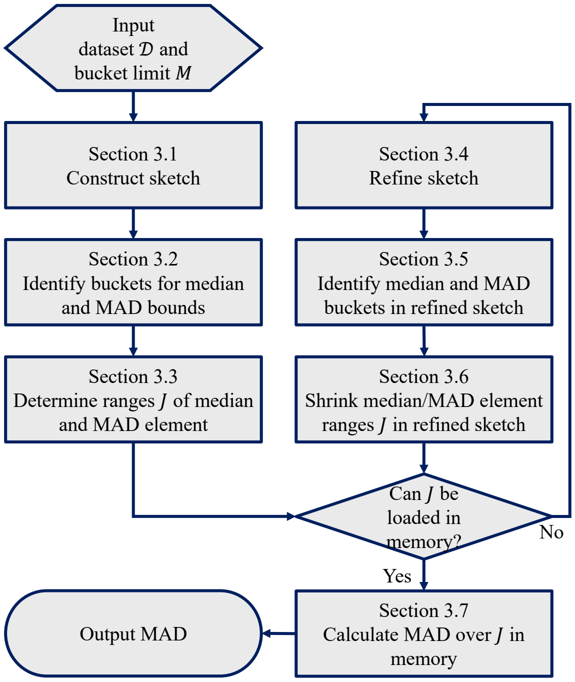
Input
dataset $\mathcal{D}$ and
bucket limit M
Section 3.1
Construct sketch
Section 3.2
Identify buckets for median and MAD bounds
Section 3.3
Determine ranges J of median and MAD element
Section 3.4
Refine sketch
Section 3.5
Identify median and MAD buckets in refined sketch
Section 3.6
Shrink median/MAD element ranges J in refined sketch
Can J be loaded in memory?
No
Yes
Section 3.7
Calculate MAD over J in memory
Output MAD

Figure 21: An overview of exact MAD computation process

Table 2: Notations

Symbol	Description
$\mathcal{D}$	A dataset composed of elements e_i
MAD	Median absolute deviation
λ	Median of $\mathcal{D}$
η	MAD element in $\mathcal{D}$ having $\lvert\lambda - \eta\rvert = MAD$
$\mathcal{Z}$	Bucket boundary domain
$\mathbb{S}(\zeta)$	CORE-Sketch with base ζ
N	The data size of dataset $\mathcal{D}$
M	Maximum number of buckets
B_i	Bucket with index i in CORE-Sketch
B_m	Median bucket
B_l, B_r	MAD buckets on the left and right side of B_m

A TECHNIQUE DETAILS

Table 2 lists the notations frequently used in this paper.

A.1 Computation Process

To find the exact MAD element η, the computation process is divided into three stages, sketch construction, sketch refinement and MAD calculation. Intuitively, sketch construction stage gives an initial CORE-Sketch of the dataset $\mathcal{D}$, sketch refinement stage then gradually narrows down the ranges of data that contain the median and MAD elements, and finally, MAD calculation stage returns an accurate MAD result according to the refined sketch.

Figure 21 shows an overview of the exact MAD computation process. It takes a dataset $\mathcal{D}$ and a bucket limit M as the input, and outputs MAD of $\mathcal{D}$.

(1) The sketch is first constructed given dataset $\mathcal{D}$ and bucket limit M according to Section 3.1. Then, the median and MAD buckets are identified to bound the median and MAD of $\mathcal{D}$ as stated in Section 3.2. Based on the median and MAD buckets, the ranges J of median and MAD elements are determined in Section 3.3.
(2) Unless elements in J can be stored in memory, the approach will go through an iteration of refinement. The sketch will be refined according to the ranges J referring to Section 3.4. Then, median and MAD buckets in the refined sketch will be identified as discussed in Section 3.5. The new ranges J will be determined again by Section 3.3, and shrunk in Section 3.6.
(3) If elements in J can be loaded into memory, MAD of $\mathcal{D}$ will be calculated over J in memory according to Section 3.7.

B PROOFS

B.1 Proof of Proposition 4

Proof. To prove the proposition, we first give a range of $|\lambda - \eta|$ according to $d_{\min}$ and $d_{\max}$. There are two possible relationships of the locations of B_m and B_l as illustrated in Figure 5. For B_m and B_r, the situation is similar.

(1) If B_m and B_l satisfy $m-l \geq 1$, then there comes $\max\{-\epsilon, \zeta^{m-1} - \zeta^{l}\} = \zeta^{m-1} - \zeta^{l}$ as shown in Figures 4 (b) and 5. Similarly, if B_m and B_r satisfy $r-m \geq 1$, we have $\max\{-\epsilon, \zeta^{r-1} - \zeta^{m}\} = \zeta^{r-1} - \zeta^{m}$. Thus, according to Proposition 2, it follows $d_{\min} < |\lambda - \eta| < d_{\max}$.
(2) If B_m and B_l satisfy $m-l = 0$, there comes $\max\{-\epsilon, \zeta^{m-1} - \zeta^{l}\} = -\epsilon$. For B_r, there is $\max\{-\epsilon, \zeta^{r-1} - \zeta^{m}\} = -\epsilon$ similarly. Given $MAD \geq 0$, together with Proposition 2, we have $d_{\min} < 0 \leq |\lambda - \eta| < d_{\max}$.

