IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 7, 2021, accepted March 26, 2021, date of publication April 8, 2021, date of current version April 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3071794

Detecting and Recovering Integer Data Manipulated by Multiplication With a Nonintegral Real Number and a Rounding Operation

TAEJUNG PARK[1], HYUNJOO SONG[2], AND SANG JUNE LEE[3]

[1]Department of Cyber Security/IT Media, Duksung Women's University, Seoul 01369, South Korea
[2]School of Computer Science and Engineering, Soongsil University, Seoul 07027, South Korea
[3]Department of Mathematics, Kyung Hee University, Seoul 02447, South Korea

Corresponding author: Sang June Lee (sjlee242@khu.ac.kr)

The work of Taejung Park was supported by the Korea Electric Power Corporation under Grant R18XA01. The work of Sang June Lee was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Education through the Basic Science Research Program under Grant NRF-2019R1F1A1058860.

ABSTRACT This paper presents a method for detecting and restoring integer datasets that have been manipulated by operations involving nonintegral real-number multiplication and rounding. As we discuss in the paper, detecting and restoring such manipulated integer datasets is not straightforward, nor are there any known solutions. We introduce the manipulation process, which was motivated by an actual case of fraud, and survey several areas of literature dealing with the possibility that manipulation may have happened or might occur. From our mathematical analysis of the manipulation process, we can prove that the nonintegral real number (α) used in the multiplication exists not as a single real number but as an interval containing infinitely many real numbers, any of which could have been used to produce the same manipulation result. Based on these analytic findings, we provide an algorithm that can detect and restore manipulated integer datasets. To validate our algorithm, we applied it to 40,000 test datasets that were randomly generated using controllable parameters that matched the real fraud case. Our results indicated that the algorithm detected and perfectly restored all datasets for which the value of the nonintegral real number was at least 16 ($\alpha \geq 16$) and the number of data entries was at least 40 ($n \geq 40$).

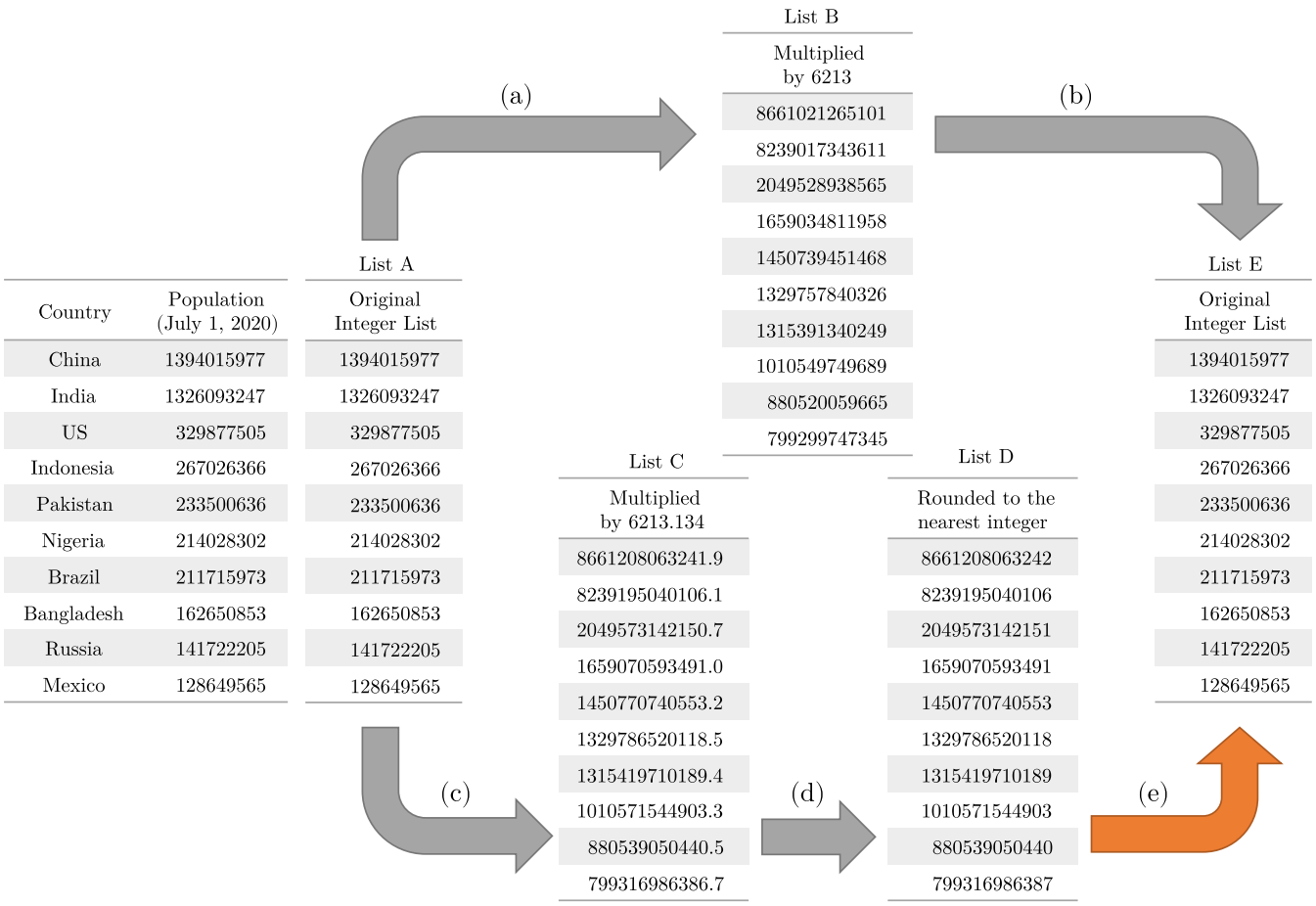
INDEX TERMS Data fraud, detection algorithm, integer data manipulation, interval operations.

I. INTRODUCTION

Since the advent of data-driven technologies, including big data and artificial intelligence, the importance of data security has only increased. In large part, this is because many social and industrial systems use statistical and other types of data to make decisions or to automate their processes. In such an automated system, there can be security risks caused by manipulation by malicious agents of the data used in the system. Such security breaches can cause multiple unacceptable problems for our communities and industries, including corruption, unfairness, and lack of public confidence in our existing social and technological systems.

Among various types of data, the ''integer list'' type is commonly found in commercial and political surveys, vote counts among candidates, academic data, and demographic data such as data based on age, gender, or region. We define an integer list as a set of integer numbers where each integer number represents a certain quantity of each component of the list. As illustrated in Figure 1, columns of tabular data often comprise integer lists. We can easily extract an integer list (e.g., List A) from a general table format (e.g., the two columns to the left of List A). In an integer list, integer entries often have different values, as shown in the figure. This integer list type of data provides policymakers or automated systems with critical information for determining the next plan of action. For example, national policy makers distribute government budgets to each region according to demographic data (e.g., population for each region), marketers require C-level managers to allocate marketing funds based on consumer surveys or revenue for each segment, and television show producers negotiate prices with advertisers based on their relative popularity (i.e., the number of viewers of each show).

The associate editor coordinating the review of this manuscript and approving it for publication was Vlad Diaconita.



