + \mathbf{U}(B_{1q_1}) - \mathbf{L}(B_{1p_1}) \\ \mathbf{U}(B_{2i}) > \mathbf{L}(B_{1p_1}) + \mathbf{L}(B_{1q_1}) - \mathbf{U}(B_{1p_1}) \end{cases}$$

∎

After pruning, we require far fewer buckets to hold elements and also avoid searching unnecessary buckets.

*Example 4.3.* Following Example 3.2, we use $\alpha_2 = \beta\epsilon \approx 0.0014$ as the parameter of APPROX-MAD in the second scan. Without the bucket pruning strategy, we get a new sketch with 6 buckets as $\mathbb{S} = \{B^+_0(1), B^+_{399}(2), B^+_{584}(2), B^+_{650}(1), B^+_{797}(2), B^+_{836}(1)\}$. After pruning, the sketch shrinks to 5 buckets as $\{B^+_{380}(1), B^+_{399}(2), B^+_{584}(2), B^+_{701}(1), B^+_{715}(3)\}$. Then Algorithm 1 returns $B^+_{584}$ (covering 5) and $B^+_{399}$ (covering 3) as $B_{2p_2}$ and $B_{2q_2}$. Finally, we get $\hat{MAD}_2 = 1.9965$ with its error bound $0.0055 < 0.01$. ∎

## 5 EVALUATION

We implement the algorithms in Java. It includes two versions, the native version and the Spark version, which uses the template of the user-defined aggregation function of Apache Spark 3.0. The methods have also been implemented in HUAWEI Cloud.

## 5.1 Experimental Settings

Experiments run on a machine with 2.4GHz CPU & 8GB memory.

*5.1.1 Baseline.* To distinguish the performance of our proposed solutions with one-pass scan or two-pass scan, we refer to them as OP-MAD and TP-MAD, respectively. As there are no existing approximate MAD algorithms to our knowledge, we compare our solution against the Java implementation of EXACT-MAD, the Java implementation of the method, called GK-MAD, that invokes GKArray (an adaptive version of GK Algorithm [15]) twice, and the Java implementation of the method, called DD-MAD, that invokes DDSketch [21] twice. GK Algorithm and DDSketch are two representative algorithms for approximate quantile computation and can therefore be used to calculate approximate MAD. We compare them in accuracy, time cost and memory cost, implemented in both versions.

*5.1.2 Parameter.* For our solution, the parameter $\epsilon$ representing the desired relative error bound is set according to the data concentration degree. Since desired rank-error (or relative error) bound for GK-MAD (or DD-MAD) is unknown, we use the parameter as defined for approximate quantiles that gives roughly similar memory footprints, also denoted as $\epsilon$. The parameter $m$ is a constant representing the limit on the sketch size for our solution and DD-MAD. These parameters are summarized in Table 2.

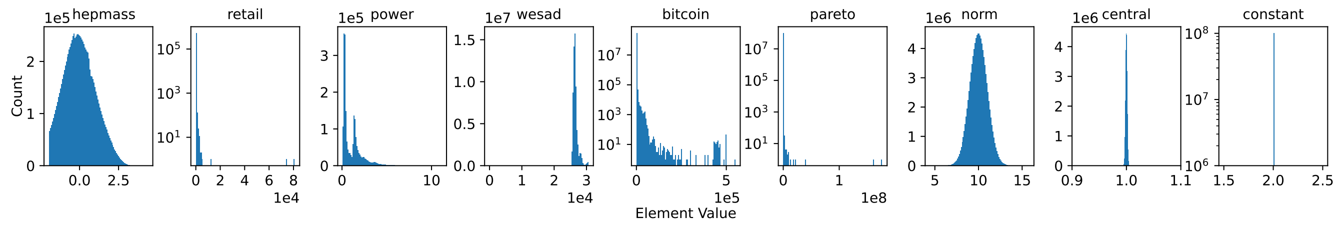

**Figure 6: Histograms of datasets. The y-axes of *retail, bitcoin* and *pareto* are in log scale due to heavy-tailed data distributions.**

5.1.3 *Dataset.* We use 9 datasets in our comparison experiments, whose distributions are shown in Figure 6. Five of them are real datasets. The *hepmass* dataset and the *retail* dataset, both used in [14], consist of the first feature of the UCI [12] HEPMASS dataset and integer purchase quantities from the UCI Online Retail dataset, respectively. The *power* dataset, used in [14, 21], consists of Global Active Power measurements from the UCI Individual Household Electric Power Consumption dataset. The *wesad* dataset is the converted temperature data collected from RespiBAN Professional (a wearable respiration monitoring device) in the UCI WESAD dataset. The *bitcoin* dataset is a Kaggle public dataset, consisting of the bitcoin transaction amount from 2009 to 2018 [11]. It has 300 M elements (71.4GB in total), which cannot be kept in main memory.