Given $d_{\min} < |\lambda - \eta| < d_{\max}$ proved above, together with $\lambda \in \left[\zeta^{m-1}, \zeta^{m}\right)$ according to Proposition 3, we have $\eta \in J_l \cup J_r$. □

B.2 Proof of Lemma 10

Proof. We show below that there always exists a refined $\zeta' = \sqrt{\zeta} < \zeta$, satisfying the bucket limit in Formula 4 of refinement

$$\sum_{p=m,l,r} \left(\left\lceil \log_{\sqrt{\zeta}} J_p^{\uparrow} \right\rceil - \left\lfloor \log_{\sqrt{\zeta}} J_p^{\downarrow} \right\rfloor\right) \leq 2l{+}2\lambda{+}2\max\{\eta, \lambda{-}\eta\}{+}8 \leq M.$$

We denote $d_i = \zeta^{i} - \zeta^{i-1}$ the difference of the left and right boundaries of bucket B_i, and similarly $d_i' = (\zeta')^{i} - (\zeta')^{i-1}$. According to Proposition 3 and 4, it can be derived

$$J_l^{\uparrow} - J_l^{\downarrow} + J_m^{\uparrow} - J_m^{\downarrow} + J_r^{\uparrow} - J_r^{\downarrow} \leq [2d_m + d_r] + d_m + [2d_m + d_r],$$

and the equal sign holds when $d_{\min} = \zeta^{r-1} - \zeta^{m}$ and $d_{\max} = \zeta^{r} - \zeta^{m-1}$. Then, we discuss J_r, J_m and J_l seperately as follows.

(1) For J_r, $2d_m + d_r \leq d_{r-1} + d_r + d_{r+1}$ holds. Thus, $\left\lceil \log_{\zeta'} J_r^{\uparrow} \right\rceil - \left\lfloor \log_{\zeta'} J_r^{\downarrow} \right\rfloor \leq 2 \times [(r+1) - (r-2)] = 6$.

(2) For J_m, it has $\left\lceil \log_{\zeta'} J_m^{\uparrow} \right\rceil - \left\lfloor \log_{\zeta'} J_m^{\downarrow} \right\rfloor = 2$ as $J_m^{\uparrow} - J_m^{\downarrow} = d_m$.

(3) For J_l, we consider at most how many buckets may be contained. In the worst case, buckets with index from 1 to $2l$ are all in J_l. As $(\zeta')^{2l} - e_{\min} > d_r$ according to Proposition 4 and 7, and the bucket B_i' with $i > 2l$ has $d_i' > d_{2l}'$, we consider how many buckets

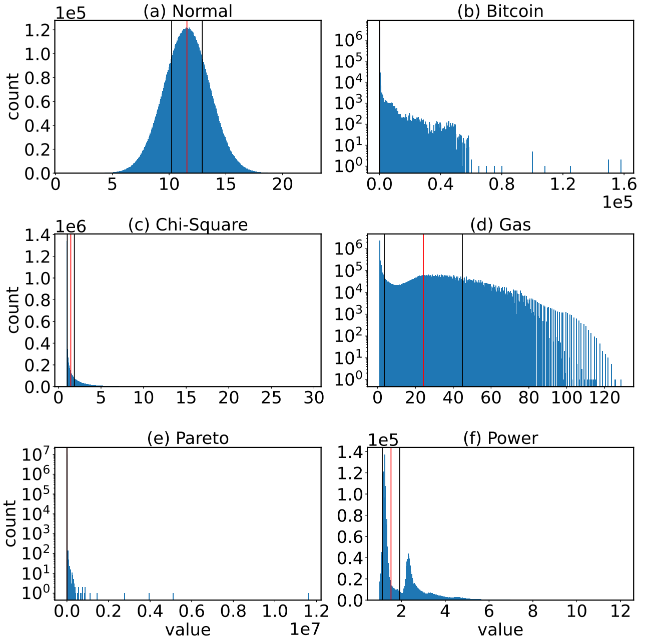

Figure 22: Histograms of datasets, with λ in red line, and $\eta = \lambda - \text{MAD}$, $2\lambda - \eta = \lambda + \text{MAD}$ in black lines

with index larger than $2l$ are in the range of length $d_r + d_m$.