Country	Population (July 1, 2020)
China	1394015977
India	1326093247
US	329877505
Indonesia	267026366
Pakistan	233500636
Nigeria	214028302
Brazil	211715973
Bangladesh	162650853
Russia	141722205
Mexico	128649565

List A
Original Integer List
1394015977
1326093247
329877505
267026366
233500636
214028302
211715973
162650853
141722205
128649565

List B
Multiplied by 6213
8661021265101
8239017343611
2049528938565
1659034811958
1450739451468
1329757840326
1315391340249
1010549749689
880520059665
799299747345

List C
Multiplied by 6213.134
8661208063241.9
8239195040106.1
2049573142150.7
1659070593491.0
1450770740553.2
1329786520118.5
1315419710189.4
1010571544903.3
880539050440.5
799316986386.7

List D
Rounded to the nearest integer
8661208063242
8239195040106
2049573142151
1659070593491
1450770740553
1329786520118
1315419710189
1010571544903
880539050440
799316986387

List E
Original Integer List
1394015977
1326093247
329877505
267026366
233500636
214028302
211715973
162650853
141722205
128649565

FIGURE 1. The populations of ten countries multiplied by an integer and by a real number (a) multiplied by an integer 6213 (b) easy to restore the original integer list by calculating the GCD (i.e., 6213) (c) multiplied by a real number 6213.134 (d) applied rounding operation to make values integer numbers (e) NOT EASY to restore!! - the GCD is 1.

An interesting phenomenon accompanying the prevailing platform economy [1] is that platform businesses consider the total number of users or subscribers more important in competitive terms because a larger user population implies more power in business. This aspect is often more meaningful to businesses than the order or rank of each entry. As a result, it can lead to inappropriate attempts at overestimating critical data in an integer list. One approach to manipulation would be to multiply every list entry by an integer number to give an overestimation that would maintain the same ratios between the entries. However, this would not easily deceive the public or auditors because it is a simple linear operation: the multiplication factor is discoverable by calculating the greatest common divisor (GCD). For example, the original integer list (List A) in Figure 1 could be manipulated by multiplication with an integer (6213) to generate the manipulated list (List B in Figure 1 (a)). We can detect that List B has probably been manipulated by calculating the GCD (6213) of the integer entries without needing to know the original data in List A.

A more sophisticated approach would be to multiply List A by a real number such as 6213.134, resulting in List C in Figure 1 (c). List C could then be rounded to make all entries integer values (List D in Figure 1 (d)). In this case, the GCD is 1 for List D and we cannot use this method to determine if List D has been manipulated. Despite the manipulation operation presented in Figure 1 being rather simple and straightforward, its reversal is neither trivial nor simple.

In an actual case, a television producer was alleged to have manipulated vote counts collected from live auditions by multiplying each vote count by a large real number (around 7494.442) and then rounding the values [2]. The intention was suspected to involve exaggerating the vote counts that directly indicated the popularity of the show. Again, this was because the popularity of any television show often determines the earnings (e.g., advertisement income) for that show. Because the rounding calculation on real numbers is a nonlinear operation, it is neither straightforward to recognize that the integer list has been manipulated by such an operation nor

easy to reconstruct the original data. Although this particular case is under investigation following great public attention, we can reasonably suspect that there are or will be similar cases that are undetected and are causing hidden damage somewhere or other.

Motivated by this scandal, we explored two issues. First, we wondered if there had been other cases involving similar types of manipulation. Second, we were curious about the mathematical and computational techniques used to detect the manipulation and/or recover the original data.

To address the first issue, we investigated several possible areas, including research misconduct, management and accounting, national statistical data, and election data. During the investigation, we learned that the intentions and values that could lead to data manipulation differed between the various areas. However, we found very few cases in each area where the manipulation described in Figures 1 (d) and 1 (e) had occurred. This observation has two implications. First, we can assume that there have been few such data manipulation cases in each area to date. Second, we can also assume that there may well have been undetected cases, because detection is difficult in the absence of reliable methods for detecting and restoring such manipulated data. Considering that our world will be increasingly dependent on data in the future, the first assumption—that there might have been no such cases—cannot justify that there is and will be no need to detect and recover from the type of data manipulation illustrated in Figure 1. In addition to practical considerations, we believe that any mathematical analyses and computational algorithms that address this issue will be valuable per se. With respect to the second assumption, we have found several possible reasons for the existence of only a few reports on the details of numerical manipulations in the literature. One is that many data manipulation cases involved ''abuse by authorities'' and were exposed by competitors, political successors, or external auditors who had relatively few opportunities to access the internal resources or details of the data manipulators [3]–[5]. For example, politicians or senior managers have been known to press subordinate staff in national statistical offices or finance departments to manipulate numerical data, aiming to fulfill unrealistic goals. The manipulation would remain concealed until political successors or competitors, who often would know little of the details, would publicize the issue. During any forensic process, most of the staff members who implemented the manipulation following some kind of incentive [6], [7] would tend to keep quiet to avoid any liability or penalty. Because of this, most forensic work has focused on the macro aspects of the data manipulation, such as statistical inconsistency among different datasets gathered by independent parties [3], [6]. In summary, although the details of the manipulation involve operations at the micro level, investigation and forensic tasks often focus on macro levels. Therefore, we will need sophisticated micro-level methods for detecting and reconstructing manipulated numerical data to avoid unnoticed cases. Most digit-based test algorithms to detect data manipulation at the micro level check the distribution of individual numeral digits. For example, some studies seek to spot fraud in numerical data based on experimental results showing that human beings are not good at generating random numbers [8]. One well-known method to detect numerical manipulation is Benford's law [9]–[11]. This law is based on the fact that the frequency of appearance in any data of the digit ''1'' is around 30%, ''2'' around 17.6%, and so on. However, Benford's law does not work well when the number of data entries is less than 50. Moreover, the data should contain all numbers from ''1'' to ''9'' as the leading digit, which makes this method less applicable to many datasets. (For example, Benford's law is not suited to a dataset involving the average height of an adult man in centimeters because most leading digits would be ''1'' or ''2''). Finally, this method has not been proved mathematically [11]. Unlike these methods, the method described in this paper deals with manipulation processes that are more specific, presents mathematical proofs and analyses of its validity, and is implemented as a practical algorithm for detecting and recovering from the specific manipulation depicted in Figures 1 (c) and 1 (d).

The second issue we explored can be summarized by the following questions: given only the manipulated integer data (e.g., List D in Figure 1), is it possible to determine whether the integer dataset has been manipulated, with a high degree of confidence? If so, is it also possible to determine the exact real number, containing nonzero decimal digits, which was used in the multiplication, thereby reconstructing the original data? At first glance, this problem might seem easy because the original manipulation process was simple. However, to our surprise, reversing the process is not at all easy, nor has our survey found a strictly proven method that addresses this issue.

In this paper, we formalize and analyze the issue in mathematical terms, aiming to address this specific manipulation problem. Based on our analysis and findings, we also propose an algorithm that detects if any integer list data has been manipulated by multiplication with a real number and rounding. For any such manipulated integer list, it finds the set of possible real numbers used in the multiplication and thereby reconstructs the original integer list data. (Note that the malicious manipulator might have used just one real number in the manipulation, but our method finds a (narrow) real interval containing the number(s) used.)

We believe it is necessary to categorize the various types of numerical data manipulation in a more systematic way, not only from a macro perspective but also from a micro perspective. However, such a categorization is beyond the scope of this paper, where we focus on the specific type of data manipulation depicted in Figure 1.

To the best of our knowledge, our approach is the first that analyzes and solves this problem thoroughly. We believe that our algorithm will help prevent the type of manipulation addressed, acting effectively as an ''automated auditor''.