The last four datasets are synthetic, all with data size $10^8$ (100M elements). The *pareto* dataset consists of elements generated from a Pareto distribution, a typical long-tailed distribution, with the scale parameter and the shape parameter both as 1. It is used in [21] as the worst case for DDSketch. The *norm* and *central* datasets consist of elements generated from two Normal distributions $\mathcal{N}(10, 1)$ and $\mathcal{N}(1, 0.0015)$, respectively. In particular, the *central* dataset is highly concentrated with $\frac{MAD}{|MEDIAN|} = 0.001$. The last *constant* dataset consists of elements with the same (constant) value 2, having $MAD = 0$. The last two datasets *central* and *constant* are non-trivial to our solution, because they are highly concentrated, i.e., having $\frac{MAD}{|MEDIAN|} \approx 0$. As discussed in Section 4.1, for such datasets, our solution needs to use smaller $\epsilon$, leading to a finer-grained sketch in Algorithm 2 and thus higher costs of time and space.

## 5.2 Comparison to Baselines

By the native version, we compare our proposals, OP-MAD and TP-MAD, to the baselines, GK-MAD and DD-MAD using the existing GK and DD algorithm for the approximate median. The performances are summarized in Table 2, where the units of time and memory are second and kB. The error of approximate MAD relative to exact MAD by EXACT-MAD follows Definition 2, i.e., $\frac{|\hat{MAD}-MAD|}{MAD}$. Figure 7 shows scalability in terms of accuracy, time cost and memory cost.

5.2.1 *Accuracy of Approximation.* TP-MAD guarantees a relative error bound on its returned MAD, while other algorithms provide no guarantees. GK-MAD (DD-MAD) provides a rank (relative) error of at most $\epsilon$ in quantile estimates for each scan. But these errors do not provide straightforward error bounds for the MAD results.

As shown in Table 2, TP-MAD provides the smallest relative-error results for each dataset. Its highest relative error is 0.0024. That is, the returned error of TP-MAD in practice is significantly smaller than the theoretical bound $\epsilon$.

The errors of OP-MAD are a bit higher, as illustrated in Table 2. It is not surprising, since OP-MAD has only one pass through the data while other methods such as TP-MAD can conduct a second scan for more accurate results.

According to Table 2, for *bitcoin*, OP-MAD, TP-MAD and DD-MAD perform much better than GK-MAD in accuracy. It is because *bitcoin* is a sparse dataset with the long-tail distribution as illustrated in Figure 6. The baseline GK-MAD, based on the rank error, is not effective in dealing with sparse datasets.

For the highly concentrated dataset *central*, having $\frac{MAD}{|MEDIAN|} = 0.001$, TP-MAD is still applicable with error bound guarantees, as shown in Table 2. For the extreme case *constant*, TP-MAD returns 0 directly according to Figure 3, which is exactly the ground truth.

5.2.2 *Time Cost.* We experiment on each dataset of each scale for ten times and use the average run time as results. As Table 2 shows, using a machine with a 2.4GHz CPU and 8GB memory, it takes less than 10 seconds for TP-MAD to give an approximate MAD over *pareto* and *norm* with $10^8$ elements (1.7GB). Compared with EXACT-MAD, it is 18× faster.

In Figure 7(b), the average run time of TP-MAD increases linearly with data size, demonstrating its linear time cost $O(n)$. Besides, TP-MAD and DD-MAD are very close in terms of time performance because of their same sketch structure.

The time costs of OP-MAD are about half of the corresponding TP-MAD time costs, in all the datasets except *constant*. The result is not surprising, given that OP-MAD needs only one-pass of data as illustrated in branch (a) in Figure 3, while TP-MAD in branch (b) scans the data twice. TP-MAD and OP-MAD have comparable time cost on dataset *constant*, since TP-MAD directly returns 0 in this constant dataset without a second scan, i.e., branch (c) in Figure 3.

For *bitcoin*, our proposals, especially OP-MAD, outperform the other approximate algorithms in time cost, because OP-MAD requires only one-pass scan. Moreover, our TP-MAD is optimized with the bucket pruning strategy proposed in Section 4.2.

Since the dataset *central* is highly concentrated, TP-MAD needs to use much smaller buckets in the second scan. It increases the sketch size and leads to longer run time than it does for *norm* with $\frac{MAD}{|MEDIAN|} = 0.068$, as illustrated in Table 2. For the extreme case of having the same value in the dataset *constant*, EXACT-MAD shows comparable time cost. The reason is that the same-value elements are naturally in order for exact computation, while other methods need extra time to allocate sketches.