$$\left\lceil \log_{\zeta'} J_l^{\uparrow} \right\rceil - \left\lfloor \log_{\zeta'} J_l^{\downarrow} \right\rfloor \leq 2l + \frac{d_m + d_r}{d'_{2l}}$$

$$\leq 2l + \frac{(\zeta^m - \zeta^{m-1})}{\left((\sqrt{\zeta})^{2l} - (\sqrt{\zeta})^{2l-1}\right)} + \frac{(\zeta^r - \zeta^{r-1})}{\left((\sqrt{\zeta})^{2l} - \sqrt{\zeta})^{2l-1}\right)}$$

$$< 2l + 2\lambda + 2\max\{\eta, 2\lambda - \eta\}$$

To sum up, the conclusion is proved. □

B.3 Proof of Proposition 12

Proof. We first show the time complexity of a single iteration, and then give the time complexity of the whole algorithm by analyzing the number of iterations.

For a single iteration, the algorithm first inserts all the data into CORE-Sketch, with time cost $O(N)$. Then, it searches for the median and MAD buckets according to Propositions 5 and 6, in $O(M \log M)$ time. Hence, the time complexity of a single iteration is $O(N + M \log M)$.

For the number of iterations, we consider the worst case, where the median and MAD element are in the buckets storing the most elements in the Sketch. We have $\max(|B_{i-1}| + |B_i| + |B_{i+1}|) \leq CN$ in this case, such that the data elements in these three buckets can fit in memory as stated in Section 3.7. This equals to

$$\max\left(F(\zeta^{i+1}) - F\left(\zeta^{i-2}\right)\right) \leq C.$$

Thus, we choose the maximum ζ_* which satisfies the above inequality, i.e., $\zeta_* = \max\left\{\zeta \mid \max_{i=1,\cdots,M-1}\left(F(\zeta^{i+1}) - F(\zeta^{i-2})\right) \leq C\right\}$. Recall that $\zeta = 2^{2^{-x}}$ with $x \in \mathbb{N}$ in Definition 2, we have

$$x_* = -\log\log\zeta_*.$$

Notice that ζ decreases by iterations according to Lemma 10. In the worst case, the algorithm starts with $\zeta = 2^{2^0}$ in sketch construction. For each iteration of refinement, it has $\zeta' = \sqrt{\zeta}$, i.e., $-\log\log\zeta$ increases by 1. To reach x_*, we have the upper bound on the number of iterations $f(C) = -\log\log\zeta_*$.

Combing the cost of a single iteration and the number of iteration, the time complexity of CORE-Sketch is obtained. □

C ADDITIONAL EXPERIMENTS

C.1 Experimental Settings

The experiments are conducted on a machine with 8-core 2.3 GHz CPU and 64 GB memory.

C.1.1 Dataset.

We employ 3 synthetic datasets and 3 real-world datasets with various distributions as illustrated in Figure 22.

(1) *Normal* is a synthetic dataset consisting of 1e10 elements generated from normal distribution $\mathcal{N}(4, 2)$.
(2) *Chi-Square* is a synthetic dataset consisting of 1e10 elements generated from chi-square distribution χ_1.
(3) *Pareto* is a synthetic dataset consisting of 1e10 elements generated from a heavy-tail distribution, pareto distribution [26] with shape parameter 1 and scale parameter 1.
(4) *Bitcoin* is a Kaggle public dataset composed of 1e7 elements [8]. It records the transaction data of bitcoin from 2009 to 2018.
(5) *Gas* is a dataset with heavy-tail distribution consisting of 1e7 elements [2]. It consists of measured values of carbon oxide concentration by a temperature-modulated metal oxide semiconductor.
(6) *Power*, acquired from UCI Individual Household Electric Power Consumption dataset [10], is composed of measurements of electric power consumption in a single household. It consists of 1e7 elements, and has two peaks in its distribution.

C.1.2 Baseline.

To the best of our knowledge, there is no advanced existing methods for computing exact MAD except searching the median twice. Thereby, we compare CORE-Sketch with the algorithms that use the median query algorithm twice, either exactly or approximately.

The No-Sketch method is based on Quickselect algorithm [11], and requires the data to be stored in memory. After two rounds of selection, as introduced in Section 1.2, median and MAD can be calculated, respectively.

The DD-Sketch method is based on the approximate median query algorithm DD-Sketch [19]. As illustrated in Figure 2, by conducting the DD-Sketch algorithm twice, we can acquire the approximate MAD of datasets.

The TP-MAD method [19] is an approximate MAD query algorithm. With a given parameter ϵ, it can return the approximate MAD of the dataset with error at most ϵ or 1.

C.1.3 Metric.

While the No-Sketch method and our CORE-Sketch return the exact MAD, the DD-Sketch approach estimates approximate MAD. Thereby, time cost, space cost and relative error are employed to compare the performances with the exact and approximate baselines. We repeat the MAD query several times, and report the average.