II. RELATED WORK

A. NUMERIC DATA MANIPULATION

1) NATIONAL STATISTICS

Politicians are often tempted to manipulate sensitive national statistical data—including but not limited to national income, unemployment, and death rates from pandemic disease—for their own political interests. Wallace [6] indicates that a hierarchical structure in an authoritarian country is vulnerable to manipulation of sensitive national data at the lower levels of the hierarchy. The central authority and subnational leaders (or lower-level officials) would be associated with the principal–agent problem [4], [5], which makes it difficult to uncover details of data manipulation at lower levels, such as that in Figure 1.

Aragão and Linsi [3] analyze the relationship between political interests and statistical manipulation in various countries and categorize government data manipulation into four types. They report the details of three manipulation cases involving macroeconomic data by interviewing former government officers and applying statistical forensics. However, they focus only on the macro perspective of the data manipulation and not on low-level details of the data manipulation. Among their category types, Type 2—politically convenient guesstimating—is often found in underdeveloped countries where the government has limited statistical resources.

Jerven [12] reports that some important macroeconomic indicators are often roughly fabricated or projected based on partial data simply because accurate data do not exist in some underdeveloped countries. He describes a data manipulation case to fabricate food production data by multiplying the estimated farming population with an estimated intake of calories per capita, which is similar to the manipulation process we discuss in Figure 1 (see Chapter 1 in [12]). Although unreliable and manipulated national data in such underdeveloped countries may have different implications from those in other cases (e.g., outright and malicious manipulation of financial data for listed companies in developed economies), we believe that the method presented in this paper would help to obtain accurate conclusions from the available public data in some cases.

2) MANAGEMENT AND ACCOUNTING

In management and accounting, data manipulation has a more direct connection with illegal economic interests. Considering its potentially huge damage to the economy and to society, any attempts to manipulate relevant data should be detected and prohibited. Unfortunately, the common structure of many modern corporations—where management and ownership are separated—often makes it vulnerable to data manipulation. According to the principal–agent theory [4], [5], [13], agents (i.e., managers) can more easily access important data than principals (e.g., shareholders) in general. Because of this asymmetric distribution of information among agents and principals, agents are often tempted to manipulate management and accounting data. As we discussed in Section 2.1.1, the principal-agent problem makes it hard to discover data manipulations at lower levels.

Young and Sherman [14] advocate the need for new analytical tools to prevent accounting fraud via practical methods. One such practical tool, Benford's law, is commonly utilized in financial markets. For example, Grammatikos and Papanikolaou [10] applied Benford's law to detect accounting data manipulation in the banking industry [10]. However, as Collins [11] indicates, Benford's law cannot be applied to every case because of its well-known limitations.

Considering that there are only a few analytical tools to detect manipulation at micro levels, we need tools that are more specific to address separately the various types of data manipulation. In particular, with the proliferation of artificial intelligence technology in every field, the demand for specialized analytical tools to detect manipulation will both increase and become more specialized. Schreyer *et al.* [15] introduce an adversarial model based on a deep neural network to study the potential impact of adversarial (''deep fake'') attacks on computerized audit systems, which will require more sophisticated and specialized micro-level analytic tools to protect computer-assisted audit and accounting systems.

3) ELECTIONS AND VOTING

Klimek and colleagues [16], [17] have developed a tool called the ''election fingerprint'', which visualizes a two-dimensional histogram of voter turnouts and votes for winners to detect election fraud. Their method is based on detecting statistical irregularities in election data from a macro perspective.

Jimńez and Hidalgo [18] apply the election fingerprint to estimate statistical irregularities at the macro level and Benford's law to verify numerical data at the micro level. However, as the authors indicate, Benford's law should be applied with care when checking for election fraud. Deckert *et al.* [19] also analyzed Benford's law and concluded that this method can be problematic if used to detect election fraud.

One approach to overcoming the limitations of Benford's law is to use a machine learning algorithm. Levin *et al.* [20] applied machine learning algorithms (including supervised and unsupervised methods) to the detection of election fraud, while Zhang *et al.* [21] focused on supervised learning (''random forest'') using a hierarchical model to overcome the labeling issues in supervised learning. Although machine learning techniques—those based on deep neural networks (DNNs) in particular—have great potential for detecting data manipulation from a macro perspective, we still have to address issues related to the acquisition of proper datasets and making sure that the datasets have not been manipulated, as discussed in [21]. We believe that the method presented in this paper will help verify and refine the training datasets used by DNN-based supervised learning methods.

4) RESEARCH MISCONDUCT

The past decades have witnessed various cases of disappointing research misconduct directly related to the manipulation of numerical data in various research fields. According to the National Science Foundation [22], there are three categories of research misconduct: data fabrication, data falsification, and plagiarism. Among the three categories, we can regard integer data manipulation as data falsification.

In a typical research project, researchers will gather a finite number of samples of a particular quantity and estimate its population based on these samples, because it is too expensive or even impossible to investigate all items in the population. As a result, one of the main motivations for research misconduct by falsification is to support a weak theory or claim by adjusting a few ''outlier'' samples to affect the statistical estimation [23]–[25]. We have found that it might be unnecessary in many cases to enlarge the number of samples while preserving their relative proportions because the main purpose of sampling is to estimate the population using a limited number of samples based on statistical rules. However, at the same time, we cannot ignore the possibility of unfairly enlarging small research datasets—particularly when sampling tasks are very expensive or risky—to meet certain minimum requirements for the number of statistical samples. Here, the method illustrated in Figure 1 could be used.

B. DIGIT TEST

One practical digit test relies on the arguable hypothesis that human beings cannot generate random numbers naturally. Although some researchers, including Persaud [26] (but criticized by Figurska *et al.* [27]), deny the hypothesis, this assumption plays a practical role in detecting digit manipulation. Mosimann *et al.* [8], [28] show that people are often only careful when selecting the leftmost digits to fit an intended magnitude but pay less attention to the remaining digits—particularly the rightmost digits—causing the digits to lose their uniform distribution. This could lead to a practical tool for detecting data manipulation. Beber and Scacco [29] applied a last (rightmost) digit test to detect fraud in election data from Sweden, Nigeria, and Senegal.

However, because this approach depends on human psychology and statistical characteristics, it cannot address any algorithm-based manipulation processes such as that illustrated in Figure 1.

C. ROUND-OFF ERROR AND INTERVAL ARITHMETIC

Because modern digital computers use a limited number of bits to represent infinitely continuous real numbers, we should consider the subtle aspects of the floating-point representation of real numbers to avoid unexpected results in calculations that involve converting between floating-point and integer representations. Overton [30] introduced rudimentary concepts and tools that involve the ''condition number'' to enable applications to use the IEEE 754 floating-point number representation appropriately. Researchers and developers often rely on interval arithmetic [31] to overcome the discrete aspects of this representation. Revol [32] analyzed the influence of the condition number on interval computations. In a slightly different context, Layer and Quinlan [33] presented a parallel algorithm for processing N-way interval set intersections in genome research. (This is somewhat similar to our approach to solving the problem illustrated in Figure 4.)

III. MATHEMATICAL ANALYSIS

In this Section, we formalize mathematically the question regarding the numerical manipulation by multiplication and rounding illustrated in Figures 1 (c) and 1 (d).