**Table 2: Comparison to different baselines over various datasets (OOM stands for out-of-memory)**

| | *hepmass* ($\epsilon$=0.01, $m$=1024, $n$=10.5M) | | | | *retail* ($\epsilon$=0.01, $m$=1024, $n$=530k) | | | | *power* ($\epsilon$=0.01, $m$=1024, $n$=2M) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm | Result | Error | Time | Memory | Result | Error | Time | Memory | Result | Error | Time | Memory |
| EXACT-MAD | 0.7080 | / | 10.11 | 147000.12 | 2.0000 | / | 0.12 | 7438.13 | 0.3960 | / | 0.61 | 28690.04 |
| TP-MAD (our) | 0.7069 | **0.0016** | 2.27 | 23.97 | 1.9952 | **0.0024** | 0.07 | **4.86** | 0.3961 | **0.0004** | 0.27 | 6.37 |
| OP-MAD (our) | 0.6954 | 0.0179 | **1.07** | 23.97 | 1.9836 | 0.0082 | **0.03** | **4.86** | 0.3889 | 0.0178 | **0.13** | 6.37 |
| GK-MAD | 0.7210 | 0.0183 | 3.19 | **3.20** | 3.0000 | 0.5000 | 0.17 | 19.31 | 0.3900 | 0.0152 | 0.41 | **3.41** |
| DD-MAD | 0.7046 | 0.0048 | 1.53 | 23.97 | 1.9937 | 0.0032 | 0.07 | 4.94 | 0.4106 | 0.0037 | 0.26 | 8.05 |
| | *wesad* ($\epsilon$=0.001, $m$=1024, $n$=63M) | | | | *bitcoin* ($\epsilon$=0.01, $m$=1024, $n$=300M) | | | | *pareto* ($\epsilon$=0.01, $m$=2048, $n$=100M) | | | |
| Algorithm | Result | Error | Time | Memory | Result | Error | Time | Memory | Result | Error | Time | Memory |
| EXACT-MAD | 329.00 | / | 47.99 | 881097.10 | OOM | OOM | OOM | OOM | 0.8279 | / | 175.63 | 1400000.12 |
| TP-MAD (our) | 329.07 | **0.0002** | 9.49 | 5.09 | 0.2347 | **0.0007** | 40.96 | 31.13 | 0.8282 | **0.0003** | 9.72 | 8.71 |
| OP-MAD (our) | 310.44 | 0.0564 | **5.41** | **2.76** | 0.2329 | 0.0084 | **21.98** | 31.13 | 0.8308 | 0.0035 | **4.62** | 8.71 |
| GK-MAD | 331.00 | 0.0061 | 22.38 | 57.74 | 0.2415 | 0.0284 | 67.79 | **3.44** | 0.8451 | 0.0208 | 20.73 | **3.26** |
| DD-MAD | 330.63 | 0.0050 | 7.94 | 33.78 | 0.2345 | 0.0012 | 42.20 | 31.13 | 0.8780 | 0.0606 | 9.45 | 18.23 |
| | *norm* ($\epsilon$=0.003, $m$=1024, $n$=100M) | | | | *central* ($\epsilon$=0.0001, $m$=71680, $n$=100M) | | | | *constant* ($\epsilon$=0.01, $m$=1024, $n$=100M) | | | |
| Algorithm | Result | Error | Time | Memory | Result | Error | Time | Memory | Result | Error | Time | Memory |
| EXACT-MAD | 0.6742 | / | 173.52 | 1400000.12 | 0.0010 | / | 175.93 | 1400000.12 | 0 | / | **5.42** | 1400000.12 |
| TP-MAD (our) | 0.6743 | **0.0002** | 9.16 | 8.62 | 0.0010 | **0.0000** | 19.83 | 1678.33 | 0 | / | 5.67 | **0.486** |
| OP-MAD (our) | 0.6756 | 0.0012 | **4.41** | **5.14** | 0.0010 | 0.0510 | **8.45** | **2.40** | 0 | / | 5.61 | **0.486** |
| GK-MAD | 0.6750 | 0.0021 | 21.71 | 10.57 | 0.0010 | 0.0001 | 41.56 | 293.69 | 0 | / | 9.40 | 3.23 |
| DD-MAD | 0.6791 | 0.0012 | 9.18 | 25.94 | 0.0010 | 0.0021 | 18.63 | 942.30 | 0.0063 | +∞ | 11.13 | **0.486** |

5.2.3 *Memory Cost.* For approximate MAD algorithms, the memory cost is mainly on storing the sketch. It is obtained by serializing each sketch to a byte array, whose length is the sketch size in byte. For algorithms with two-pass, we report the larger sketch size.

In Figure 7(c), the memory cost of OP-MAD and TP-MAD keeps almost constant for most datasets. The reason is that the sketch size of TP-MAD is related to value range instead of data size. For Pareto distributions, while the data size increases, the sampling probability of larger values goes up, expanding the value range. Hence the sketch size of OP-MAD and TP-MAD for *pareto* increases.

In all cases, the memory cost of OP-MAD is less than or equal to that of TP-MAD, because TP-MAD uses a finer-grained sketch during the second scan. It may lead to a larger number of buckets even with the bucket pruning strategy. Especially in dataset *central*, the data is highly concentrated (with $\frac{MAD}{|MEDIAN|} = 0.001$) that TP-MAD in the second scan needs to decrease the buckets' granularity substantially, making its memory cost much higher than the first scan, i.e., the cost of OP-MAD.

For *bitcoin* with a long-tail distribution, with the increase of data size, the value range expands and enlarges the sketch sizes of OP-MAD, TP-MAD and DD-MAD. Note that EXACT-MAD is not applicable to the dataset owing to out-of-memory (OOM), showing again the necessity of approximate MAD over large scale data.

