Hindawi
Security and Communication Networks
Volume 2021, Article ID 5564176, 17 pages
https://doi.org/10.1155/2021/5564176

Research Article

Generalized and Multiple-Queries-Oriented Privacy Budget Strategies in Differential Privacy via Convergent Series

Yunlu Bai,[1,2] **Geng Yang**,[1,3,4] **Yang Xiang**,[5] **and Xuan Wang**[1]

[1]*College of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210003, China*
[2]*Nanjing University of Chinese Medicine, Nanjing, Jiangsu 210023, China*
[3]*Key Laboratory of Broadband Wireless Communication and Sensor Networks Technology of Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210003, China*
[4]*Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing, Jiangsu 210003, China*
[5]*School of Software and Electrical Engineering, Swinburne University of Technology, Hawthorn, Victoria, Australia*

Correspondence should be addressed to Yunlu Bai; ylb@njucm.edu.cn

Received 17 February 2021; Accepted 8 November 2021; Published 8 December 2021

Academic Editor: Emanuele Maiorana

For data analysis with differential privacy, an analysis task usually requires multiple queries to complete, and the total budget needs to be divided into different parts and allocated to each query. However, at present, the budget allocation in differential privacy lacks efficient and general allocation strategies, and most of the research tends to adopt an average or exclusive allocation method. In this paper, we propose two series strategies for budget allocation: the geometric series and the Taylor series. We show the different characteristics of the two series and provide a calculation method for selecting the key parameters. To better reflect a user's preference of noise during the allocation, we explored the relationship between sensitivity and noise in detail, and, based on this, we propose an optimization for the series strategies. Finally, to prevent collusion attacks and improve security, we provide three ideas for protecting the budget sequence. Both the theoretical analysis and experimental results show that our methods can support more queries and achieve higher utility. This shows that our series allocation strategies have a high degree of flexibility which can meet the user's need and allow them to be better applied to differentially private algorithms to achieve high performance while maintaining the security.

1. Introduction

Collecting an individual's data is crucial for many applications [1], and privacy protection has always been one of the focuses of researchers. Classical privacy protection technologies such as anonymous algorithms and homomorphic encryption have already achieved remarkable results. There are also other technologies to achieve privacy protection; for example, Qi et al. [2, 3] proposed a faster privacy protection method with locality-sensitive hashing. However, most of these protection technologies do not take the adversary's background knowledge into consideration. Differential privacy, as a privacy protection technology with a rigorous mathematical foundation, not only takes into account the maximization of the adversary's background knowledge but also protects private data, which has received considerable attention.

Differential privacy works by incorporating noise into the statistical results obtained from the sensitive underlying data. This approach provides strong privacy guarantees by requiring the involvement of an individual in the data set to be indistinguishable in the released information; and one of the important factors affecting the performance of differential privacy is privacy budget.

Privacy budget ε (Dwork also called it "privacy loss" in the literature [4] in 2017), together with sensitivity of a query, is one of the critical parameters that will affect security and utility. For a specified query, the larger ε is, the lower the security is, and the less noise there is. In our common understanding, the noise is inversely proportional

to the utility; the larger the noise is, the lower the utility is. This means that if we want to achieve a high level of performance, the amount of noise cannot be too large; that is, a large privacy budget needs to be allocated. For sure, the above analysis is only for the application of a single query. But we tend to have similar thoughts when facing multiple queries. Thus, average allocation method usually becomes the first choice for most differentially private algorithms owing to its least overall noise [5–10].

However, does the average allocation strategy still bring a high level of utility if there are correlations between the queries? For example, we know that there are iterations in lots of data mining algorithms. A significant feature of iterations is to obtain the result of one certain iteration (or execution) that needs to rely on the results of previous iterations, and the most representative is clustering. When applying differential privacy to these algorithms, the input of each iteration is the noisy output of the previous iteration, and the output is still the result with noise to satisfy differential privacy [11]; and it raises two interesting questions: (1) Are there any iterations that have a greater impact on the final result of an algorithm than the other ones? Taking clustering as an example, we know that the initial centroids will affect the utility of the final clustering results. Does it mean that the first few iterations are more important and demand larger privacy budget terms than the other iterations? Or, in order to converge to a stable state of clusters, should the last few iterations be more important than other ones and a less amount of noise is required? (2) Does the noise have a direction? For any numerical data, we can regard them as points in space. Furthermore, since the noise value can be positive or negative, the noise added to each dimension of a point in the space can be regarded as different components of a vector that is, the overall noise has a direction. Once the direction is taken into account, the performance and utility of a differentially private algorithm depend not only on the amount of noise but also on the direction in which noise is added at each iteration. Both questions seem to imply that the average allocation strategy may not be so ideal. In fact, Cormode et al. [12] have already shown by experiments that the average strategy in their tree-based decomposition does not work well compared with the nonaverage method, which confirms our doubts from the side.

In addition, in the setting of random queries without considering correlation, although the average strategy may be superior in performance, there is a more serious problem, namely, the security problem. Since the budget for each query in the average allocation is equal, if the mean of the distribution of noise is 0, then the true result can be solved by conducting the exact same queries or queries of the same semantic [13]; it cannot defend against collusion attacks.

As such, we need to solve the problem of how to cut the pizza of the privacy budget.

On the issue of nonaverage allocation of the privacy budget, a simple method once given by Dwork is bisection allocation (the budget allocated for the first time is $\varepsilon/2$, and then the budget for the next query is 1/2 of the previous query). However, bisection allocation is rarely applied in practice. This is because of the fact that the allocation speed of the budget is too fast and the noise is too large to support many iterations. On the other hand, people usually determine the number of iterations in advance, and they will naturally choose the average allocation once they have determined the number of iterations. Many researchers have discussed optimizing the privacy budget allocation for their work; however, most optimizations are exclusive or just the average allocation in disguise. In addition, the current nonaverage allocation methods seem to be limited to a fixed mindset in which the allocation of budgets should be a monotonic strategy, especially monotonically decreasing. To date, there is still no discussion on the nonmonotonic allocation. Moreover, although Dwork once discussed the theoretical lower limit of the budget, each user's tolerance for noise is different in practice. The discussion on how to construct a budget allocation strategy that meets the practical needs of a user is also lacking.

To solve these problems, we consider the portion of the overall budget assigned to a query as one term in a sequence; hence, the whole process of allocation could be treated as a series. We focus on using the existing series technology to allocate the privacy budget. The main contributions of this paper can be summarized as follows:

(i) We propose two series allocation strategies which are of different monotonic features. The geometric series provides a monotonic allocation of the budget, while the Taylor series provides a nonmonotonic allocation of the budget. We show how a parameter of a series affects the performance of series allocation methods.

(ii) We give a simple calculation method to select ratio r and the initial parameter t to meet the utility requirements of multiple queries in practice from the perspective of the interactive scheme.

(iii) To reflect user's preference on noise and to meet a higher utility, we discuss in detail the relationship between sensitivity and noise and propose the idea of sensitivity normalization. Then, we present the concept of an acceptable budget and an optimization approach for the series strategies on the basis of the acceptable budget.

(iv) To defend the collusion attacks and improve security, we provide three different ideas for protecting the allocation sequence.

The remainder of this paper is organized as follows. We introduce related work in Section 2 and give the preliminary results in Section 3. In Section 4, we propose two series strategies and discuss the method to select the key parameters. In Section 5, we explore the relationship between sensitivity and noise, and we also present an approach for optimization. In Section 6, we discuss the protection of the budget allocation sequence. In Section 7, we present the experimental analysis of series strategies. Some conclusions are given in Section 8.

2. Related Work

As a core parameter, the privacy budget determines the security level of differential privacy. Although there is a lack

of a specialized discussion on the allocation of privacy budget, it has been briefly discussed in many differentially private algorithms by researchers owing to the importance of this topic. The existing allocation methods generally fall into the three following kinds of strategies.

The first kind is to apply the total privacy budget as a whole without allocation, which is according to the parallel composition theorem [4, 14]. This situation has been widely used in the data publishing. As mentioned in the following articles, one of the methods proposed in the work of Xu et al. [15] is to independently add Laplace noise to every count x_i; Xiao et al. [16] added independent Laplace noise to each wavelet coefficient; and both research groups provided the methods that could be implemented with no need to allocate the privacy budget.

The second strategy is to allocate the budget uniformly, namely, the average strategy. This is the most commonly used strategy in the fields of data publishing and data analysis. It is the simplest approach to implement and may be very effective at times. Several studies have been widely performed. For example, Chen et al. [17] and Qardaji et al. [18] implemented differentially private data publication approaches with a uniform budget allocation, as does another method proposed in the work of Xu et al. [15]. Dwork [13] and Su et al. [6, 7] allocated the budget equally with a fixed number of iterations for k-means clustering. Friedman et al. [5], Jagannathan [8], and Rana et al. [19] all applied a uniform budget to construct decision trees. Bhaskar et al. [20] proposed two differentially private frequent pattern mining algorithms with uniform allocation. Using the average method, Shin et al. [9] divided the overall budget into each iteration for the recommendation system. There are also a large number of other applications that seem to use nonuniform allocations, but, essentially, they still utilize the average method. For example, Hua et al. [21] adopted the linear allocation strategy that divides ε into two parts ε_1 and ε_2, and thus $k \times \varepsilon_1 + \varepsilon_2 = \varepsilon$. Similarly, Wu et al. [22] divided ε into four different parts corresponding to the four original parameters in the density estimation with the Gaussian mixtures model (GMM). Then, they both divided some parts of the budget into even smaller values with the average method to conduct iterations. Li et al. [23] and Cheng et al. [10] also demonstrated similar approaches. However, as we already know, security issues might occur with the average strategy, and the average method is not always the most effective one, which was confirmed by Cormode et al. [12].

The last kind of allocation strategy, namely, the adaptive strategy, is to make an optimization only oriented to some specific work. The representatives are as follows. For tree-based decomposition, Cormode et al. [12] adopted a geometric allocation method that minimizes the error according to the height of the tree, and let $\varepsilon_i = 2^{(h-i)/3} \times \varepsilon \times (2^{1/3} - 1)/(2^{(h+1)/3} - 1)$, where ε_i is the privacy budget allocated to level i and h is the height of the tree. In addition, inspired by [12], Qardaji et al. [18] used a local optimal allocation strategy that minimizes the average error variance by letting $\varepsilon_1 = \varepsilon / \sum_{i=1}^{h} (a_i/a_1)^{1/3}$ and $\varepsilon_i = (a_i/a_1)^{1/3} \times \varepsilon_1$, where a_i is the weight of level i. Fan et al. [24] proposed the idea of arithmetic progression allocation based on the work of Su et al. [6], but the biggest problem of this allocation is the convergence of the budget sequence. To satisfy the requirement for the total budget, the arithmetic progression allocation has been very close to the average allocation even when the number of queries is not large. Basically speaking, all the methods mentioned above have one common problem, which is too exclusive to use in general situations.