A. PROBLEM FORMALIZATION

Suppose that nonnegative integers $y_1, y_2, \ldots, y_n$ are the entries in an integer dataset whose integrity is in question (List D in Figure 1). Without loss of generality, we can assume that $y_i \le y_j$ for $i < j$. Can we determine if the integer element y_i was obtained from some original integer element x_i by multiplication with a large constant real number α and then rounding? If so, can we recover the values for the original integer dataset $x_1, x_2, \ldots, x_n$ and α? This is expressed schematically in Figure 2.

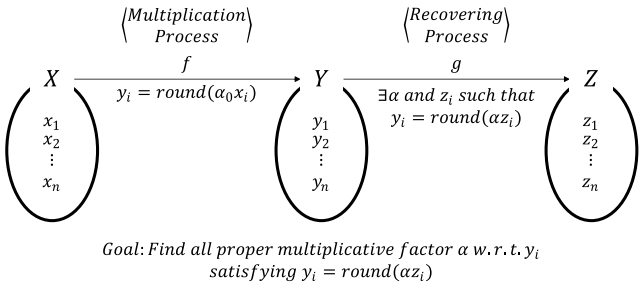


FIGURE 2. Multiplication process and recovery process.

More specifically, let $\alpha \ge 2$ be a nonintegral real number[1] and let $x_1, x_2, \ldots, x_n$ be nonnegative integers representing the genuine dataset before the multiplication process. We also assume that $x_i \le x_j$ for $i < j$ without loss of generality. Let

$$\widetilde{y}_i := \alpha x_i \tag{1}$$

for each $i = 1, 2, \ldots, n$. Because $\widetilde{y}_i$ may not be an integer, we denote y_i as the rounded value of $\widetilde{y}_i$. That is,

$$y_i := round(\alpha x_i). \tag{2}$$

where y_i represents the manipulated value, which is revealed publicly.

[1]We note that the assumption $\alpha \ge 2$ can be replaced by $\alpha \ge 1 + \delta$ for some $0 < \delta < 1$. However, in practice, a ''blow-up'' process is used when the data needs to be substantially enlarged, so the assumption $\alpha \ge 2$ will suffice. With this assumption, we do not need to introduce δ as above, and the analysis becomes simpler.

For each nonintegral real number α, denote $z_i := round(y_i/\alpha)$ for $1 \le i \le n$ and let

$$D(\alpha) = \sum_i |y_i - round(\alpha z_i)|. \tag{3}$$

Our problem can then be summarized as determining if α with $D(\alpha) = 0$ exists for a given y_i.

Our goal is to propose an algorithm that detects all α with $D(\alpha) = 0$ within a "finite" number of operations. To the best of our knowledge, no viable solution to this problem exists in the literature.

There are some technical hurdles in addressing this problem. First, $D(\alpha)$ is not unimodal and has many local minima and maxima caused by the two rounding functions. Second, because the interval containing α with $D(\alpha) = 0$ could be very narrow, common iterative approaches (including the "brute force" search) may not succeed in finding the interval even with a small step size (i.e., with a low level of granularity).

To enable a succinct discussion in the remainder of this paper, we define a proper multiplicative factor as follows.

For arbitrary integers $y_1, \cdots, y_n$, we say that a nonintegral real number $\alpha \ge 2$ is a *proper multiplicative factor* with respect to y_i if there exist integers x_i such that (2) holds for $1 \le i \le n$.

For given integers y_i, a proper multiplicative factor α and the corresponding original data x_i may not be unique for two reasons. First, all elements of a narrow interval around a proper α might also be proper. Second, it follows that, if α is proper, then α/k for an integer $k \ge 2$ is also proper for (a different) original dataset kx_i. However, in this case, all integers kx_i are multiples of k, which would be unrealistically improbable for a randomly selected set of integers. Therefore, it makes sense that our goal can be to find a proper factor α that is as large as possible.

B. REDUCING SEARCH RANGE

We first reduce the search range of proper multiplicative factors. Recall that we assume that $y_i, \widetilde{y}_i$ and x_i are not decreasing in i. Consider the differences between two adjacent values

$$\Delta y_i := y_i - y_{i-1}, \quad \Delta\widetilde{y}_i := \widetilde{y}_i - \widetilde{y}_{i-1} \quad \text{and} \quad \Delta x_i := x_i - x_{i-1},$$

where $1 \le i \le n$ and $y_0 = \widetilde{y}_0 = x_0 = 0$. Equation (1) leads to

$$\Delta\widetilde{y}_i = \alpha \Delta x_i.$$

The fact that $|\widetilde{y}_i - y_i| \le 1/2$ implies that

$$\Delta y_i - 1 \le \alpha \Delta x_i \le \Delta y_i + 1.$$

We denote

$$\Delta y = \min_{\substack{1 \le i \le n \\ \Delta y_i \ne 0}} \Delta y_i \quad \text{and} \quad \Delta x = \min_{\substack{1 \le i \le n \\ \Delta x_i \ne 0}} \Delta x_i,$$

and therefore,

$$\Delta y - 1 \le \alpha \Delta x \le \Delta y + 1.$$

Consequently,

$$\frac{\Delta y - 1}{\Delta x} \le \alpha \le \frac{\Delta y + 1}{\Delta x}. \tag{4}$$

We assume $\Delta x \ge 1$ as defined in (4), while allowing duplicated entries in x_i (i.e. $x_i = x_{i+1}$). We can then determine the range of Δx from (4) and $\alpha \ge 2$ as

$$\frac{\Delta y - 1}{\alpha} \le \Delta x \le \frac{\Delta y + 1}{\alpha} \le \frac{\Delta y + 1}{2}.$$

Because Δx is a positive integer,

$$1 \le \Delta x \le \left\lfloor \frac{\Delta y + 1}{2} \right\rfloor.$$

Therefore, there is a finite number of possible ranges for α, according to the minimum differences between two adjacent original integer values x_i. We set a positive integer ℓ as a possible Δx to scan and define $\alpha_s = \alpha_s(l)$ and $\alpha_t = \alpha_t(l)$ from (4) as

$$\alpha_s = \frac{\Delta y - 1}{\ell} \quad \text{and} \quad \alpha_t = \frac{\Delta y + 1}{\ell}.$$

We also let $I_\ell = \alpha_s, \alpha_t$ and

$$I = \bigcup_{1 \le \ell \le \lfloor \frac{\Delta y + 1}{2} \rfloor} I_\ell. \tag{5}$$

It follows from (4) that $\alpha \in I$ and the range of the set I is much smaller than the trivial range $2 \le \alpha \le \Delta y$ because the length of I is

$$|I| = \sum_{\ell=1}^{\lfloor \frac{\Delta y + 1}{2} \rfloor} \frac{2}{\ell} \approx 2 \log \Delta y$$

as $\Delta y \to \infty$. Effectively, therefore, the search range for α is decreased from Δy to $2 \log \Delta y$.