For *central*, TP-MAD requires larger space than in dataset *norm* as aforesaid, but it is still significantly more efficient in memory than EXACT-MAD as illustrated in Table 2. For *constant*, TP-MAD occupies the same memory as OP-MAD does, because it does not have a second scan but outputs 0 directly. They are 2800000× less than EXACT-MAD in memory cost.

## 5.3 Evaluation of Proposed Techniques

5.3.1 *Varying the Error Bound $\epsilon$.* The parameter $\epsilon$ specifies the trade-off between effectiveness and efficiency. Besides, with smaller $\epsilon$, TP-MAD is applicable to estimate MAD for more concentrated datasets. To illustrate these points, we vary $\epsilon$ and apply TP-MAD to each entire dataset, as shown in Figure 8.

Sharp upward/downward trends are observed in Figure 8. It is because when the data is too concentrated, i.e., $\frac{MAD}{|MEDIAN|}$ is too small, TP-MAD returns $\langle 0, 1 \rangle$ directly. For other cases, with $\epsilon$ decreasing, i.e., to get a more accurate result, memory cost increases while the rise of time cost is gentle. Thereby we can set $\epsilon$ to an appropriate value if the dataset is very concentrated without much extra time cost. In practice, to set desired error bounds while keeping memory cost as low as possible, we can take a sample as the input for TP-MAD to help find an $\epsilon$ that make TP-MAD follow branch (b) in Figure 3, e.g., setting $\epsilon = 0.01$ for *pareto*.

5.3.2 *Evaluating the Bucket Pruning Strategy.* Figure 9 shows the sketch size of TP-MAD in the second scan with or without bucket