Although much work has been done with differential privacy, the research that focuses on the privacy budget allocation strategy is still limited. Previous studies in the literature [25] have explored how different users allocate and compete for privacy budgets but did not address the budget allocation for a query sequence. Existing approaches either have adopted uniform allocation or are exclusive and can only be used in some specific scenarios. Therefore, because of these limitations, our interest has been stimulated and we are motivated to carry out this work.

3. Preliminaries

Differential privacy was a result of a line of work that was presented by Dwork [11, 26] in 2006. A brief introduction of differential privacy is given as follows.

Definition 1 (ε– differential privacy). A (randomized) mechanism M satisfies ε– differential privacy if any data sets D and D' differ with respect to at most one element (one is the subset of the other; we consider them as neighbours) and for any subset of the output $O \in \text{Range}(M)$,

$$\Pr[M(D) = O] \leq e^{\varepsilon} \times \Pr[M(D' = O)]. \tag{1}$$

Parameter ε controls the leakage of the privacy of M and is referred to as the privacy budget.

Definition 2 (global sensitivity). For any function $f: D \longrightarrow R^d$, the global sensitivity of f denoted as Δf is defined to be the maximum L_1 distance of the output from any two neighbouring data sets D and D'.

$$\Delta f = \max_{D,D'} \left\| f(D) - f(D') \right\|_1. \tag{2}$$

Any mechanism meeting Definition 1 can be considered as differentially private.

To achieve differential privacy, Laplace mechanism has been widely used, which adds random noise to the true answer [11]. Let $\text{Lap}(b)$ denote the Laplace distribution with mean 0 and a scale parameter b. Then the probability density function is

$$p(x) = \frac{1}{2b} e^{-|x|/b}. \tag{3}$$

Theorem 1 (Laplace mechanism). *Given a data set D, for any function $f: D \longrightarrow R^d$ with sensitivity Δf, the random mechanism $M(D)$ provides ε– differential privacy if*

$$M(D) = f(D) + \eta. \tag{4}$$

We have $\eta \sim \text{Lap}(\Delta f/\varepsilon)$.

There are two widely used properties in this paper: sequential composition and parallel composition [14]. The sequential composition is the basis for the establishment of this article.

Theorem 2 (sequential composition). *For a sequence of mechanisms $M_1, M_2, \ldots, M_n$, if each of them provides ε_i- differential privacy, then the composed mechanism $M = \sum_{i=1}^{n} M_i(D)$ over the data set D provides $(\sum_{i=1}^{n} \varepsilon_i)-$ differential privacy.*

Theorem 3 (parallel composition). *For a sequence of mechanisms $M_1, M_2, \ldots, M_n$, each of them provides ε_i- differential privacy; let D_i be arbitrary disjoint subsets of the data set D; then the composed mechanism $M = \sum_{i=1}^{n} M_i - (D)$ provides max $(\varepsilon_i)-$ differential privacy.*

We list notations that are frequently used throughout the paper in Table 1.

4. Privacy Budget Allocation via a Convergent Positive Series

In this section, we will introduce two privacy budget allocation strategies based on two convergent and positive series (the geometric series and the Taylor series). We show the role of core parameters and the method for selecting them.

Although these two series are well-known classics in the field of mathematics, they have not been widely used in budget allocation. To the best of our knowledge, it is the first time that the Taylor series has been applied to budget allocation. With regard to the geometric series, it is the easiest to understand and apply, but only a few existing allocations are geometric methods, and those methods lack generality and flexibility. In addition, there is still no induction of such methods. Given that, our clear ideas of these two series allocation strategies are presented for the first time to date.

4.1. Privacy Budget Allocation and Convergent Positive Series. If the overall budget for a differentially private mechanism M is ε, then the budget allocated to the i_{th} query can be expressed as follows:

$$\varepsilon_i = k_i \varepsilon, \tag{5}$$

where k_i is the proportion of the budget term ε_i in the overall budget ε and $0 < k_i < 1, i \in N^+$.

The allocation of a privacy budget ε is like cutting a pizza; no matter how many people there are, it must be ensured that everyone can get a slice, no matter how small the slice is. This implies that the mathematical form of budget allocation needs to be discrete, positive, and with a summation of a constant; and the convergent positive series just perfectly meets these requirements. Therefore, the overall budget ε allocated to a sequence of queries can be expressed as a budget sequence $\{\varepsilon_n\}$. Similarly, we have the proportion sequence $\{k_n\}$, and the sum is $S_n = \sum_{i=1}^{n} k_i < 1$. If $n \longrightarrow \infty$, then S_n can be expressed as a convergent infinite series $\sum_{i=1}^{\infty} k_i$, and there is $\sum_{i=1}^{\infty} k_i = 1$ (in fact, we should obey the sublinear constraint proposed in the literature [27], but, for the analysis, it will not be considered in this section).

Theorem 4. *If a positive series $\sum_{i=1}^{\infty} p_i$ converges to a constant c, the overall privacy budget is ε, given a sequence of mechanisms $M_1, M_2, \ldots, M_n$, $n \in N^+$, each of them provides ε_i- differential privacy, and $\varepsilon_i = k_i\varepsilon = p_i/c\varepsilon$; then the composed mechanism $M = \sum_{i=1}^{n} M_i(D)$ over the data set D satisfies ε-differential privacy.*

Proof. For any $n \in N^+$, the sum of the sequence $\{k_n\}$ is $S_n = \sum_{i=1}^{n} p_i/c = 1/c \sum_{i=1}^{n} p_i$. Since $\sum_{i=1}^{\infty} p_i = c$, it is easy to see that $\sum_{i=1}^{n} p_i \le c$, $S_n = 1/c \sum_{i=1}^{n} p_i \le 1$, and we have $\sum_{i=1}^{n} \varepsilon_i = \sum_{i=1}^{n} p_i/c\,\varepsilon = \varepsilon S_n \le \varepsilon$. According to Theorem 2 (Sequential Composition), we learned that the composed mechanism $M = \sum_{i=1}^{n} M_i(D)$ provides $(\sum_{i=1}^{n} \varepsilon_i)-$ differential privacy; that is, M satisfies $\varepsilon-$ differential privacy. □

4.2. Geometric Series Strategy. Geometric series is the easiest series for people to think of and implement when allocating privacy budget.

A geometric series can be expressed as $\sum_{i=1}^{n} ar^{i-1}$, where a is the start term, and r is the ratio of two adjacent terms. According to D'Alembert's test, the series converges to 1 only if $0 < r < 1$ and $a = 1 - r$. The series is represented as follows:

$$\sum_{i=1}^{\infty} ar^{i-1} = \sum_{i=1}^{\infty} (1-r)r^{i-1} = 1. \tag{6}$$

Hence, we have

$$k_i = (1-r)r^{i-1}, \quad i \in N^+. \tag{7}$$

According to Theorem 4, when using geometric series for budget allocation, the $\varepsilon-$ differential privacy can be satisfied. Although the geometric series strategy is simple and easy, the value of r will significantly affect the steepness of the series curve. If r is small, for example, $r = 1/4$, then the speed of allocating the budget is relatively fast and the first term is large. In addition, if the overall budget ε is allocated too quickly, the remaining budget available for allocation will soon be reduced to 0. However, if r is large enough and close to 1, then the series curve will be quite flat and the first budget term would be relatively small. For example, when $r = 0.9$ and $r = 0.5$, the comparison of these two series is shown in Figure 1. The series with $r = 0.9$ is much gentler than that of $r = 0.5$.

In fact, the bisection method and Cormode method mentioned in the previous sections are essentially the special cases of geometric series. To be specific, the bisection method is the geometric series with $r = 1/2$, and the Cormode method is the geometric series with $r = \sqrt[3]{2}$. However, they are both fixed-form methods and do not have the ability to be flexible according to practical needs.

Thus, choosing a reasonable r to meet more meaningful queries should be considered when allocating via the geometric series strategy.

Table 1: List of frequent notations.

Notation	Meaning
ε	The overall privacy budget for a sequence of queries
ε_i	The budget allocated to the i_{th} query
k_i	The proportion of ε_i in ε
r	The ratio of successive terms in the geometric series
t	The initial term of the Taylor series
Δf	The global sensitivity of a query function f
n	The number of queries
k_i^*	The proportion k_i of ε_i in ε after calibration
α	The multiplicative factor for optimization
U_s	The upper bound of the standard deviation of noise with $\Delta f = 1$
$\{\varepsilon_n\}$	The budget sequence with n terms $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$
$\{k_n\}$	The proportion sequence with n terms $k_1, k_2, \ldots, k_n$
S_n	The sum of the first n terms of the sequence $\{k_n\}$

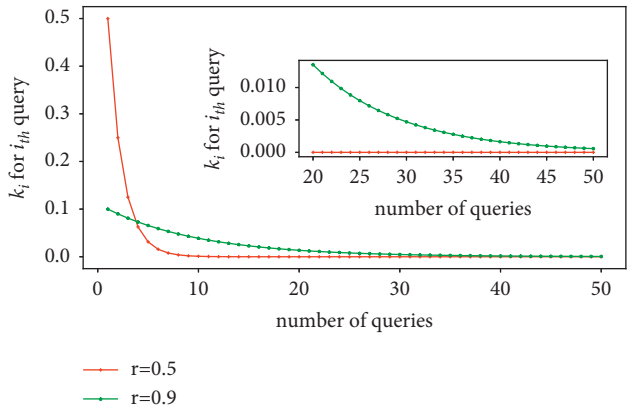

Figure 1: Comparison of geometric series curves with different r.

4.3. Taylor Series Strategy. A monotonically decreasing budget allocation method is in line with people's daily thoughts and behavior habits, but it may not be the most appropriate. Thus, sometimes we can change our way of thinking, that is, using a nonmonotonic way of allocation. Although it is hard to find a convergent series that is both nonmonotonic and positive, we found an unexpected surprise after analyzing the Taylor series carefully.