C. NAÏVE ALGORITHMS AND COMPUTATIONAL ISSUES

Having found finite search ranges for possible α intervals based on Δx (see Section III-B), one approach might then be to apply standard numerical algorithms to (3). However, because of the many local minima and its nondifferentiability, solving $D(\alpha) = 0$ using iterative or gradient-based optimization methods would not be successful. These obstacles can be shown in plots of $D(\alpha)$ near any intervals of proper multiplicative factors. Figure 3 shows $D(\alpha)$ near an interval of proper multiplicative factors for the procedures in Figures 1 (c) and 1 (d) at two different scales. In the upper graph of Figure 3, the interval of proper multiplicative factors appears as a point in the red circle. The lower graph of Figure 3 magnifies $D(\alpha)$ near the red circle in the upper graph by around 10^5 times. Unlike the upper graph, the magnified graph suggests that the proper multiplicative factor is not a single real number, but that there are infinitely many proper multiplicative factors in the interval [6213.1339999996999, 6213.1340000001492). Note that any real number α_0 in this interval satisfies $D(\alpha_0) = 0$. This includes the multiplier given in Figure 1 (i.e. 6213.134). From Figure 3, we can observe

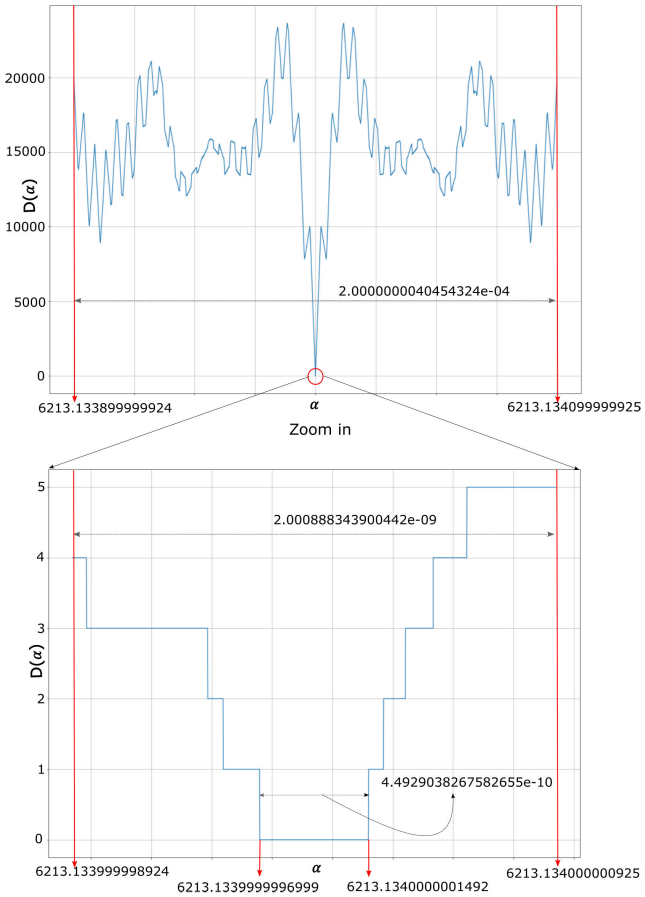


FIGURE 3. Plot of D(α) near the interval containing the proper multiplicative factor $\alpha \in$ [6213.1339999996999, 6213.1340000001492] for the procedures shown in Figures 1 (c) and 1 (d).

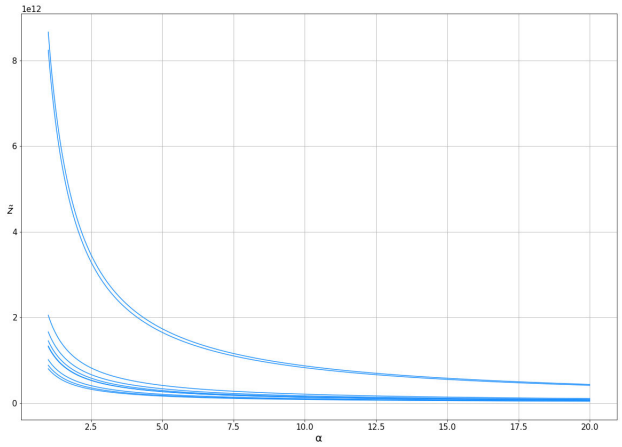


FIGURE 4. $x = \frac{y_i}{\alpha} (i = 1 \ldots 10)$.

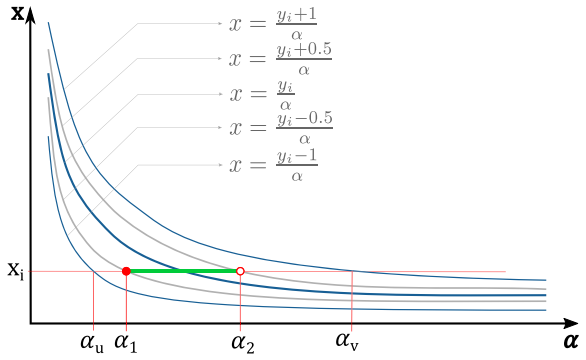


FIGURE 5. Reconstruction of the manipulation process for x_i.

some of the issues hindering any iterative or gradient-based optimization methods for $D(\alpha) = 0$.

- Because of the very large number of local minima in $D(\alpha)$, any optimization approach could fail to find the global minimum (the upper graph in Figure 3).
- With $D(\alpha)$ being nondifferentiable (because of the rounding operation), gradient-based optimization methods would not work well (the lower graph in Figure 3).
- Although the proper multiplicative factors form a continuous interval, the length of the interval is quite short (it is 4.5×10^{-10} in Figure 3). The infinitesimal length of the interval and the narrow valley near the global minimum make it hard to determine an effective iteration step size.
- Given that we consider only $\ell's$ (from (5)) that have proper multiplicative factors in Figure 3 and there are far more $\ell's$ that do not, scanning all $\ell's$ based on iterative methods could be impractical.
- Finally, common iterative methods could be numerically unstable because executing floating-point operations with very small real numbers and very large real numbers does often cause instability.

In this paper, we propose a numerically robust algorithm for solving $D(\alpha) = 0$ based on interval union and intersection operations to overcome the above issues.

D. ANALYSIS BASED ON INTERVALS

Figure 4 shows $x = \frac{y_i}{\alpha} (i = 1 \ldots 10)$ plots for the Figure 1 case. Note that there are 10 curves for all y_i. From these graphs, we can obtain a clearer understanding of the nature of the manipulation we are dealing with by reconstructing the manipulation process.

Figure 5 focuses on the specific data manipulation process for an original integer data entry x_i, showing that the real number α should be within the interval $[\alpha_1, \alpha_2]$ (green line), for which

$$\alpha_1 = \frac{y_i - 0.5}{x_i} \quad \text{and } \alpha_2 = \frac{y_i + 0.5}{x_i}$$

when the manipulator multiplies α and the original integer value x_i to give the disclosed integer value y_i. Note that x_i multiplied by the interval $[\alpha_1, \alpha_2]$ makes $y_i - 0.5 \le \widetilde{y} < y_i + 0.5$, leading to $round(\widetilde{y}) = y_i$. Otherwise, we would obtain $y_i - 1$ when we multiply x_i and any real number in $[\alpha_u, \alpha_1]$ and $y_i + 1$ when in $[\alpha_2, \alpha_v]$, as shown in Figure 5. Figures 3 and 5 together show that the proper multiplicative factor in

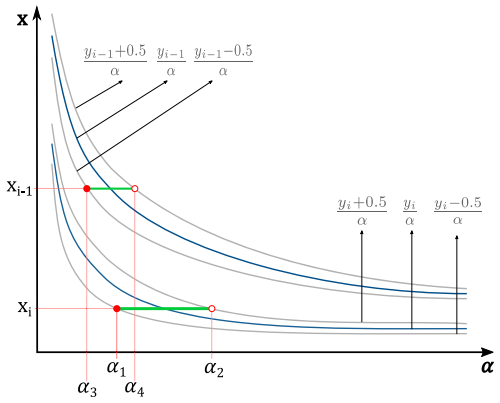


FIGURE 6. Reconstruction process with two original values, x_{i-1} and x_i.

this problem cannot be a single real number but is an interval of real numbers.