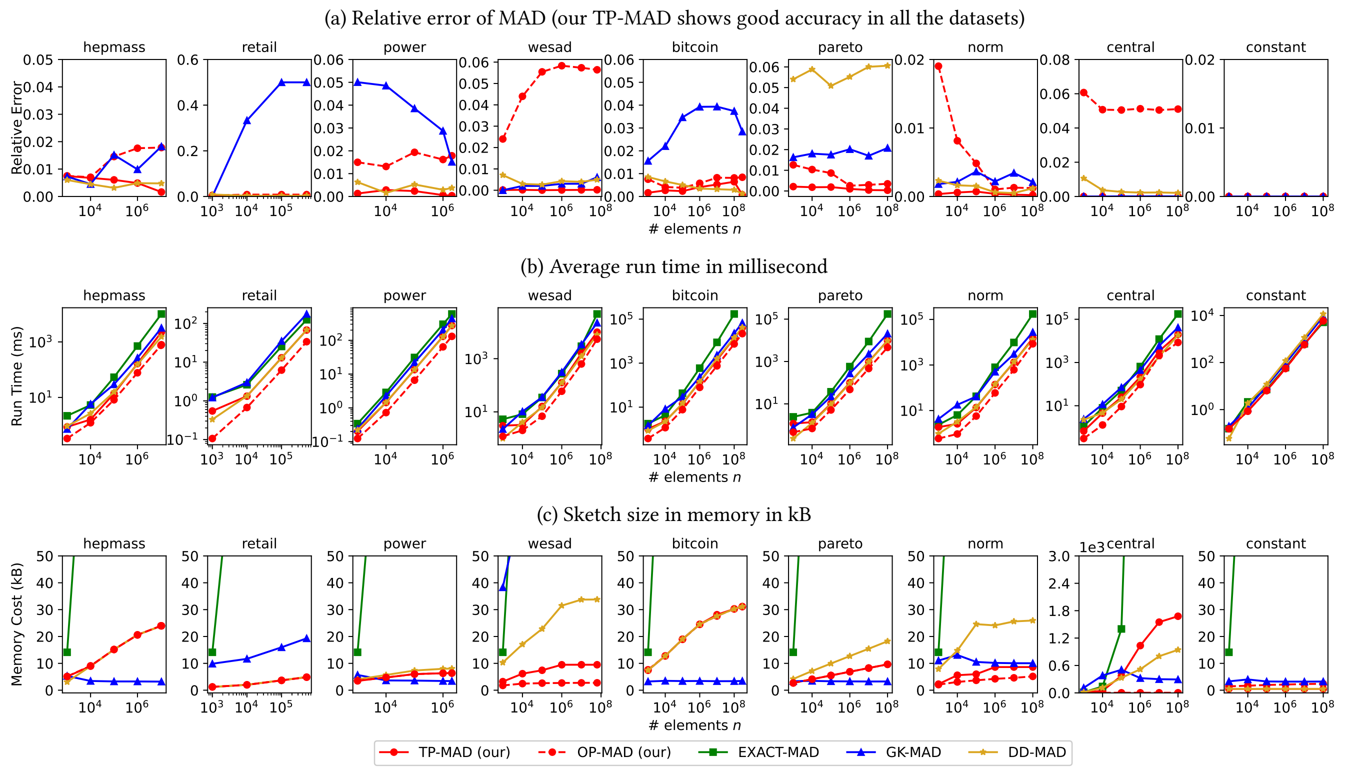

**Figure 7: Scalability in accuracy, time and memory costs over various datasets**

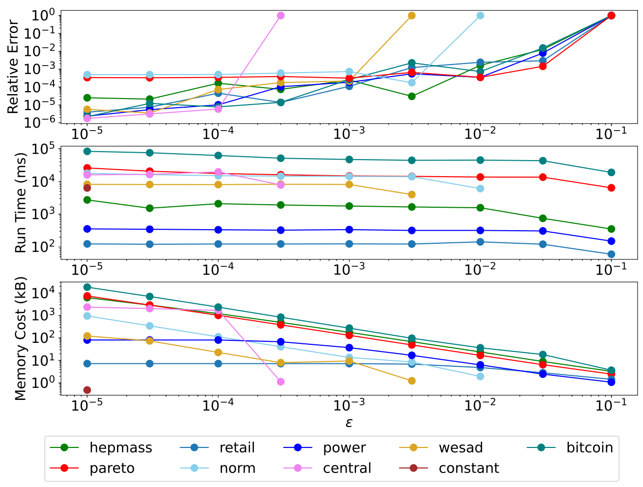

**Figure 8: Varying the desired error bound $\epsilon$ in TP-MAD**

pruning. When data size increases, the memory cost of the algorithm without bucket pruning increases owing to the wider value range. Note that the range is not the difference between the maximum and minimum elements, but the region where elements exist. Therefore, for *norm* whose extreme elements are not likely to vary too much, its value range still expands and thus the memory cost of TP-MAD increases. After pruning, the sketch can ignore unnecessary range, which helps TP-MAD keep the sketch size at a relatively low level in the second scan.

#### 5.3.3 *Evaluating the Distributed Implementation.*

To evaluate our proposals on large distributed datasets, we compare TP-MAD implemented in Apache Spark and OP-MAD implemented in Structured Streaming [8] (a stream processing engine built on Spark) with GK-MAD using *approxQuantile*, the Spark built-in GK algorithm. Spark provides three interfaces in the template of the user-defined aggregation function, *reduce*, *merge* and *finish*, corresponding to INSERT, MERGE and Lines 13-24 of APPROX-MAD in our solution, respectively. MERGE is the process to merge two sketches, realized by combining buckets with the same index from the two sketches and summing up their counts. The fully mergeable OP-MAD is implemented a similar way in Structured Streaming.

Figure 10 shows the average run time of algorithms on two datasets. Under the same settings, TP-MAD outperforms GK-MAD in time performance. It is 14.66% and 9.35% faster when using 8 cores for $10^9$ elements from the Pareto distribution and the Normal distribution respectively. It is not surprising that the mergeable OP-MAD with Structured Streaming implementation is more efficient than TP-MAD having two scans.

## 5.4 Application Study

In addition to outlier detection in Examples 1.1 and 1.2, we present another application in out-of-control monitoring [2].

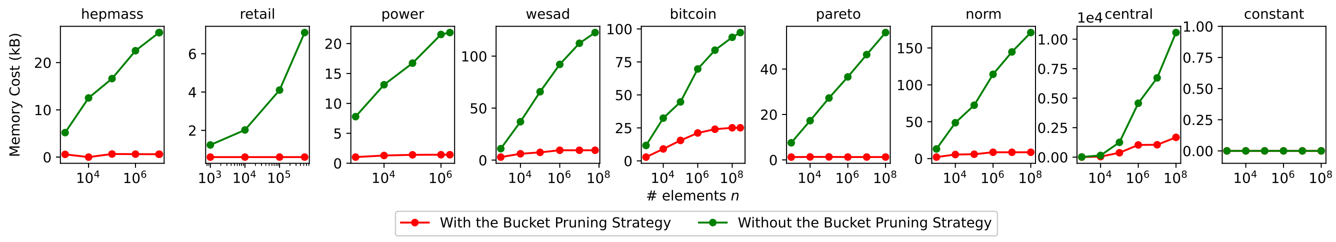

**Figure 9: Memory cost in kB of the second scan in TP-MAD with or without the bucket pruning strategy in Section 4.2**

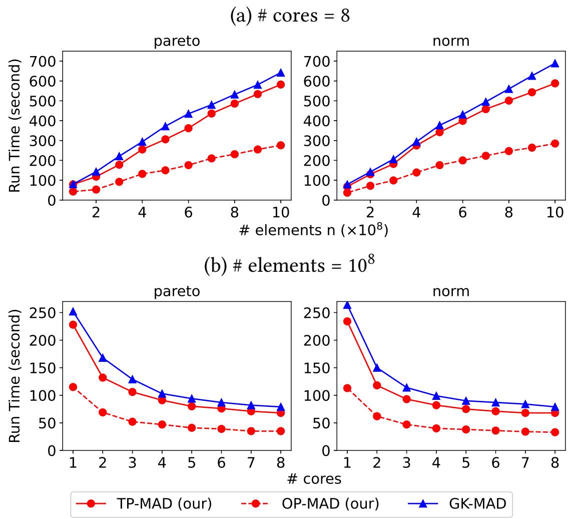

**Figure 10: Time performance of Apache Spark/Structured Streaming implementation by varying (a) the number of elements in datasets and (b) the number of cores in the cluster**