A general definition of Taylor series is shown below. If a function $f(x)$ has derivatives of all orders at a single point x_0, then $f(x)$ could be represented by a Taylor series, which is an infinite sum of terms calculated from the values of the function's derivatives at x_0:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^n(x_0)}{n!}(x - x_0)^n + \cdots. \tag{8}$$

To ensure that each term is positive, we choose the exponential function $f(x) = e^x$, and the Taylor series for e^x at $x = x_0$ is as follows:

$$e^x = e^{x_0} + e^{x_0}(x - x_0) + \frac{e^{x_0}}{2!}(x - x_0)^2 + \cdots. \tag{9}$$

When $\varepsilon = e^x$ and $\varepsilon_1 = e^{x_0}$, the equation above can be written as follows:

$$\varepsilon = \varepsilon_1 + \varepsilon_1 \ln\frac{\varepsilon}{\varepsilon_1} + \frac{\varepsilon_1}{2!}\left(\ln\frac{\varepsilon}{\varepsilon_1}\right)^2 + \frac{\varepsilon_1}{3!}\left(\ln\frac{\varepsilon}{\varepsilon_1}\right)^3 + \cdots. \tag{10}$$

Since $\varepsilon_1 = k_1\varepsilon, 0 < k_1 < 1$, we can turn the equation into

$$\begin{aligned}\varepsilon &= k_1\varepsilon + k_1\varepsilon \ln\frac{\varepsilon}{k_1\varepsilon} + \frac{k_1\varepsilon}{2!}\left(\ln\frac{\varepsilon}{k_1\varepsilon}\right)^2 + \cdots \\ &= \sum_{i=1}^{\infty} k_i\varepsilon = \varepsilon\sum_{i=1}^{\infty}\frac{k_1}{(i-1)!}\left(\ln\frac{1}{k_1}\right)^{i-1}.\end{aligned} \tag{11}$$

Let $k_1 = t$. Then, we have the following:

$$k_i = \frac{t}{(i-1)!}\left(\ln\frac{1}{t}\right)^{i-1}, \; i \in N^+, \; 0 < t < 1. \tag{12}$$

Since $\sum_{i=1}^{\infty} t/(i-1)!\,(\ln 1/t)^{i-1} = 1$, the Taylor series strategy still meets the $\varepsilon-$ differential privacy after an unlimited allocation of the privacy budget.

What Taylor series strategy needs to concern and select is the first term k_1, namely, the value of t. We draw the curves of Taylor series as shown in Figure 2. Interestingly, when t is small in the ordinary sense, as shown in Figure 2(a), it creates an illusion that no matter how small t is, the speed of budget consumption is extremely fast. However, when t is extremely small, the Taylor series begins to show an approximately symmetrical waveform, as shown in Figure 2(b); and this provides a new way of allocation to us.

A direct comparison of the two series is shown in Figure 3, which makes it easier to see their respective allocation characteristics. The characteristic of the geometric series is to allocate the budget in a monotonically decreasing way, and the speed of the budget decrement can be adjusted by adjusting parameter r. The Taylor series is characterized by the allocation with a waveform; that is, the allocation speed is higher in the middle stage and slow in the other stages, and the parameter that controls the waveform is t.

4.4. Determine the Key Parameters of the Series Strategies. Determining the key parameters r and t for the series strategies is actually determining the consumption speed of budget ε, in other words, the steepness of the series curve.

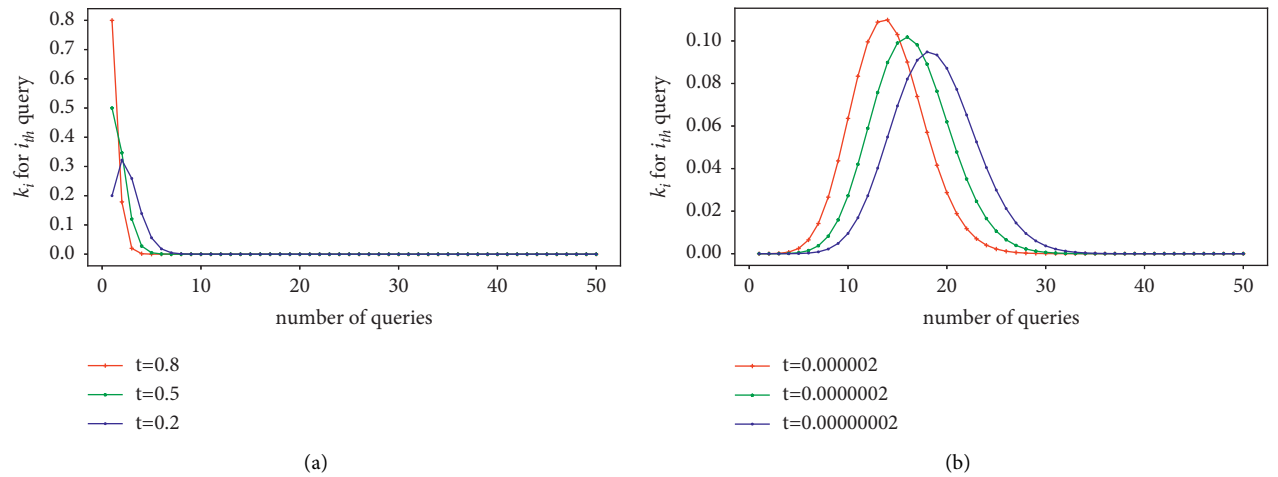

Figure 2: Comparison of the Taylor series curves with different t values.

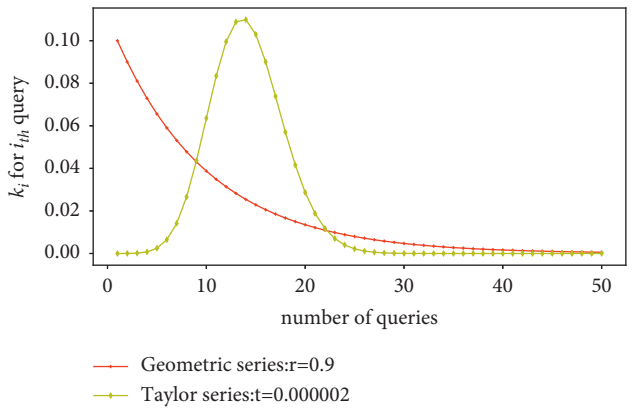

Figure 3: Comparison of two series curves.

The speed of budget consumption and the total amount of noise will both affect the actual output utility. By calculation, we can know that the total amount of noise added with the average method must be the least; and although the series strategy is a kind of uneven allocation, it may also perform poorly with excessive imbalance. An unbalanced budget allocation will bring an unbalanced addition of noise. As a result, very little noise will be added to some queries, while others are meaningless due to excessive noise.

We note that, to satisfy the convergence, although both series curves can be extremely flat and nearly average in the limiting case, their unique advantages are also lost at the same time. Therefore, we need to make a trade-off between average and nonaverage allocation strategies.

Since the expressions and features of two series are different, we solve the problem separately.

Geometric Series. For the geometric series, reasonable r can ensure that the difference between the first budget term and the last budget term is not too small or too large. By analyzing Figure 1, we can find that, at any i_{th} term, there must be a certain geometric series whose k_i value is greater than k_i of all other geometric series. Based on this, we only focus on the last budget term, and the proportion k_n of ε_n can be written as a function of r:

$$k_n(r) = (1-r)r^{n-1}. \tag{13}$$

We can derive the function to obtain the extreme point as

$$r = \frac{n-1}{n}. \tag{14}$$

This equation allows us to calculate a relatively ideal r value for n queries.

Taylor Series. Since the Taylor series can exhibit a waveform which is approximately symmetric, we focus on the budget term in the middle, that is, the $\lceil n/2 \rceil_{\text{th}}$ term. To ensure a complete waveform, we need to let $n \geq 3$. Then, the proportion of $\varepsilon_{\lceil n/2 \rceil}$ for the Taylor series can be written as a function of t.

$$k_{\lceil n/2 \rceil}(t) = \frac{t}{(\lceil n/2 \rceil - 1)!}\left(\ln \frac{1}{t}\right)^{\lceil n/2 \rceil - 1}, \tag{15}$$

and the extreme point can be derived as

$$t = e^{1-\lceil n/2 \rceil}. \tag{16}$$

Take 50 queries as an example, the curves of the two series with their ideal parameters are shown in Figure 4. It seems that the two series perform well. However, we still have some extra budget available.

4.5. Calibration of the Series Strategies. Since any kind of convergent series is for infinite terms, the series strategy can certainly be assigned to an infinite number of queries. But we only use the first n terms for n queries. This means that our overall budget ε has not been used up, so extra budget is available to magnify each term in the budget sequence. Assuming that the sum of the proportion sequence $\{k_n\}$ is $S_n = \sum_{i=1}^{n} k_i < 1$, the actual proportion k_i^* of ε_i in the total budget ε should be

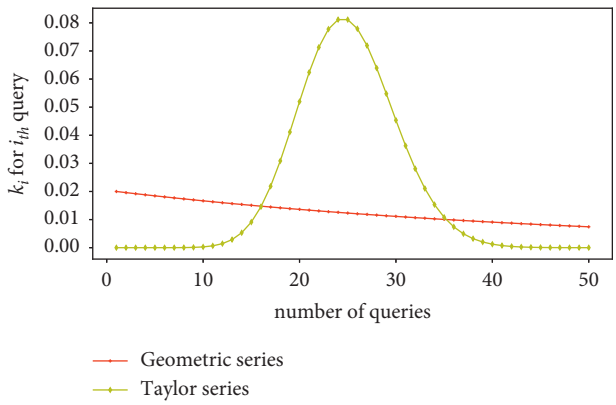

Figure 4: Two series with ideal parameters for 50 queries.

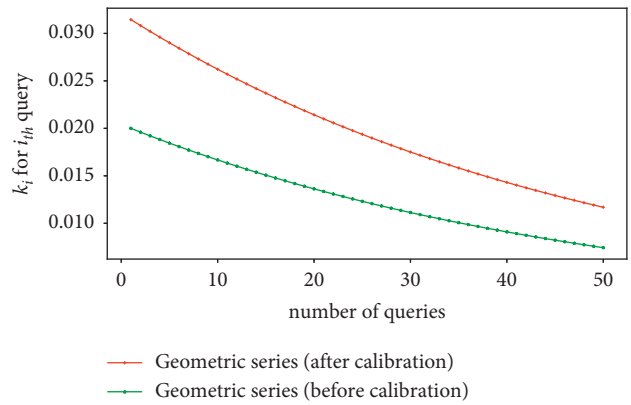

Figure 5: Calibrated budget sequence for 50 queries.

$$k_i^* = \frac{k_i}{S_n}. \tag{17}$$

The actual privacy budget for the i_{th} query should be $\varepsilon_i^* = \varepsilon k_i^*$. We refer to this process as the calibration (the essence of calibration is scaling up; to distinguish it from the data scaling operation later in the article, we use the notion of calibration). Taking the geometric series as an example, the calibrated budget sequence is shown in Figure 5.

Note that the following text describes why we solve the ideal parameters after the calibration rather than before it.