However, Figure 6 provides us with a clue to solving this puzzle because other original data entries $x_k (k \neq i)$ will have corresponding "proper" intervals like that for x_i within each searching range $I_\ell = \alpha_s, \alpha_t$. We could search for the common multiplicative factor used in the manipulation via the intersection of all proper intervals for x_k.

Figure 6 shows an additional proper multiplicative interval for x_{i-1} within the search range $I_\ell = \alpha_s, \alpha_t$, namely $[\alpha_3, \alpha_4]$. This gives the interval $[\alpha_1, \alpha_4]$ as containing the proper multiplicative factors used for the manipulation of the two original data entries x_{i-1} and x_i, produced by the intersection of two intervals (i.e., $[\alpha_1, \alpha_2] \cap [\alpha_3, \alpha_4]$). In this way, we can determine if a proper multiplicative interval exists in each search range $[\alpha_s, \alpha_t]$ and obtain the exact interval that satisfies $D(\alpha) = 0$ for all x_i, if it exists at all. However, because we still do not know x_i (we know only y_i), we need to test integer candidates for x_i based on y_i and the search ranges $I_\ell = \alpha_s, \alpha_t$ defined in 4.

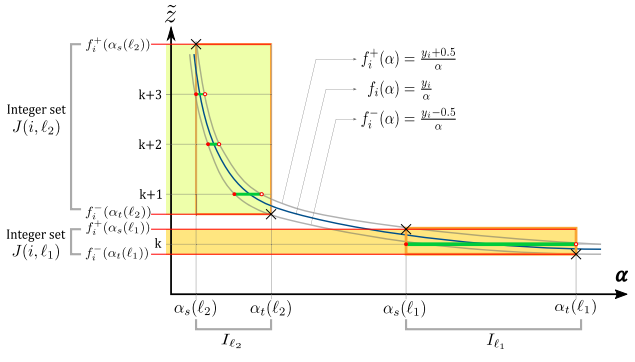


FIGURE 7. The sets $J(i, \ell_1)$ and $J(i, \ell_2)$.

Figures 5 and 6 illustrate the situation where the x_i values are known. If they are unknown, we have to estimate the validity of several candidate integer values along the vertical axis. For example, Figure 7 shows two search ranges I_{ℓ_1} and I_{ℓ_2}, where $\ell_1 < \ell_2$. For I_{ℓ_1}, the corresponding range along the vertical axis contains at least one integer (i.e., k) but the range for I_{ℓ_2} includes more than one integer (i.e., $k+1, k+2$, and $k+3$) (see Lemma 1, below).

To formalize the validation of candidates along the vertical axis, we set a real variable $\tilde{z}_i$ and an integer variable z_i.

$$tz_i := \frac{y_i}{\alpha} \quad \text{and } z_i := round(\tilde{z}_i) \tag{6}$$

for the $i = 1, 2, \ldots, n$ that can represent the original data x_i.

Lemma 1 gives a necessary and sufficient condition for a proper multiplicative factor to exist.

Lemma 1: A real number α is a proper multiplicative factor with respect to y_i if and only if

$$-\frac{1}{2\alpha} < \tilde{z}_i - z_i \leq \frac{1}{2\alpha} \tag{7}$$

for $1 \leq i \leq n$.

Lemma 1 also implies that large α values will lead to narrow ranges along the vertical axis and small α values to wide ranges (see Figure 7). (Appendix A gives a proof of Lemma 1.)

To validate integer candidates along the vertical axis, which is now the $\tilde{z}$ axis, we focus on candidate integers $z_i = z_i(\ell)$ such that (7) holds for $i \in [1, n]$ and $\alpha \in I_\ell = [\alpha_s, \alpha_t]$. Note that, for fixed i and ℓ, there is only a finite number of such z_i, which should help us find all proper α using a finite number of operations.

Inequality (7) is equivalent to $-1/(2\alpha) \leq z_i - \tilde{z}_i < 1/(2\alpha)$. That is,

$$\tilde{z} - 1/(2\alpha) \leq z_i < \tilde{z}_i + 1/(2\alpha),$$

and therefore (6) gives

$$f_i^-(\alpha) \leq z_i < f_i^+(\alpha),$$

where

$$f_i^-(\alpha) = \frac{y_i - 0.5}{\alpha}, \quad f_i(\alpha) = \frac{y_i}{\alpha} \text{ and } f_i^+(\alpha) = \frac{y_i + 0.5}{\alpha}.$$

Because f_i^-, f_i, and f_i^+ are decreasing in α, we can infer that the set of all integers z_i satisfying (7) for some $\alpha \in I_\ell$ is

$$J(i, \ell) := \left\{ j \mid f_i^-(\alpha_t) \leq j < f_i^+(\alpha_s), \quad j \text{ is an integer} \right\},$$

which is shown in Figures 7 and 8.

Lemma 1 implies that α_ℓ is proper if and only if, for each $i \in [1, n]$, there exists an integer $j \in J(i, \ell)$ such that $f_i^-(\alpha) \leq j < f_i^+(\alpha)$, or equivalently, $\alpha \in \left[\frac{y_i-0.5}{j}, \frac{y_i+0.5}{j}\right)$.

Note that those horizontal intervals $\left[\frac{y_i-0.5}{j}, \frac{y_i+0.5}{j}\right)$ with $\alpha > 1$ do not overlap when projected onto the α axis (see Figure 8).

Lemma 2: For each $i \in [1, n]$ and $\ell \in [1, \lfloor \frac{\Delta y+1}{2} \rfloor]$, the intervals $\left[\frac{y_i-0.5}{j}, \frac{y_i+0.5}{j}\right)$ for $j \in J(i, \ell)$ do not overlap.

(See Appendix B for a proof.)

Considering all $j \in J(i, \ell)$, a proper factor $\alpha \in I_\ell$ is contained in

$$I^*(i, \ell) := \bigsqcup_{j \in J(i,\ell)} \left[\frac{y_i - 0.5}{j}, \frac{y_i + 0.5}{j}\right), \tag{8}$$

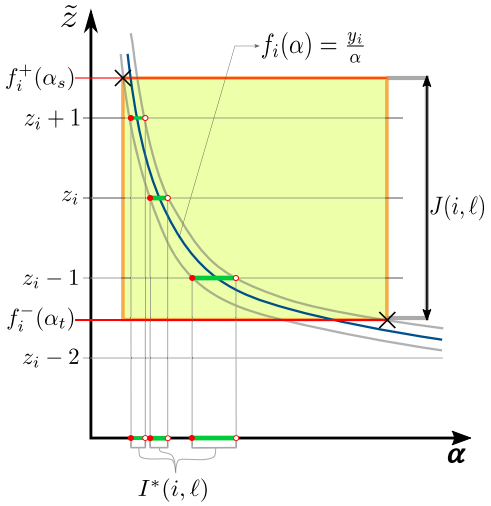


FIGURE 8. The set $I^*(i, \ell)$ when $|J(i, \ell)| = 3$.

for all $1 \le i \le n$, where the notation $\bigsqcup$ means a disjoint union. (See Figure 8 for $I^*(i, \ell)$ when $|J(i, \ell)| = 3$.)