Analogous to Figure 1 of comparing SD and MAD for outlier detection, Figure 11 presents the Shewhart S-control charts for determining whether a manufacturing or business process is out of control, using SD and MAD, respectively. Specifically, consider a data stream of sensor readings, monitoring the manufacturing process. The central line (CL), lower control limit (LCL) and upper control limit (UCL) are computed from historical data, using either SD or MAD. When new data come, SD or MAD is computed over the samples of the new data. If the SD or MAD of the samples of the newly coming data exceeds the limits, the manufacturing process is identified as out-of-control. The less the data is sampled, the better the performance is. Figure 11(a) and (b) show that SD needs to sample 37 times to identify out-of-control (i.e., SD exceeding LCL), while MAD needs only 3 samples, i.e., more effective.

Similar to Figure 2(a), comparing approximate MAD and exact MAD in the outlier detection application, Figure 11(c) evaluates the sample number needed to identifying the out-of-control process by exact MAD and its approximation. Almost the same sample numbers illustrate again that using approximate MAD is sufficient in practice. The corresponding time cost for computing approximate MAD is much lower than that of exact MAD in Figure 11(d).

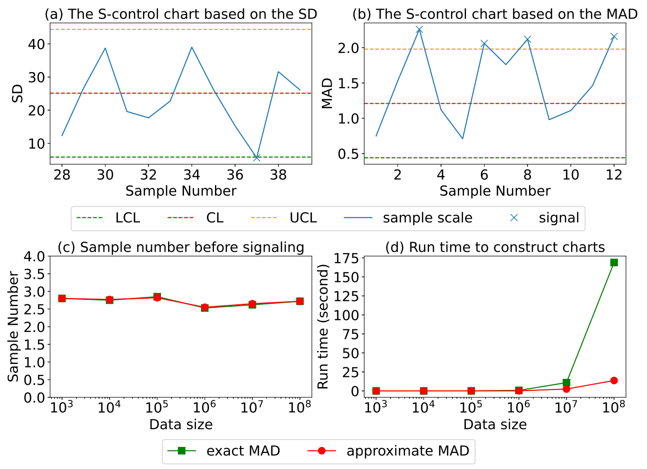

**Figure 11: Application of MAD in the Shewhart S-control chart to identify out-of-control manufacturing process. (a) SD needs to sample 37 times to identify out-of-control, i.e., SD exceeding LCL, while (b) MAD needs only 3 samples, finding MAD exceeding UCL, i.e., more effective. Approximate MAD has almost no difference to exact MAD in the effectiveness of identifying out-of-control states in (c), but with much lower cost for computing the MAD in (d).**

## 6 RELATED WORK

While MEDIAN is found to be more robust and useful than MEAN in identifying outliers [23], MAD also works better than SD with outlier detection and the Cauchy distribution [16].

### 6.1 Adaption of Approximate Median

While sketches have been widely used for approximating quantiles and median (see Section C in the full version technical report [1] for an introduction), the methods for approximating median can be adapted for MAD, by calling them twice, e.g., the baseline DD-MAD in Section 5.1.1 calling DDSketch [21] twice. However, this direct adaption of MEDIAN approximation to MAD is not effective (unknown theoretical error bounds and poor practice performance). The novelty of our proposal compared to [21] and its two-pass adaption for MAD approximation is as follows.

(1) New problem. To the best of our knowledge, this is the first work on approximating MAD computation, while [21] addresses only quantile/median computation. Estimating quantiles is to find

the elements in the desired rank in the sorted set, a problem of ranking elements. In contrast, approximating MAD is to rank the distances between the elements in the set and the median of the set, a problem of ranking distances on elements.

(2) Finding buckets. To find the bucket $B_p$ holding MEDIAN, [21] searches the sketch in the order of buckets' boundaries. In contrast, MAD is the distance between two elements, i.e., $e_x$ and $MEDIAN$ with $|e_x - MEDIAN| = MAD$. To find the bucket $B_q$ holding $e_x$, our Algorithm 1 searches the sketch in the order of distances to the boundaries of $B_p$ (the bucket holding MEDIAN). We prove in Lemma 3.3 that MAD is bounded by the minimum and maximum distances between the found buckets $B_p$ and $B_q$.

(3) One pass of data. The baseline DD-MAD needs to call DDSketch [21] twice to compute MAD, i.e., two passes of the data. In contrast, since our Algorithm 1 successfully finds the bucket $B_q$ holding $e_x$ as aforesaid, Algorithm 2 in Section 3.2 can directly output an approximate MAD together with an error bound computed referring to Lemma 3.3, i.e., one-pass solution.

(4) Error bound. The baseline DD-MAD that calls DDSketch [21] twice has unknown error bounds for the MAD result. The reason is that the approximate MEDIAN from the first scan would bias the elements for the second scan, i.e., the deviations from MEDIAN, leading to an incorrect bucket for MAD. In our solution, Section 4.1 advances the method by calling Algorithm 2 twice and narrowing down the size of buckets so that a MAD result with desired error bound is output according to Proposition 4.1.