First, there is no general formula for the Taylor series to quickly obtain the summation, so the way to solve after calibration is not realistic. Second, although there is a convenient formula for the geometric series to calculate the sum of the first n terms written as follows:

$$S_n = 1 - r^n, \tag{18}$$

solving the extreme point of function $k_n^*(r) = k_n(r)/S_n$ is also complicated and difficult. Moreover, it can be judged empirically and intuitively that the extreme point of the n_{th} term of the geometric series after calibration is just $1/n$, which is the same as the average allocation. Therefore, we do not have to seek the optimal solution; an approximate optimum solution is enough to meet our needs.

It should be noted that the drawing of the series curve and the calculation of the series term easily give us an illusion; that is, given any function and randomly taking n outputs, the same allocation effect can be achieved. But it is not feasible. The biggest difference between function and our series strategy is that the sum of all terms in a function does not converge to a constant; and whether it is a continuous or discrete function, how to determine the value interval is also a problem. This means that the function method cannot predict the monotonicity of budget allocation and the speed of consumption; that is, we cannot allocate effectively with function method.

5. Optimization for the Privacy Budget Sequence $\{\varepsilon_n\}$

In this section, we will present the optimization from two perspectives. One is to transform the obtained budget sequence $\{\varepsilon_n\}$, and the other is to improve the sequence to meet the user's requirements for accuracy.

5.1. Flip the Sequence. Our series strategies are flexible and simple. The process of budget allocation is only related to the number of queries and is independent of the actual database and specific query function. Once the number of queries has been determined in advance, the whole budget sequence $\{\varepsilon_n\}$ will be known, which means that we can transform the sequence for some specific applications. A good way to transform the sequence while still maintaining the allocation characteristic is to flip the sequence. The monotonic feature of an allocation sequence can be quickly adjusted by flipping.

This approach is simple for the geometric series; the sequence only needs to be flipped horizontally (namely, reverse). However, for the Taylor series, in addition to flipping the sequence vertically, a calibration is also required.

The budget term for the i_{th} query after horizontal flipping is as follows:

$$\varepsilon_i^F = \varepsilon_{n-i+1}^*. \tag{19}$$

Proportion k_i^F of ε_i^F is as follows:

$$k_i^F = k_{n-i+1}^*. \tag{20}$$

For the Taylor sequence, the budget term for the i_{th} query after flipping vertically and its proportion are as follows:

$$\begin{aligned} \varepsilon_i^F &= \frac{\varepsilon - \varepsilon_i}{n-1}, \\ k_i^F &= \frac{1 - k_i}{n-1}. \end{aligned} \tag{21}$$

The transformations of the two series after calibration for 50 queries are shown in Figure 6. For strategies that have been flipped, we add the prefix "Flip-" to their names.

5.2. Acceptable Budget. Although series strategies and their transformations look good, they still do not reflect users' willingness for accuracy. Since noise can be a direct reflection of accuracy, we want users to be able to adjust the

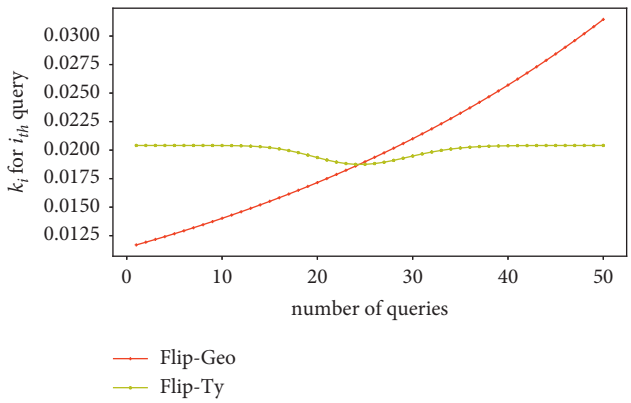

FIGURE 6: Comparison of the flipping series with $n = 50$.

allocation methods based on their tolerable upper bound of noise.

5.2.1. Sensitivity Normalization. A user can easily give an expected upper bound of noise if all queries are of the same type. However, when facing different query types, it is difficult for users to provide a uniform and tolerable upper noise limit, since different types of queries have different sensitivities, and the sensitivity is an important factor that affects the amount of noise. With the same budget, the higher the sensitivity of a query is, the larger the noise required is. Then, how can we set a uniform upper bound of noise for different queries without considering their sensitivities? That is, can we make the different sensitivities the same?

From the perspective of mathematical definition, the query sensitivity should not depend on the data set itself, since the privacy guarantee of differential privacy should hold for all possible data sets. However, in the practice of differential privacy for realistic data set, the sensitivity of a query function is affected not only by its own mathematical property but also by some specific numerical bounds or semantic constraints. This means that the data added with noise also needs to satisfy these constraints. For instance, data such as score and age should be between 0 and 100. No matter the value after adding noise is over or under the interval, it is meaningless, not to speak of interference to the adversary. Since differential privacy itself does not limit the adversary's background knowledge, we should assume that these data intervals have been known by the adversary. In other words, for some specific types of data in a data set, we may have a clear border constraint. As a result, the sensitivity of a query function conducted on this data set may be affected, which also implies that the sensitivity is closely related to the range (domain) of data. In fact, similar ideas have already been proposed by Dwork [6, 13].

It seems that such a phenomenon conflicts with the original definition of query sensitivity, but it does not. We can analyze it from another perspective; the same function that acts on the different data range can actually be regarded as a different function. Namely, $f(x), x \in (a, b)$ and $f(x), x \in (c, d)$ are the same query function if and only if $a = c, b = d$; otherwise, they are two different functions.

Therefore, we argue that one can scale the query sensitivity by scaling the value domain of the data set, except for the count query, since its sensitivity can always be 1. Therefore, we consider adjusting the sensitivity of different queries to 1. For example, given a sum query, assuming the original value domain of the data set is $[-v, +v]$, the sensitivity is $\Delta f = v$ (here readers need to notice the definition of the neighbour data set we used in Definition 1). However, if the domain is reduced by a factor of v and converted to $[-1, +1]$, then the sensitivity will be $\Delta f = 1$, which is the same as the count query [4]. Same operation has been adopted when processing in literatures [6, 13] to clamp the sensitivity of a function within an interval.

However, it does not mean that if we clamp the value domain of all the dimensions of a data set to $[-1, +1]$, all statistical queries can satisfy the sensitivity $\Delta f = 1$. For instance, for a max query, the domain to satisfy $\Delta f = 1$ should be $[0, 1]$ or $[-0.5, +0.5]$ instead of $[-1, +1]$. For complex practical applications, we cannot guarantee that unifying the domain for all dimensions could constrain the sensitivity of all queries to 1. Therefore, we allow each query function to scale its range separately to satisfy $\Delta f = 1$ before execution. Then, the noise will only be related to the budget and be independent of the sensitivity. We define this operation as sensitivity normalization.

Definition 3 (sensitivity normalization). If the global sensitivity of a query function f after scaling the value domain of data set D is $\Delta f_{\text{norm}} = 1$, then this process is considered as sensitivity normalization.

That is, we need to meet the following equation:

$$\Delta f_{\text{norm}} = \max_{D,D'} \left\| f(D_z) - f(D'_z) \right\|_1 = 1, \tag{22}$$

where z is the factor to scale the value domain of D for the query function f and D_z is the data set after scaling. If the original sensitivity is $\Delta f = v$, then we have $f(D_z) \times v = f(D)$, since $\Delta f_{\text{norm}} \times v = \Delta f$ (readers should be careful not to associate query functions with arbitrary mathematical functions).

5.2.2. Acceptable Budget and the Upper Bound of Noise. If the sensitivity of a query has been normalized, then its corresponding noise compared to the original noise has also changed, and we call it the normalized noise.

Although the corresponding noise has changed, the impact of noise on the true result has not changed. Assume that the true result before sensitivity normalization is

$$\text{TR} = f(D), \tag{23}$$

and if the original sensitivity is $\Delta f = v$, then the standard deviation of Laplace noise is

$$\text{NR} = \frac{\sqrt{2}\Delta f}{\varepsilon} = \frac{\sqrt{2}v}{\varepsilon}. \tag{24}$$

The actual result after sensitivity normalization and the standard deviation of normalized noise are as follows:

$$\begin{aligned} \mathrm{TR}^* &= f(D_z) = \frac{\mathrm{TR}}{\nu}, \\ \mathrm{NR}^* &= \frac{\sqrt{2}}{\varepsilon}. \end{aligned} \tag{25}$$

The ratio of noise to the true result remains consistent before and after normalization; that is, $\mathrm{TR}/\mathrm{NR} = \mathrm{TR}^*/\mathrm{NR}^*$. The principle is simple, since we only adjust the data domain proportionally without modifying the essence of the data. Therefore, no matter how much data have been scaled, it does not affect the ratio between noise and its corresponding true result.

By normalizing the sensitivity, we can achieve a unified noise constraint on different query functions. In other words, we can obtain the lower bound of the budget from the perspective of the user.

Definition 4 (acceptable budget). Assuming that one gives the upper bound of the normalized noise that he/she can tolerate as U_s, a query f has been sensitively normalized and its standard deviation of the normalized noise with privacy budget ε' satisfies

$$\frac{\sqrt{2}}{\varepsilon'} \le U_s, \tag{26}$$

and then we consider ε' as an acceptable budget.

Note that the noise we draw from the Laplace distribution is random, so we use the upper bound of noise provided by user as the maximum standard deviation of random noise when allocating budget.

Lemma 1. *If the minimum budget term $\varepsilon_{\min}$ in a budget sequence $\{\varepsilon_n\}$ satisfies*

$$\varepsilon_{\min} \ge \frac{\sqrt{2}}{U_s}, \tag{27}$$

then all the budgets in this sequence are acceptable.

Proof. According to the definition of an acceptable budget, for any acceptable ε', we have $\varepsilon' \ge \sqrt{2}/U_s$. Since $\varepsilon_{\min}$ is the minimum budget term of the budget sequence $\{\varepsilon_n\}$, for any ε_i, $i \in [1, n]$, there is $\varepsilon_i \ge \varepsilon_{\min} \ge \sqrt{2}/U_s$; thus, any budget term ε_i in the sequence is acceptable. The proof is complete.