It follows that $\alpha \in I_\ell$ is a proper multiplicative factor if and only if

$$\alpha \in I_\ell^* := \bigcap_{i \in [1,n]} I^*(i, \ell). \tag{9}$$

Therefore, if I_ℓ^* is not empty, we obtain a proper multiplicative factor $\alpha \in I_\ell^*$ and we have the following lemma.

Lemma 3: There exists a proper multiplicative factor α if and only $\bigcup_{1 \le \ell \le \lfloor \frac{\Delta y+1}{2} \rfloor} I_\ell^* \neq \emptyset$.

Based on (8) and (9), we can infer that the set of all proper α is not a discrete set but a union of disjoint intervals. If we want to obtain a specific large proper α, then such a multiplicative factor α can be chosen from the rightmost interval of $\bigcup_\ell I_\ell^*$.

Based on Lemma 3, it is clear that if the y_i are obtained by multiplication, then the probability that $\bigcup_\ell I_\ell^* \neq \emptyset$ is 1.

Next, suppose that the y_i values have not been multiplied. Can we estimate the probability that $\bigcup_\ell I_\ell^* \neq \emptyset$? Because this is not easy, we could instead consider a weak estimate for the related value $\sum_{1 \le \ell \le \lfloor \frac{\Delta y+1}{2} \rfloor} \mathbb{P}\alpha \in I_\ell^* \mid \alpha \in I_\ell$ in the best possible situation. Then,

$$\sum_{1 \le \ell \le \lfloor \frac{\Delta y+1}{2} \rfloor} \mathbb{P}\alpha \in I_\ell^* \mid \alpha \in I_\ell \le \frac{(\Delta y)^{n+1}}{(n+1)2^{n+1}(\Delta y - 1)^n} \cdot \prod_{1 \le i \le n} \frac{2 + \frac{\Delta y}{y_i}}{2 - \frac{1}{y_i}}. \tag{10}$$

(See Appendix C for the weak estimation of (10)). Here, the probability decreases as n and $y_i/\Delta y$ increase. We can speculate that the probability of $\bigcup_\ell I_\ell^* \neq \emptyset$ has a similar property.

Inequality (10) gives an upper limit to the conditional probability that any arbitrary α values we are using in the estimation would be located erroneously in the interval given by (8). That is, if we use higher upper limits in (10), we would be more likely to obtain erroneous predictions. Conversely, we would be more likely to obtain accurate predictions with lower upper limits. This is consistent with our experimental results, described in Section V.

Algorithm 1 Detection Algorithm

Data:
- Disclosed integer numbers $\{y_1, \ldots, y_n\}$, (sorted, $y_{i+1} \ge y_i$)

Result:
- Flag $\boldsymbol{m}$ indicating whether $\{y_i\}$ is multiplied or unmultiplied
- Rightmost interval $\boldsymbol{I^*}$ of the set of all proper multiplicative factors (i.e. α)

$m \leftarrow$ *unmultiplied*
$I^* \leftarrow \varnothing$
$I_\ell^* \leftarrow \varnothing$
$\Delta y \leftarrow max(1, min(\{y_{i+1} - y_i | y_i \in \{y_1, \ldots, y_n\}\}))$
if $\Delta y = 1$ **then**
  return m, I_ℓ^*
end
for $\ell \leftarrow 1$ **to** $\lfloor \frac{\Delta y+1}{2} \rfloor$ **do** // $\Rightarrow$ § *III* − *B*
  $\alpha_s \leftarrow \frac{\Delta y-1}{\ell}$, $\alpha_t \leftarrow \frac{\Delta y+1}{\ell}$
  if $\alpha_s < 2$ **then**
    $\alpha_s \leftarrow 2$
  end
  $I_\ell^* \leftarrow [\alpha_s, \alpha_t]$
  for $i \leftarrow 1$ **to** n **do**
    $I^*(i, \ell) \leftarrow \varnothing$
    $z_{low} \leftarrow \left\lceil \frac{y_i - 0.5}{\alpha_t} \right\rceil$, $z_{high} \leftarrow \left\lfloor \frac{y_i + 0.5}{\alpha_s} \right\rfloor$
    for $j \leftarrow z_{low}$ **to** z_{high} **do** // $\Rightarrow$ `Eq. (8)`
      $I^*(i, \ell) \leftarrow I^*(i, \ell) \cup \left[\frac{y_i - 0.5}{j}, \frac{y_i + 0.5}{j}\right)$
    end
    $I_\ell^* \leftarrow I_\ell^* \cap I^*(i, \ell)$
    if $I_\ell^* = \varnothing$ **then**
      break
    end
  end
  if $I_\ell^* \neq \varnothing$ **then**
    $m \leftarrow$ *multiplied*
    $I^* \leftarrow$ *the rightmost interval of* I_ℓ^*
    break
  end
end
return m, I^*

IV. ALGORITHM

A. OVERVIEW

Based on the findings described in Section III, Algorithm 1 determines if the disclosed integer dataset y_i has been manipulated by multiplication with a nonintegral real number $\alpha (\ge 2)$ and a rounding operation. If so, it finds the rightmost

Algorithm 2 Restoration Algorithm

Data:
- Disclosed integer numbers $\{y_1, \ldots, y_n\}$, (sorted, $y_{i+1} \geq y_i$)
- Proper multiplicative factor interval $I^* = [\alpha_p, \alpha_q)$ obtained from Algorithm 1

Result:
- The original integer data $\{x_1, \ldots, x_n\}$, (sorted, $x_{i+1} \geq x_i$)

1 $\alpha_{avg} \leftarrow \frac{\alpha_p + \alpha_q}{2}$
2 **for** $i \leftarrow 1$ **to** n **do**
3 $\quad x_i \leftarrow \texttt{round}\,(\frac{y_i}{\alpha_{avg}})$
4 **end**
5 **return** $\{x_1, \ldots, x_n\}$

interval $\boldsymbol{I^*}$ of the set (i.e., the largest α) comprising all proper multiplicative factors. As discussed in Section III, any real number α in $\boldsymbol{I^*}$ generates the same y_i after the operation $y_i = round(\alpha x_i)$.

Algorithm 2 restores the original data x_i, after the disclosed data y_i are determined as *modified*, using the proper multiplicative factor interval $I^* = [\alpha_p, \alpha_q]$ returned from Algorithm 1. For better numerical robustness, we apply the average of α_p and α_q in the restoration. The algorithm is numerically robust in that it guarantees a bounded number of loop operations (Line 8 and 14 in Algorithm 1) and relies mainly on interval union and intersection operations (Line 18 and 20 in Algorithm 1) for subtle floating-point numbers. These features also make it well suited to parallelization and/or hardware-based implementation. (See Section IV-C for further discussion on the numerical robustness of Algorithm 1.)

B. IMPLEMENTATION AND CODE OPTIMIZATION

We implemented Algorithm 1 using 64-bit double-precision floating-point variables in C++ and Python 3.7 within the Ubuntu 18.04.3 LTS environment. Most of the major calculations in our algorithm involve the union and intersection of intervals in (8) and (9) (Line 18 and 20 in Algorithm 1). (The problem of finding the intersection of disjoint unions of intervals is known as the *N-way interval set intersection problem* [33].)