(5) Pruning bucket. The baseline DD-MAD calling DDSketch [21] twice needs to maintain all the buckets in the sketch that hold all elements in both scans. In contrast, we prove in Proposition 4.2 that the second scan in our solution only needs to maintain the necessary buckets related to MEDIAN and MAD. The bucket pruning strategy in Section 4.2 thus leads to lower space cost. Experiments in Section 5.2.3 demonstrate that our solution outperforms the baseline DD-MAD in memory cost.

## 6.2 Mergeability and Parallelization

Many approximate quantile/median algorithms, e.g., GK Algorithm [15] and DDSketch [21], are mergeable [5], i.e., their sketches can be merged with each other while preserving the error guarantees. While our one-pass Algorithm 2 OP-MAD is also fully mergeable, the two-pass method TP-MAD, calling Algorithm 2 twice, is not.

(1) OP-MAD, i.e., Algorithm 2 returning an approximate MAD together with an error bound $\epsilon'$ computed, is fully mergeable. It requires only one-pass scan just like other mergeable algorithms, e.g., HLL for the count-distinct problem [13]. As discussed in Section 3.2, the sketch of Algorithm 2 can be pre-computed on historical data and updated when new data arrives (as described in Lines 6-12 of Algorithm 2), e.g., in Apache Druid. The subsequent online query is executed as Lines 13-24, based on the pre-computed sketch. The query supports basic filter operations, e.g., by skipping buckets whose boundaries are excluded from filter conditions while searching buckets for MAD, i.e., in Line 1 of Algorithm 1. The fully mergeable OP-MAD is implemented and evaluated in Structured Streaming in Section 5.3.3.

(2) TP-MAD, which guarantees to return an $(\epsilon, 1)$-accurate MAD by two-pass scan as presented in Section 4, is parallelizable. The reason is that an additional pass may still be required if the merged precomputed results are not accurate enough by OP-MAD. Fortunately, it can be processed in parallel over distributed batch data, e.g., in HDFS. We can compute a sketch for each data partition in parallel and merge them afterwards, since the sketches are mergeable as discussed in Section 3.1. An implementation of TP-MAD in parallel is presented and evaluated in Apache Spark in Section 5.3.3.

## 6.3 Continuous and Discrete Values

In addition to the continuous median, the discrete median is also studied [24]. While all the examples illustrated in this study such as Example 3.1 are about continuous medians, the proposed approach could also be adapted to the discrete median.

For the median of discrete values [24], it consists of a value set $\mathbb{D} = \{1, 2, 3, 4\}$ and a frequency set $\mathbb{F} = \{2, 2, 3, 2\}$. Each element in $\mathbb{F}$ represents the frequency of the corresponding value in $\mathbb{D}$. To compute its discrete median, one can convert the discrete values to continuous values by repeating each value based on its frequency, $\mathbb{D}' = \{1, 1, 2, 2, 3, 3, 3, 4, 4\}$, from which the median 3 is computed. Likewise, with the converted $\mathbb{D}'$ we also compute MAD=1.

## 7 CONCLUSION

In this paper, we propose to find approximate MAD with bounded error. The computation of MAD is costly and relies on finding the median, i.e., the median of the absolute deviations from the dataset's median. An exact MAD algorithm by calling the median algorithm twice needs $O(n)$ space cost over a set of $n$ elements. Directly applying the existing approximate median algorithms for MAD is with unknown error bounds in theory and ineffective in practice (as illustrated in the experiments in Section 5.2.1). In contrast, our solution guarantees to return an $(\epsilon, 1)$-accurate MAD, with space complexity $O(m)$ and time complexity $O(n + m \log m)$, where $m$ is the size of the sketch used to compress data. The bucket pruning strategy further reduces the space cost of our proposal. Remarkably, the proposed OP-MAD is fully mergeable and TP-MAD is parallelizable, naturally making them support implementations in a streaming environment, e.g., in Structured Streaming, and a distributed environment, e.g., in Apache Spark, respectively. The extensive experiments over various datasets demonstrate the superiority of our solution, e.g., 160000× less memory and 18× faster than the aforesaid exact method in datasets *pareto* and *norm*. The improvement is more significant over the large datasets that are not highly concentrated. The reason is that the time and space costs of the exact algorithm increase heavily with data sizes as discussed in Section 2.1, while the costs of approximation algorithms rely more on the sketch sizes as analyzed in Section 3.4. If the data is not highly concentrated, the approximation algorithms can use a more coarse-grained sketch for the second scan, i.e., decreasing sketch size and thus the costs of time and space.

## ACKNOWLEDGMENTS

This work is supported in part by the National Key Research and Development Plan (2019YFB1705301, 2019YFB1707001), the National Natural Science Foundation of China (62072265, 62021002, 71690231), and the MIIT High Quality Development Program 2020. Shaoxu Song is the corresponding author.

## REFERENCES

[1] [n.d.]. Full version technical report. *https://sxsong.github.io/doc/mad.pdf* ([n. d.]).