Lemma 1 indicates that if one needs to conduct n queries with n acceptable budget terms, then the minimum budget term $\varepsilon_{\min}$ in the budget sequence must satisfy $\varepsilon_{\min} \ge \sqrt{2}/U_s$. □

Lemma 2. *Given the overall budget ε and the user's upper bound U_s of the normalized noise, to allocate budget for n queries successfully, there must be $\sqrt{2}n/\varepsilon \le U_s$.*

Proof. Assume that there is $U_s < \sqrt{2}n/\varepsilon$. Then, the minimum budget term for n queries is $\varepsilon_{\min} \ge \sqrt{2}/U_s > \varepsilon/n$; thus, $n \times \varepsilon_{\min} > \varepsilon$, which is a contradiction to $\sum_{i=1}^{n} \varepsilon_i \le \varepsilon$. Therefore, this assumption is incorrect, and the proof is complete. □

Lemma 3. *Given an allocation budget sequence $\{\varepsilon_n\}$, the minimum term in $\{\varepsilon_n\}$ is ε_k, $1 \le k \le n$, and the overall budget is $\sum_{i=1}^{n} \varepsilon_i = \varepsilon$. To make the noise bound meaningful and for effective optimization, there should be $\sqrt{2}n/\varepsilon \le U_s \le \sqrt{2}n/\varepsilon_k$.*

Proof. Since the noise that current budget sequence can endure is $\sqrt{2}n/\varepsilon_k$ at most, we think that there is no need for a further improvement if one gives $U_s > \sqrt{2}n/\varepsilon_k$. According to Lemma 2, there should be $\sqrt{2}n/\varepsilon \le U_s$. The proof is complete. □

5.3. Optimization with an Acceptable Budget. How could we improve the obtained budget sequence $\{\varepsilon_n\}$ to satisfy the requirement of an acceptable budget? This question makes us rely on the average method again, that is, to compound the obtained budget sequence $\{\varepsilon_n\}$ with the average budget sequence $\{\varepsilon/n, \ldots, \varepsilon/n\}$. If the average budget is just enough to meet equation (27), then any of the other nonaverage allocation methods cannot be improved further. As long as the averaged budget is larger than the minimum budget that satisfies equation (27), we can adjust the budget sequence to satisfy the acceptable budget.

Let $k^*_{\min}$ be the minimum term of the proportion sequence $\{k^*_n\}$ that has been calibrated. Then, to satisfy the acceptable budget by compounding with the average budget sequence, we must have

$$\frac{1/n \times \alpha + k^*_{\min}}{\alpha + 1} \times \varepsilon \ge \frac{\sqrt{2}}{U_s}, \tag{28}$$

where α is a multiplicative factor. Thus, according to Lemma 1, we can make all the terms in the budget sequence satisfy the acceptable budget if we have

$$\alpha \ge \frac{\sqrt{2}/\varepsilon U_s - k^*_{\min}}{1/n - \sqrt{2}/\varepsilon U_s}. \tag{29}$$

After compounding, the actual assigned proportion k_i^{A*} is as follows:

$$k_i^{A*} = \frac{(\alpha/n + k_i^*)}{(\alpha + 1)},\ i \in N^+. \tag{30}$$

There is no need for calibration again because this optimization is performed based on the calibrated sequence. For those series methods that have been optimized to satisfy the acceptable budget, we add the prefix "AC-" to their names. Taking a Taylor series method as an example, its optimization is called the AC-Ty and is shown in Figure 7.

5.4. Useful Allocation Strategy. For a better evaluation of the usefulness of the queries, we learn from a well-known usefulness definition, which was proposed by Blum et al. [28] and could be used for measuring the utility of the released synthetic database in the noninteractive scheme. To make the definition more suitable for the scenarios and methods of

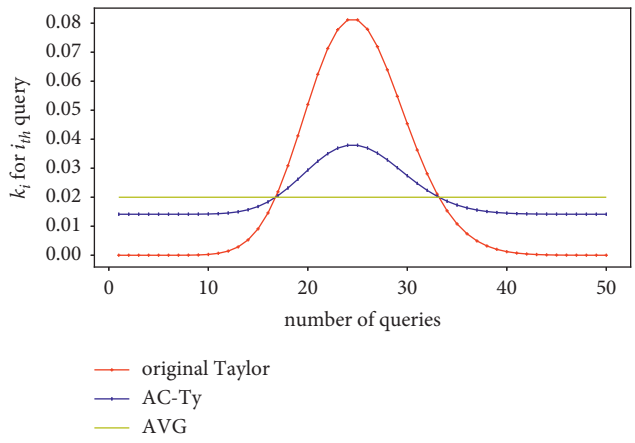

Figure 7: The AC-Ty series method with $U_s = 100$ and $n = 50$.

our work, we make a slight modification and obtain the following definition.

Definition 5 (useful strategy). Given a sequence of queries Q and the total amount of privacy budget ε, if for each query $q_i \in Q$, $i \le n$, then the normalized noise of q_i with strategy A is $N_A(i, \varepsilon)$, which satisfies the following equation:

$$\Pr\left[\max_{q_i \in Q, i \le n} \left|N_A(i, \varepsilon)\right| \le \gamma\right] \ge \delta. \tag{31}$$

Then, we consider the budget allocation strategy A to be $(\gamma, \delta, \varepsilon, n)-$ useful.

According to Definition 4, we can obtain the following lemma.

Lemma 4. *Given the total amount of privacy budget ε, the number of queries is n. If all the terms in budget sequence $\{\varepsilon_n\}$ with allocation strategy A satisfy the acceptable budget, then A is $(\sqrt{2}/\varepsilon_{\min}, 1 - e^{-\sqrt{2}}, \varepsilon, n)-$ useful, where $\varepsilon_{\min}$ is the minimum in the budget sequence.*

Proof. If strategy A satisfies the ideal query, then, for any $i \le n$, it must be subject to

$$\sqrt{2}\left(\frac{1}{\varepsilon_i}\right) \le \frac{\sqrt{2}}{\varepsilon_{\min}}. \tag{32}$$

Let $U_s = \sqrt{2}/\varepsilon_{\min}$; then, for any query in Q, the noise less than U_s has the following probability:

$$\delta \ge \int_{-U_s}^{U_s} \frac{\varepsilon_{\min}}{2} e^{-\varepsilon_{\min}|x|} dx = 1 - e^{-\varepsilon_{\min} U_s} \ge 1 - e^{-\sqrt{2}}. \tag{33}$$

The proof is complete. □

6. Protection for the Budget Sequence $\{\varepsilon_n\}$

In this section, we will discuss the collusion attack for the nonaverage strategies and our countermeasures.

6.1. Collusion Attack for Nonaverage Budget Strategies. We already know that one of the problems of the average allocation is security, and the series strategies seem to be able to resist this risk. However, let us imagine the following situation of a collusion attack. There are m users, each submits w queries, and the i_{th} query for everyone is the same. With the queries completed, they exchange information with each other, which will also lead to a breach of privacy. Since the series strategy is independent of specific queries and is only related to the number of queries, as long as the number of queries and the type of the series are known, the budget of the i_{th} query can be predicted in advance. Thus, for these m users, the same query has been conducted m times, and the worst is almost wm times if all the w queries are semantically the same. Therefore, we cannot say that the current series strategies must be safer than the average method. We need to protect the budget sequence.

6.2. Protect the Budget Sequence with a Random Arrangement. The simplest and most intuitive way is to randomize the budget sequence completely. Although this approach can guarantee the security, it also loses the characteristics and advantages of the series itself. So we are more willing to seek other approaches that could preserve these characteristics.

6.3. Protect the Budget Sequence with Probability. Although the budget sequence is a numeric type, if we only want to adjust the order of terms, then we can treat the terms as different options, and each option will be chosen with a certain probability. Similar to the sampling without replacement, we choose a budget term for a query according to the probability; we call this protection perturbation. There have been many comprehensive and mature results in the field of computer science regarding this topic. We only give a simple idea of determining the probability for each budget term in this article.

We score each budget term and give the probability based on the sum of all the scores. Assuming that $x - 1$ budget terms have been chosen for the queries, we are about to conduct the x_{th} query and choose a budget term for it; then, the remaining budget terms available for choosing are a total of $n - x + 1$. We represent the remaining terms as remaining sequence $\mathscr{E}_r$ and the score for the j_{th} term in $\mathscr{E}_r$ as q_j. If a budget term is sorted as i in the original budget sequence $\{\varepsilon_n\}$ and as j in $\mathscr{E}_r$, then the score q_j assigned to the j_{th} term in $\mathscr{E}_r$ could be

$$q_j = (N - i) \cdot e^{x-i}. \tag{34}$$

We believe that the smaller the number i is, the higher the score should be, and the more i is less than x, the higher the score is. Then the probability for the j_{th} term in $\mathscr{E}_r$ is as follows:

$$\Pr_j = \frac{q_j}{\sum_{l \in \mathrm{E}_r} q_l}. \tag{35}$$

The actual budget sequence after the perturbation is shown in Figure 8. We can see that the trend of the original series is still maintained.

Alternatively, we can construct a noisy budget sequence with noisy parameters.

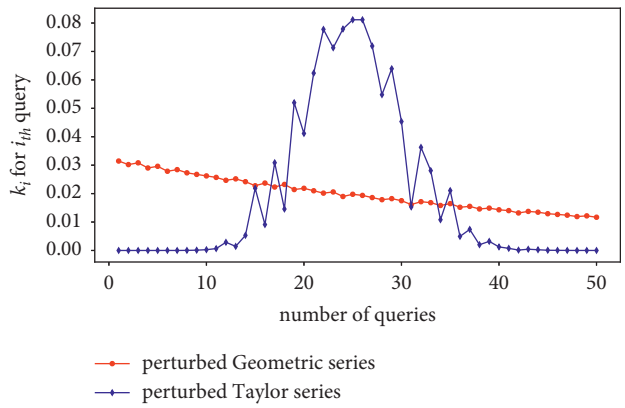

Figure 8: Comparison of two budget sequences after perturbation.

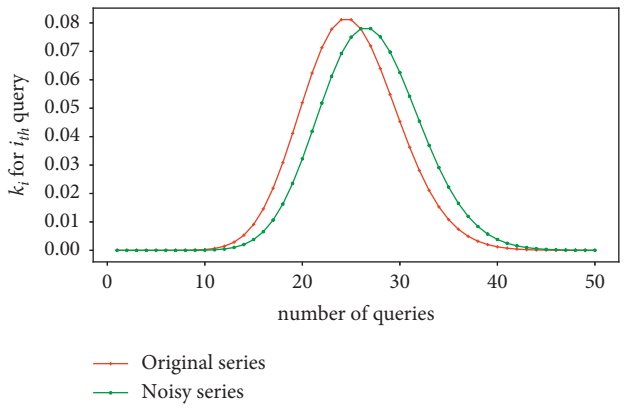

Figure 9: Comparison of the Taylor series with noisy n.