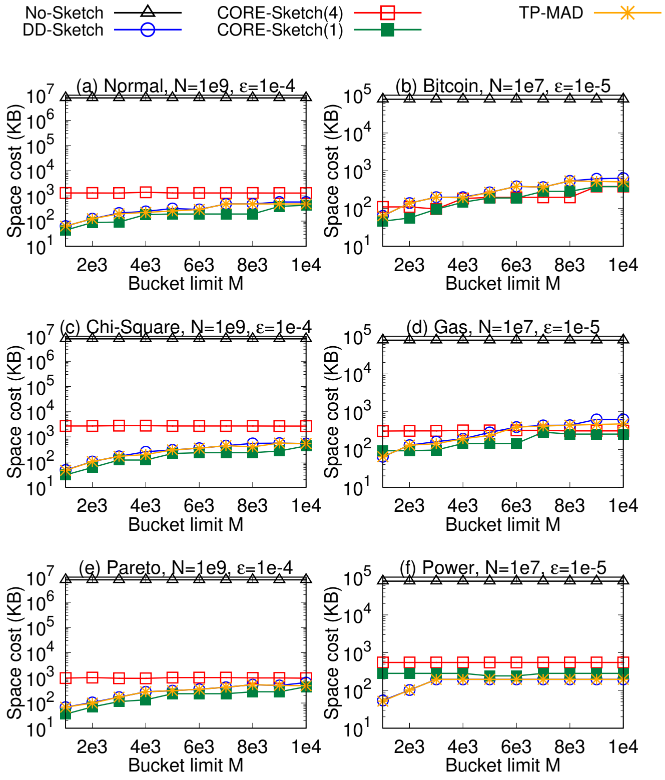

Figure 24: Space cost over bucket limit M

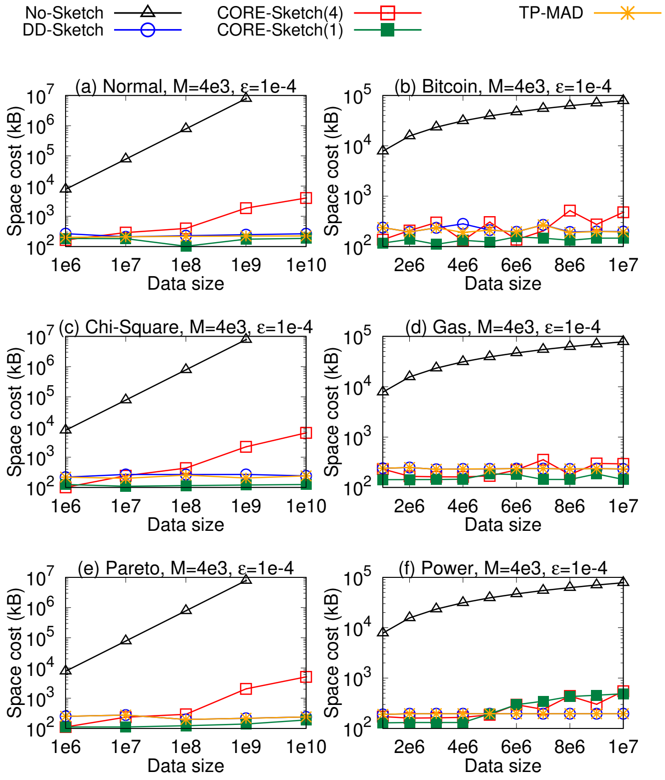

Figure 23: Space cost over data size N

(1) Time cost denotes the whole time of the computation process, including the initialization, iteration, input, output and so on.
(2) Space cost is the maximum memory occupied by the algorithm, including the sketch and the data elements stored in memory.
(3) Relative error is defined as the relative difference of the approximate MAD to the exact one.
(4) F1 score of anomaly detection shows the effectiveness of anomaly detection using median±kMAD. We set the anomaly detection result of exact MAD as ground truth.

C.2 Space cost

The space cost of CORE-Sketch and baseline algorithms over data size N and bucket limit M are shown in Figure 23 and 24.

C.3 Parallel Degree Evaluation

We study the performance of parallel computation in our proposed CORE-Sketch. The data size is $1e9$ for synthetic datasets and $1e8$ for real-world datasets.

Figure **??** depicts the time cost of CORE-Sketch by varying the number of threads in parallel computation. As shown, time cost decreases significantly when the number of threads increases from 1 to 4, while there is little improvement when the number of threads grows from 4 to 8. The reason is that the larger number of threads may bring about larger differences of ranges J among the threads as less elements may perform worse in representing the distribution of the whole dataset. Thus, the algorithm may go through more iterations to let $J \cap \mathcal{D}$ be read into memory. In brief, parallel computation in CORE-Sketch makes the efficiency of MAD exact computation comparable to the approximation method.