[2] Moustafa OA Abu-Shawiesh. 2008. A simple robust control chart based on MAD. *Journal of Mathematics and Statistics* 4, 2 (2008), 102.

[3] MO Abu—Shawiesh, FM Al—Athari, and HF Kittani. 2009. Confidence Interval for the Mean of a Contaminated Normal Distribution. *Journal of Applied Sciences* 9, 15 (2009), 2835–2840.

[4] KS Adekeye and PI Azubuike. 2012. Derivation of the limits for control chart using the median absolute deviation for monitoring non-normal process. *Journal of Mathematics and Statistics* 8, 1 (2012), 37–41.

[5] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. 2013. Mergeable summaries. *ACM Trans. Database Syst.* 38, 4 (2013), 26:1–26:28. https://doi.org/10.1145/2500128

[6] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. 2017. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* 262 (2017), 134–147. https://doi.org/10.1016/j.neucom.2017.04.070

[7] Faris M Al-Athari. 2011. Confidence interval for locations of non-kurtosis and large kurtosis leptokurtic symmetric distributions. J. *Applied Sci* 11 (2011), 528–534.

[8] Michael Armbrust, Tathagata Das, Joseph Torres, Burak Yavuz, Shixiong Zhu, Reynold Xin, Ali Ghodsi, Ion Stoica, and Matei Zaharia. 2018. Structured Streaming: A Declarative API for Real-Time Applications in Apache Spark. 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, 601–613. https://doi.org/10.1145/3183713.3190664

[9] 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

[10] Namjin Chung, Xiaohua Douglas Zhang, Anthony Kreamer, Louis Locco, Pei-Fen Kuan, Steven Bartz, Peter S. Linsley, Marc Ferrer, and Berta Strulovici. 2008. Median Absolute Deviation to Improve Hit Selection for Genome-Scale RNAi Screens. *Journal of Biomolecular Screening* 13, 2 (2008), 149–158. https://doi.org/10.1177/1087057107312035 arXiv:https://doi.org/10.1177/1087057107312035 PMID: 18216396.

[11] Kaggle Public Dataset. 2020. Bitcoin data from Jan 2009 to Feb 2018. https://www.kaggle.com/shiheyingzhe/bitcoin-transaction-data-from-2009-to-2018. Accessed: 2021-04-23.

[12] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[13] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*. Discrete Mathematics and Theoretical Computer Science, 137–156.

[14] Edward Gan, Jialin Ding, Kai Sheng Tai, Vatsal Sharan, and Peter Bailis. 2018. Moment-Based Quantile Sketches for Efficient High Cardinality Aggregation Queries. *Proc. VLDB Endow.* 11, 11 (2018), 1647–1660. https://doi.org/10.14778/3236187.3236212

[15] Michael Greenwald and Sanjeev Khanna. 2001. Space-Efficient Online Computation of Quantile Summaries. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001*, Sharad Mehrotra and Timos K. Sellis (Eds.). ACM, 58–66. https://doi.org/10.1145/375663.375670

[16] David C. Howell. 2005. *Median Absolute Deviation.* American Cancer Society. https://doi.org/10.1002/0470013192.bsa384 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470013192.bsa384

[17] H. H. Khalil, R. O. K. Rahmat, and W. A. Mahmoud. 2008. Chapter 15: Estimation of Noise in Gray-Scale and Colored Images Using Median Absolute Deviation (MAD). In *2008 3rd International Conference on Geometric Modeling and Imaging*. 92–97. https://doi.org/10.1109/GMAI.2008.7

[18] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. 2013. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology* 49, 4 (2013), 764 – 766. https://doi.org/10.1016/j.jesp.2013.03.013

[19] Edmund R Malinowski. 2009. Determination of rank by median absolute deviation (DRMAD): a simple method for determining the number of principal factors responsible for a data matrix. *Journal of Chemometrics: A Journal of the Chemometrics Society* 23, 1 (2009), 1–6.

[20] Aris Marjuni, Teguh Bharata Adji, and Ridi Ferdiana. 2019. Unsupervised software defect prediction using median absolute deviation threshold based spectral classifier on signed Laplacian matrix. *J. Big Data* 6 (2019), 87. https://doi.org/10.1186/s40537-019-0250-z

[21] 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

[22] Vadim A. Raikhlin and Roman K. Klassen. 2020. Clusterix-Like BigData DBMS. *Data Sci. Eng.* 5, 1 (2020), 80–93. https://doi.org/10.1007/s41019-020-00116-2

[23] Shaoxu Song, Aoqian Zhang, Jianmin Wang, and Philip S. Yu. 2015. SCREEN: Stream Data Cleaning under Speed Constraints. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. 827–841. https://doi.org/10.1145/2723372.2723730

[24] Engineering ToolBox. 2012. Discrete Data Sets - Mean, Median and Mode Values. https://www.engineeringtoolbox.com/mean-median-modal-d_1846.html. Accessed: 2021-05-11.

[25] Chunjie Wu, Yi Zhao, and Zhaojun Wang. 2002. The median absolute deviations and their applications to Shewhart control charts. *Communications in Statistics-Simulation and Computation* 31, 3 (2002), 425–442.