6.4. Protect the Budget Sequence with the Laplace Mechanism. Since the construction of a budget sequence depends on the number of queries n, we can apply differential privacy again to protect the budget sequence. Therefore, the third idea is to add the Laplace noise η to the number n according to Theorem 1 and make the budget sequence satisfy $\varepsilon-$ differential privacy. Noisy n is calculated as follows:

$$n_s = n + \eta, \quad \eta \sim \text{Lap}\left(\frac{1}{\varepsilon}\right), \tag{36}$$

where since the number n (of the query set) and the data set for queries are two disjoint sets, the privacy budget is still ε according to Theorem 3 (parallel composition).

It should be noted that the noise drawn from the Laplace distribution can be positive or negative. If the negative noise occurs, then n_s will be smaller than actual n, which may result in constructing a budget sequence that is too unbalanced and significantly affect the final results. Therefore, in the absence of the U_s constraint, we believe that positive noise should be added.

The Taylor series curve with $n_s = 53$ is as shown in Figure 9, and the difference from the original curve is obvious.

If we want to increase the difference between users, then we can assign different privacy budgets for different users.

7. Analysis and Experiment

In this section, we evaluate our strategies with two different sets of experiments. First, we compare the total amount of noise generated with different budget allocation strategies after multiple interactive queries. Second, to demonstrate that our series strategies have better adaptability and utility in practice than the average allocation, we apply our strategies to the k-means clustering and compare the accuracy of the noisy clustering results. For convenience, we use AVG, Geo, and Ty to represent the average strategy, the geometric series strategy, and the Taylor series strategy, respectively, in the graphs and tables.

7.1. Analysis of the Total Amount of Noise for Multiple Interactive Queries. We implement this experiment with the data set "Document" [29, 30]. "Document" is a database that contains 353,160 records describing the frequency of 3,430 politically relevant news and reviews on professional vocabulary. There are 6,906 professional vocabularies, and the data set contains three attributes: "article ID," "word ID," and "wordcount." The meaning of a record is counting the number of word IDs that appear in the article ID.

For comparison, we conduct N sum queries with each allocation method and calculate the total amount of noise, where N is 20, 50, and 100. Since the queries are of the same type and the first two attributes are indexes, we can scale the value domain of the "wordcount" to $[0, 1]$ to normalize the sensitivity (though it is a sum query, the semantics of "wordcount" require the value to be greater than 0). We compare the mean square error (MSE) of the overall noise. To avoid the extremums caused by random noise, we repeat l rounds ($l = 100$), and the mean square error of each round is denoted as $\text{MSE}_i\,(i = 1, \ldots, l)$. Then, we obtain the averaged mean square error $\overline{\text{MSE}}$. A large $\overline{\text{MSE}}$ indicates substantial noise and low accuracy of the query results. Let x_i be the true result of the i_{th} query, and let y_i be the noisy result with approach A. Then, we have the following:

$$\text{MSE}_i = \sum_{i=1}^{n} \left(y_i - x_i\right)^2 = \sum_{i=1}^{n} N_A\left(i, \varepsilon\right)^2, \overline{\text{MSE}} = \frac{1}{l}\sum_{i=1}^{l}\text{MSE}_i. \tag{37}$$

The total privacy budget ε in this experiment is set to 1 and U_s is 30, 95, and 200, which correspond to N values. We compare five conventional methods and the results are shown in Table 2.

It is noted that since the Taylor series strategy without considering U_s is certainly injecting much more noise into the queries at the early and late stages of the allocation than other methods, we default to optimize the Taylor series with $\alpha = 1$ in the experiments, and this method is expressed as "bl-Ty."

The results show that the total amount of noise generated by the geometric series strategy is very close to the average allocation, since a geometric series method can be relatively "balanced" by setting a large r. The Taylor strategy is inferior to the geometric strategy, but it still can have the same order of magnitude of noise. In addition, the methods after

Table 2: Comparison of the $\overline{\text{MSE}}$ of noise.

Strategies	$N = 20$ $U_s = 30$	$N = 50$ $U_s = 95$	$N = 100$ $U_s = 200$
AVG	$1.53E+04$	$2.56E+05$	$2.08E+06$
Geo	$2.14E+04$	$3.26E+05$	$2.54E+06$
bl-Ty	$3.03E+04$	$5.96E+05$	$5.26E+06$
AC-Geo	$1.63E+04$	$2.88E+05$	$2.20E+06$
AC-bl-Ty	$1.67E+04$	$3.00E+05$	$3.04E+06$

optimization can have less noise than the original ones. These results are consistent with our previous theoretical analysis.

From the perspective of noise, the average allocation has the least total amount of noise, while our series strategies can ensure that the total noise is always close to the average method and provides an uneven allocation for the budget.

7.2. Differentially Private k-Means Clustering with Different Strategies. Clustering is a very typical representative of differential privacy in data mining applications. A brief description of one iteration in differentially private clustering is as follows [11] (Algorithm 1).

By analysis, we can find the following: (1) since the clustering algorithm itself requires multiple iterations and to satisfy the guarantee of differential privacy, clusters usually cannot achieve a steady state (what we also call "converge" in this paper) after adding the noise; (2) within an iteration, the privacy budget still needs to be consumed many times [31]. Therefore, the differentially private clustering has been focused on by many researchers.

To compare the performances of different allocation strategies carefully, we apply the k-means clustering to two data sets, which are of different clustering features; and we will also use different metrics for comparative analysis.

7.2.1. Data Sets and Evaluation Methodology. The first data set we use is "Unbalance" [29], which is a small and labelled data set. It has 3 large clusters of 2000 points and 5 small clusters of 100 points (8 explicit clusters and 6500 2D points). The second data set we use is Brich3 [29], which has a large number of 100,000 2D random points, and the clusters are not explicit; we set $k = 20$ for the clusters. There are only two types of queries for data sets in the clustering task: one is sum and the other is count. Thus, we could scale all dimensions of the data set to $[-1, +1]$ in advance to satisfy the sensitivity normalization.

To evaluate the performance of clustering on the labelled data set, the commonly used method is the confusion matrix, which is used to calculate true positive (TP), true negative (TN), false positive (FP), and false negative (FN) for the clusters. To show our results more intuitively and comprehensively, we employ the F_1-score, which can be used for evaluating the clustering with labels. The definition of F_1 for cluster i is shown below:

$$\begin{aligned} \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ F_{1\_i} &= \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}. \end{aligned} \tag{38}$$

Because there are k clusters after partition and they are of different sizes, we consider giving them weights according to their size to obtain the overall F_1-score:

$$\overline{F_1} = \sum_{i=1}^{k} \frac{\left|O^i\right|}{|D|} \times F_{1\_i}, \tag{39}$$

where D is the given data set with k clusters and $O = \{O^1, O^2, \ldots, O^k\}$ represents the true clusters. To map each noisy cluster C^i to its corresponding true cluster O^j, we use the Kuhn – Munkres algorithm.

However, the F_1-score is not suitable for a random and unlabelled data set, since there might be more than one best partition after iterations. Therefore, we decide to compare the normalized intracluster variance (NIVC), as mentioned in [6]:

$$\text{NIVC} = \frac{1}{|D|} \sum_{i=1}^{k} \sum_{p^j \in O^i} \left\| p^j - o^i \right\|^2. \tag{40}$$

Here, o^i is the centroid of O^i and p^j is a point of O^i. The lower the NIVC is, the better the performance is.

7.2.2. Analysis of the Unbalance Data Set. Since the total number of iterations to achieve an optimal result of differential privacy clustering cannot be computed, we trace the clustering results of each method with different total number of iterations and repeat 100 times to compare the averaged F_1-score. Meanwhile, to decide a maximum total number of iterations for comparison, we use the method proposed by Su et al. [6], according to which the maximum number of iterations could be set as 14 (note that Su et al. used the average strategy).

The value of the privacy budget has always been a major problem in differential privacy [32]. Although some researchers have discussed the upper or lower limit of the privacy budget, it has not been widely approved yet. Taking the total number of iterations and queries into account, and with a lot of experiments, we found that a budget of about 0.3 can get good clustering results and reflect the gap between different allocation methods as well; therefore, we set the total budget as $\varepsilon = 0.3$.

We divide our comparison into three groups, which correspond to different strategies, and all treat the average allocation curve as a benchmark.

(1) Comparison of the Performance of Geometric Strategy. The first comparison is of geometric strategy, which includes 6

(1) Partition the points into k clusters with a set of centroids $u_1, \ldots, u_k$
(2) **for** each of the cluster C_i, $i \in 1, \ldots, k$ **do**
(3) Count the noisy number of points in C_i as $\overline{c_i}$
(4) Compute the noisy sum of points in C_i as $\overline{m_i}$
(5) Update a new centroid $\overline{u_i} = \overline{m_i}/\overline{c_i}$.
(6) **end for**

Algorithm 1: One iteration of differentially private k-means clustering.

kinds of different geometric methods. Among them, the bisection method is a fixed series allocation method, the Comrade method is only used for decision trees, and the others are general and flexible geometric methods proposed by us. In order to be able to make a fair comparison, we change the tree height h in the Comrade method to the total number of clustering iterations N, and i is the number of the current iteration; namely, $\varepsilon_i = 2^{(N-i)/3} \times \varepsilon \times (2^{1/3} - 1)/(2^{(N+1)/3} - 1)$. Meanwhile, to ensure that all budgets are exhausted, we have calibrated all the methods.

According to Lemma 3, the boundary of U_s is different with different numbers of iterations when optimizing the series strategies. The boundary of U_s calculated for different numbers of iterations is shown in Table 3, and the actual values of U_s that we set are roughly in the middle of the interval. The comparison of the F_1-score for the geometric strategy is shown in Figure 10. Affected by the probability, the curve fluctuates slightly, but it does not affect the overall variation trend.

From Figure 10, we can observe the following phenomena: Firstly, although both the bisection method and the Cormode method belong to series strategy, their results are obviously not as effective as the series methods proposed in this paper (readers should pay attention to the scale of the y-axis in two subfigures). Secondly, among the methods designed according to the series strategy proposed in this paper, two monotonically increasing allocation methods, the Flip-Geo and the AC-Flip-Geo, are significantly better than all the other methods; and although the monotonically decreasing allocation methods are better than the bisection method and the Cormode method, they are inferior to the average allocation especially when the total number of iterations is larger than 8. Thirdly, the best performing method is the Flip-Geo instead of the AC-Flip-Geo when the total number of iterations is larger than 10. Fourthly, when the total number of iterations is less than 8, the monotonically decreasing method AC-Geo is better than the average method. Finally, all methods can reach the highest level of performance when the total number of iterations is around 5, except for the bisection method.

Based on these results, we can analyze several conclusions. First of all, not only is the series strategy proposed in this paper flexible and variable but also it guarantees the utility of output of an algorithm; even the monotonically decreasing methods are only slightly worse than the average method. Second, in differentially private clustering, monotonically increasing allocation methods are more practical. It means that, in the differentially private algorithms in which queries are associated with each other, the total amount of noise is not the only factor that determines the final output, and budget allocation strategy will also have a significant impact. We can see that when the total number of iterations is less than 10, the method AC-Flip-Geo is the best performing one; and when the total number of iterations is less than 6, the monotonically decreasing AC-Geo is also better than the average method. All these show that both the total amount of noise and the allocation strategy make an impact on the output; and this is also the reason why the method Flip-Geo has larger noise than the AC-Flip-Geo but it gradually shows an absolute advantage as the number of iterations increases. Last, the actual number of iterations required for clustering may be very small, and the choice of allocation strategy will affect the number of iterations that can be supported.

Obtaining such result is inseparable from the monotonicity of the geometric allocation strategy itself and the requirements of convergence for clustering. Although the clustering algorithm depends on the result of the previous iteration, in order to converge, the accuracy of the clustering should be gradually improved; otherwise, it will be difficult to converge. Under this demand, two flipped methods for allocation are very prominent.

(2) Comparison of the Performance of the Taylor Strategy. For the same reason mentioned in Section 7.1, we use the bl-Ty instead of the original Taylor method in this group of comparisons. Since the Flip-Ty is very close to the average method, the AC-Flip-Ty constrained by U_s is much closer to the average method. Thus, a comparison of the method AC-Flip-Ty and the average method is meaningless. As a result, this group of comparisons only includes four methods: AVG, bl-Ty, AC-bl-Ty, and Flip-Ty. The boundary of U_s for different numbers of iterations is shown in Table 4, and the comparison of the F_1-score for the Taylor methods is shown in Figure 11.

We can see from Figure 11 that the results are similar to those from the geometric series. Just as what we have concluded in the experiment of geometric series, allocating low budget terms to the later iterations is not conducive to the convergence of clustering. Thus, although both bl-Ty and the AC-bl-Ty are nonmonotonic allocation methods, they perform poorly. This is due to the fact that their curves are similar to convex functions, and most of the budget terms in the allocation sequence are very low, especially at the early and late stages of the allocation, while for the Flip-Ty, though

Table 3: Boundaries and values for U_s of geometric strategy with $\varepsilon = 0.3$.

Total iterations ($\varepsilon = 0.3$)	Boundary of U_s	Actual U_s is set
$N = 3$	(14.15, 22.40)	18
$N = 4$	(18.86, 30.56)	25
$N = 5$	(23.58, 38.70)	30
$N = 6$	(28.29, 46.81)	35
$N = 7$	(33.00, 54.93)	45
$N = 8$	(37.72, 63.04)	50
$N = 9$	(42.43, 71.14)	55
$N = 10$	(47.15, 79.30)	65
$N = 11$	(51.86, 87.36)	70
$N = 12$	(56.57, 95.46)	75
$N = 13$	(61.29, 103.56)	82
$N = 14$	(66.00, 111.67)	90

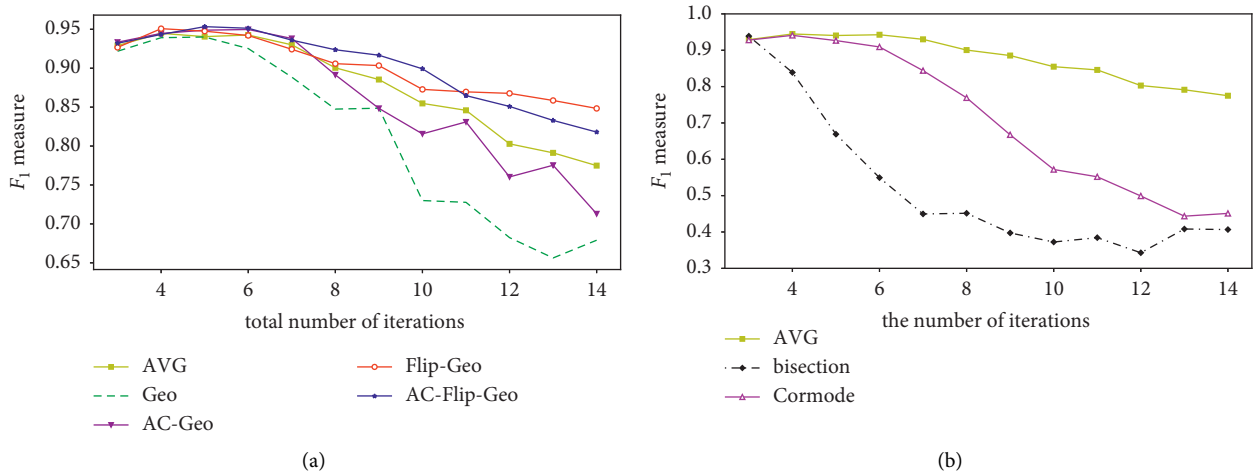

Figure 10: Comparison of the geometric methods in k-means clustering.

Table 4: Boundaries and values for U_s of Taylor strategy with $\varepsilon = 0.3$.

Total iterations ($\varepsilon = 0.3$)	Boundary of U_s	Actual U_s is set
$N = 3$	(14.15, 17.67)	16
$N = 4$	(18.86, 24.53)	21
$N = 5$	(23.58, 31.93)	27
$N = 6$	(28.29, 39.88)	34
$N = 7$	(33.00, 48.50)	41
$N = 8$	(37.72, 60.42)	48
$N = 9$	(42.43, 72.62)	57
$N = 10$	(47.15, 84.70)	65
$N = 11$	(51.86, 96.46)	75
$N = 12$	(56.57, 107.79)	80
$N = 13$	(61.29, 118.70)	90
$N = 14$	(66.00, 129.25)	98

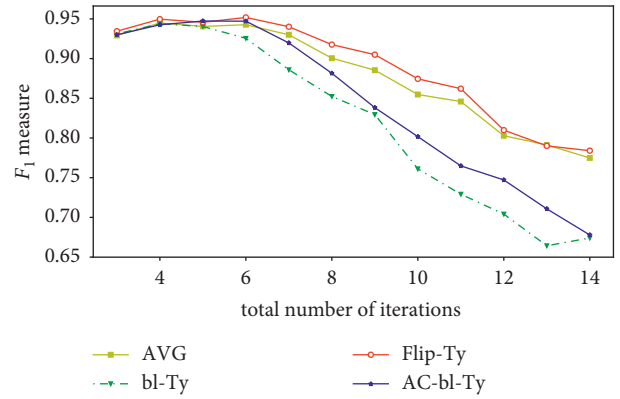

Figure 11: Comparison of the Taylor methods in k-means clustering.

it is close to the average strategy, it has the ability to maintain a higher budget allocation at the early and late stages; thus its performance is the best among this group of methods. Although it is only slightly higher, it is very effective. At the same time, we also noticed that as the number of iterations increases, the performance of the method Flip-Ty gradually tends to the average method.

(3) Comparison of the Performance of the Strategy with Different Protections. In Figure 12, we take the Flip-Geo as representative and compare its performance with two different protections. We can see that the performances of the two Flip-Geo methods with different protections are still better than the average allocation; they all maintain a high

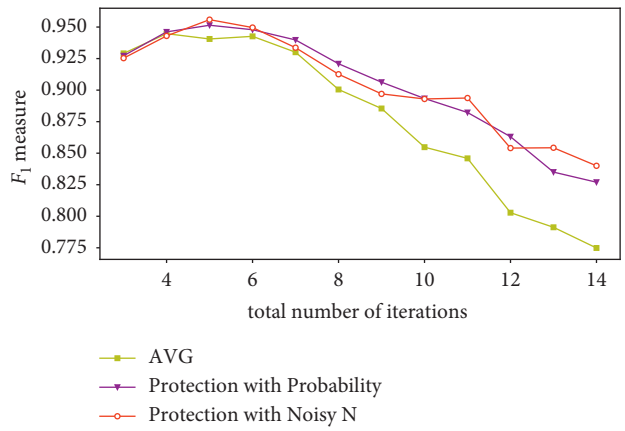

Figure 12: Comparison of the geometric methods with different protections in k-means clustering.

level of accuracy. Relatively speaking, the protection with noisy N is more stable than the protection with probability. It shows that the budget sequence determined by the series strategies in this paper could still obtain higher performance than the average method even if it has been disturbed or not been allocated according to the original sequence to some extent.

To sum up, with the total budget unchanged, both two series strategies can achieve a better performance than the average strategy regardless of whether the total number of iterations is small or large. Furthermore, we fully believe that it is not the allocation strategy with the least amount of noise which brings good results. It is necessary for us to choose the appropriate allocation strategy according to the specific applications.

7.2.3. Analysis of the Brich3 Data Set. Although our series strategies show obvious advantages on well-clustered data sets, we are curious to know whether they still perform well and support more iterations on complex, high-density, large data sets. We select three methods that performed well with the "Unbalance" data set for comparison in this experiment. This time we will conduct more iterations. Similar to the operation of the previous data set, we calculated the maximum number of iterations as 47 according to Su et al.'s method in literature [6], and we separately compare the clustering results of different allocation methods after $N = \{5, 10, 15, 20, 25, 30, 35, 40, 45\}$ iterations. Considering the complexity of the data set and the instability of the results, we repeat each method 1000 times to avoid some extreme cases and compare the averaged clustering results of each method at different total iterations. The boundaries and the actual setting values of U_s are shown in Table 5; and, in this experiment, we set budget ε to 0.5.

Recall the NIVC introduced in 7.2.1; the smaller the value is, the better the clustering result is. In addition, since each method repeats the experiment up to 1000 times, we also analyzed the standard deviation between these 1000 results.

It can be seen from Figures 13 and 14 that, even in complex data sets which are with more iterations, our methods are still better than the average strategy, which is consistent with the results of our previous set of experiments. We certainly also observe something new.

Table 5: Boundaries and values for U_s in Brich3 with $\varepsilon = 0.5$.

Total iterations ($\varepsilon = 0.5$)	Boundary of U_s	Actual U_s is set
$N = 5$	(14.15, 23.21)	18
$N = 10$	(28.29, 47.55)	35
$N = 15$	(42.43, 71.86)	55
$N = 20$	(56.57, 96.16)	75
$N = 25$	(70.72, 120.47)	95
$N = 30$	(84.86, 144.77)	115
$N = 35$	(99.00, 169.07)	135
$N = 40$	(113.14, 193.37)	155
$N = 45$	(127.28, 217.67)	170

First, as the total number of iterations increases, the stability of the final result is broadly in line with the budget terms allocated to the last few iterations, but it has nothing to do with the performance. We can see that although the standard deviation of 1000 results of the Flip-Ty is almost the same as the average method, the performance is significantly better and sometimes even better than the AC-Flip-Geo. The reason for the above phenomenon is not complicated. One of the reasons we believe is that, in addition to the last budget terms and the total amount of noise, the factors affecting the clustering should also include the direction of the noise. In the clustering algorithm, we usually regard the data that needs to be clustered as points in a space; it means that the noise we add to each centroid has direction. Although a low budget usually implies noise of large magnitude, if we take the noise direction into consideration, a large amount of noise can also have a positive effect, such as correcting the deviation caused by the previous noise. Therefore, the method Flip-Ty which allocates some low budget terms during the allocation process may not cause a negative impact.

Second, when the total number of iterations exceeds a certain number, the gaps between four methods will gradually get close. The reason is simple. The gaps between series strategies and average strategy will increase with the total number of iterations, which is the advantage of our series strategies. However, as the number of iterations increases further, any kind of strategy will gradually tend to be average, which is inevitable. Therefore, we find that even the geometric strategy will also approach the average method gradually as the number of iterations increases. However, this does not mean that our series strategies are not as good as the average strategy, since the experiments on two data sets have shown that, within the maximum total number of iterations, our series strategies are better than average strategy all the time.

Third, the complexity of this data set is very close to the real one, and, in such a data set, we still find that the number of iterations to achieve the best performance is actually much smaller than we thought. This implies that although the differential privacy mechanism brings noise, it does not mean that the original algorithm needs to become more complicated. Of course, it should be emphasized that this is only for clustering.

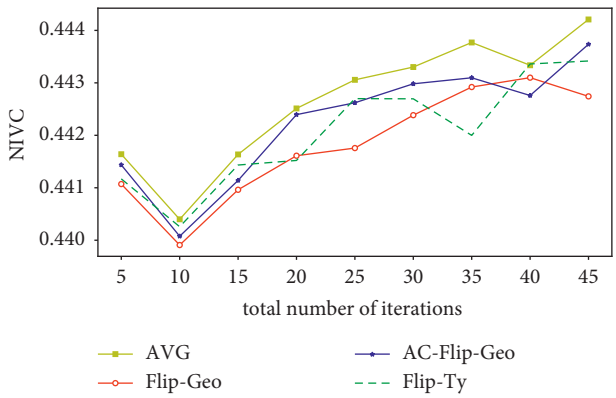

Figure 13: Comparison of the performance in Brich3.

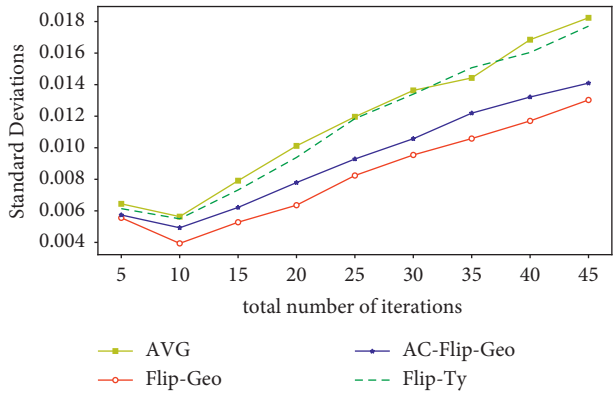

Figure 14: Standard deviation of the each method.

Therefore, it can be ensured that, on the premise of satisfying the clustering performance requirements, our series strategies can be sufficiently better than the average allocation and can support more iterations meanwhile.

8. Conclusion

From the perspective of monotonicity and convergence, we studied two series strategies for privacy budget allocation in this paper: the geometric series strategy and the Taylor series strategy. In order to quickly find out a series to allocate the privacy budget efficiently to satisfy the actual needs in practice, we provided a simple calculation method to determine the key parameters for two series strategies: the ratio r and the initial term t. Our series strategies have a high level of flexibility and adaptability, which is reflected in that the budget sequence can easily change the monotonicity to meet the application requirements and set the minimum acceptable budget to adjust the consumption speed of the budget. To further improve security, we briefly explored protection for the budget sequence. Several sets of experiments have fully shown the effectiveness of our strategies and verified that the choice of budget allocation method will obviously affect the utility of the final output especially for the field of data mining and machine learning. Taking clustering as an example, our series budget allocation method can achieve higher accuracy compared to other existing methods and can also support much more iterations. We believe that the series strategy can certainly become a very effective and convenient tool for budget allocation.

Of course, we also find some interesting problems during the experiment. For example, at present, we add noise to the centroid of each cluster to obtain the next clustering results, but, in fact, adding noise to any one centroid affects all clusters partitioned more or less in the next iteration. Thus, does the existing method add too much unnecessary noise? In addition, even if the sensitivities of two query functions are both 1, due to the different ranges of results, the tolerance of the two query functions to noise is different. For example, for the count query with range $[0, +\infty)$ and the max query with $[0, 1]$, the count function can tolerate more noise obviously. How to deal with it more reasonably will be our future work.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grant nos. 61972209, 61572263, 61502251, 61502243, and 61602263 and in part by China Postdoctoral Science Foundation under Grant 2016M601859 and Grant 2015M581794.

References

[1] J. Zhang, X. Liang, and Z. Zhang, "Re-DPoctor: real-time health data releasing with w-day differential privacy," in *Proceedings of the GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, Singapore, December 2017.

[2] L. Qi, C. Hu, X. Zhang, M. R. Khosravi, and S. Sharma, "Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4159–4167, 2021.

[3] L. Qi, X. Wang, X. Xu, W. Dou, and S. Li, "Privacy-Aware cross-platform service recommendation based on enhanced locality-sensitive hashing," *IEEE Transactions on Network Science and Engineering*, vol. 8, p. 1, 2020.

[4] C. Dwork, F. McSherry, and K. Nissim, "Calibrating noise to sensitivity in private data analysis," *Journal of Privacy and Confidentiality*, vol. 7, no. 3, pp. 17–51, 2017.

[5] A. Friedman and A. Schuster, "Data ming with differential privacy," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*, pp. 493–502, Washington, DC, USA, July 2010.

[6] D. Su, J. Cao, N. Li, E. Bertino, M. Lyu, and H. Jin, "Differentially private k-means clustering and a hybrid approach to private optimization," *ACM Transactions on Privacy and Security*, vol. 20, no. 4, pp. 1–33, 2017.

[7] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin, "Differentially private k-means clustering," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy (CODASPY'16)*, pp. 26–37, Orleans, LA, USA, March 2016.

[8] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A practical differentially private random decision tree classifier," in *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, pp. 114–121, Miami, FL, USA, December 2009.

[9] H. Shin, S. Kim, J. Shin, and X. Xiao, "Privacy enhanced matrix factorization for recommendation with local differential privacy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1770–1782, 2018.

[10] X. Cheng, S. Su, S. Xu, and Z. Li, "DP-Apriori: a differentially private frequent itemset mining algorithm based on transaction splitting," *Computers and Security*, vol. 50, no. 9, pp. 74–90, 2015.

[11] C. Dwork, "Differential privacy," in *Proceedings of the 33rd International Conference on Automata, Languages and Programming (ICALP)*, pp. 1–12, Venice, Italy, July 2006.

[12] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu, "Differentially private spatial decompositions," in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pp. 1–12, Arlington, VA, USA, April 2012.

[13] C. Dwork, "A firm foundation for private data analysis," *Communications of the ACM*, vol. 51, no. 1, pp. 86–95, 2011.

[14] F. McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," *Communications of the ACM*, vol. 53, no. 9, pp. 89–97, 2010.

[15] J. Xu, Z. Zhang, X. Xiao, Y. Yang, G. Yu, and M. Winslett, "Differentially private histogram publication," *The VLDB Journal-The International Journal on Very Large Data Bases*, vol. 22, no. 6, pp. 797–822, 2013.

[16] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 8, pp. 1200–1214, 2011.

[17] R. Chen, B. Fung, B. Desai, and N. M. Sossou, "Differentially private transit data publication: a case study on the Montreal transportation system," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 213–221, Beijing, China, August 2012.

[18] W. Qardaji, W. Yang, and N. Li, "Understanding hierarchical methods for differentially private histograms," *Proceedings of the VLDB Endowment*, vol. 6, no. 14, pp. 1954–1965, 2013.

[19] S. Rana, S. K. Gupta, and S. Venkatesh, "Differentially private random forest with high utility," in *Proceedings of the 2015 IEEE International Conference on Data Mining*, pp. 955–960, Atlantic City, NJ, USA, November 2015.

[20] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta, "Discovering frequent patterns in sensitive data," in *Proceedings of the KDD'10 Proceedings of the 16th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pp. 503–512, Washington, DC, USA, July 2010.

[21] J. Hua, Y. Gao, and S. Zhong, "Differentially private publication of general time-serial trajectory data," in *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 549–557, Hong Kong, China, May 2015.

[22] Y. Wu, Y. Wu, H. Peng, J. Zeng, H. Chen, and C. Li, "Differentially private density estimation via Gaussian mixtures model," in *Proceedings of the 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pp. 1–6, Beijing, China, June 2016.

[23] N. Li, W. Qardaji, D. Su, and J. Cao, "PrivBasis: frequent itemset mining with differential privacy," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1340–1351, 2012.

[24] Z. Fan and X. Xu, "APDPk-means: a new differential privacy clustering algorithm based on arithmetic progression privacy budget allocation," in *Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems*, pp. 1737–1742, Zhangjiajie, China, August 2019.

[25] A. Bkakria, A. Tasidou, N. Cuppens-Boulahia, and F. Cuppens, "Optimal distribution of privacy budget in differential privacy," in *Proceedings of the International Conference on Risks and Security of Internet and Systems*, pp. 142–163, Arcachon, France, October 2018.

[26] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of the Theory of Cryptography Conference*, pp. 265–284, New York, NY, USA, March 2006.

[27] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of the PODS'03 Twenty-Second ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems*, pp. 202–210, San Diego, CA, USA, June 2003.

[28] A. Blum, K. Ligett, and A. Roth, "A learning theory approach to non-interactive database privacy," in *Proceedings of the STOC'08 Fourth Annual ACM Symposium on Theory of Computing*, pp. 609–618, New York, NY, USA, May 2008.

[29] P. Fränti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Applied Intelligence*, vol. 48, no. 12, pp. 4743–4759, 2018.

[30] D. Newman, "UCI machine learning repository," 2008, https://archive.ics.uci.edu/ml/datasets/Bag+of+Words.

[31] T. Zhu, G. Li, W. Zhou, and P. S. Yu, "Differentially private data publishing and analysis: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1619–1638, 2017.

[32] C. Dwork and S. Adam, "Differential privacy for statistics: what we know and what we want to learn," *Journal of Privacy and Confidentiality*, vol. 1, no. 2, pp. 1–16, 2010.