We implemented most of the code in the Python 3.7 programming language, but we improved the efficiency of the critical interval operations by using native binary executables (as a single-threaded shared library) implemented via the *interval_set* class template in the *Boost* C++ library [34] for better performance.

Algorithm 1 may require a large computation time, such as when the second **for** loop (Line 14) is repeated for all i values in the unmultiplied-dataset case. One approach to improving this process is to find a way to obtain $I^*_\ell = \varnothing$ earlier in the loop to enable early termination. From observation, we have found a tendency towards $I^*(i, \ell) \cap I^*(i+1, \ell) \neq \varnothing$ in the loop because the adjacent unions of intervals $I^*(i, \ell)$ and $I^*(i+1, \ell)$ are likely to have similar interval patterns. Conversely, unions of distant intervals (e.g., $I^*(1, \ell)$ and $I^*(n, \ell)$) are likely to have different interval patterns, which makes the intersection a null interval set. We can apply this heuristic simply by *rolling* the search indices from $(1, 2, .., n)$ to $(n, 1, 2, \ldots, n-1)$ in the second **for** loop (Line 14). For example, we found that the computation speed for a specific integer dataset with 35 entries (and which had not been multiplied) was increased by a factor of about 160.

Another mechanism for early termination begins with the third **for** loop (Line 17) in Algorithm 1. When there are no integer candidates between z_{low} and z_{high} (i.e., $z_{low} = z_{high}$ in Line 16), we can ignore the third **for** loop, abandon the operations for the current candidate Δx, and move on to the next candidate Δx (see Line 22). Note that even a single null interval within $I^*(i, \ell)$ can make I^*_ℓ a null interval, which means that we would be unable to find a valid α under the current assumption about the minimum difference between two consecutive original data entries (i.e. Δx).

C. NUMERICAL ROBUSTNESS

As shown in Figure 4, the typical interval length for valid α values is around 10^{-10}. These narrow intervals imply we cannot use 32-bit single floating-point variables because their calculation results are only accurate to 7 significant digits at most. We therefore use 64-bit double floating-point variables, which can calculate accurately to 16 significant digits for decimal numbers [30]. Because we are dealing with nonintegral real numbers at high precision, numerical instability can occur during incremental traverse operations (e.g., brute force search or optimization algorithms). Here, the finite machine precision can lead to erroneous cancellation results. A useful tool for analyzing numerical robustness in interval algorithms involves the condition number [32]. As a rule of thumb [30], the number of valid—i.e., *not cancelled by floating-point operations*—decimal digits N_{digit} in the *significand* of floating-point numbers based on the IEEE 754 standard is estimated as

$$N_{digit} \approx \lceil p \log_{10} 2 - \log_{10} k_f, \rceil$$

where k_f is the condition number of operation function f, $p = 24$ for single precision, and $p = 52$ for double precision.

However, Algorithm 1 depends on the intersection and union of interval sets and they are the two most prevalent operations, following which we need only to estimate the *sign* bit of floating-point numbers. That is, intersection and union compare two real numbers α_1 and α_2, which represent either the start point or the end point of two different intervals by estimating

$$f(\alpha_1, \alpha_2) = sgn(\alpha_1 - \alpha_2),$$

where $sgn(x)$ is the sign function. As a result, Algorithm 1 does not rely on having valid digits in the significand but depends only on its sign bit, which makes Algorithm 1 robust

against instability caused by limited-precision-based erroneous cancellations.

V. EXPERIMENTS

For the data in Figure 1, Algorithm 1 returned $m = \textit{modified}$ and $I^* = [6213.1339999996999, 6213.1340000001492]$, which includes the actual manipulation factor 6213.134. As shown in Figure 3, every real number α in I^* satisfies $D(\alpha) = 0$ in (3). Moreover, Algorithm 2 successfully restored the original data for the manipulation depicted in Figures 1 (c) and 1 (d).

Because a single successful data test is insufficient to justify the validity of our algorithms or to explore its limitations, we now report on our stress testing of Algorithm 1, using random datasets generated according to specific rules.

A. DESIGN OF TEST DATASETS

To stress test Algorithm 1, we prepared 20,000 multiplied positive integer datasets and the corresponding 20,000 unmultiplied positive integer datasets as a control group. The following design rules for each parameter were motivated by the recent data manipulation case disclosed to the public [2].

For the multiplied datasets, we first chose 1,000 original datasets $\{x_1, x_2, \ldots, x_n\}$ involving three parameters, namely the number n of integer data entries and the lower and upper bounds of x_i, as follows.

1) $n = 10, 20, \ldots, 100$.
2) The lower bound of $x_i = 100 \times 2^i$ for $0 \le i \le 9$.
3) The ratio $\dfrac{\text{upper bound of } x_i}{\text{lower bound of } x_i} = 2, 4, 6, \ldots, 20$.

We generated the multiplied datasets $\{y_1, y_2, \ldots, y_n\}$ as follows. For each choice of $\{x_1, x_2, \ldots, x_n\}$, we chose 20 α values, with five nonintegral real numbers being uniformly and randomly chosen from each of the four ranges [2, 10], [10, 100], [100, 1000], and [1000, 10000], under the condition that the fractional part should have five digits.

Next, for the control group (i.e., unmultiplied datasets), we generated random datasets such that the i-th dataset had the same smallest and largest positive integer data entries as the corresponding i-th multiplied dataset.

The calculations were executed in parallel by distributing the single-thread-based tasks in Algorithm 1 among 40 Xeon processor cores.

We discuss the details of our results in the following subsections.

B. DEFINITIONS FOR EXPERIMENTS

To compare the results of our tests, we introduce some definitions.

[Integrity of Dataset] A disclosed dataset $\{y_i\}$ is called *actually multiplied* if $\{y_i\}$ is generated by multiplication with a multiplicative factor $\alpha_0 \ge 2$. Conversely, a disclosed dataset $\{y_i\}$ is called *actually unmultiplied* if $\{y_i\}$ is generated without multiplications.

[Predictions From Algorithm 1] A disclosed dataset $\{y_i\}$ is called *predicted as multiplied* if Algorithm 1 determines that it has been multiplied (i.e., $\{y_i\}$ has a proper multiplicative factor $\alpha \ge 2$). Conversely, a disclosed dataset $\{y_i\}$ is called *predicted as unmultiplied* if Algorithm 1 determines it has not been multiplied (i.e., $\{y_i\}$ has no proper multiplicative factor $\alpha \ge 2$).

From Definitions V-B and V-B, we can categorize erroneous predictions from Algorithm 1 into three cases.

1) **False Positive (FP)** - A disclosed dataset that was *actually unmultiplied* but is *predicted as multiplied*.
2) **False Negative (FN)** - A disclosed dataset that was *actually multiplied* but is *predicted as unmultiplied*.
3) **Error in finding α (ER)** - A disclosed dataset that was *actually multiplied* using a multiplicative factor α_0 is predicted (correctly) as *actually multiplied* but the rightmost interval returned from Algorithm 1 does not contain α_0.

We also categorize correct predictions into two cases.

1) **True Positive (TP)** - A disclosed dataset that was *actually multiplied* and is *predicted as multiplied*.
2) **True Negative (TN)** - A disclosed dataset that was *actually unmultiplied* and is *predicted as unmultiplied*.

C. TEST RESULTS

The table in Figure 9 (a) summarizes the overall results. For all the datasets that were *actually multiplied*, Algorithm 1 predicted them correctly as *actually multiplied* with no FN cases. However, there were some extremely rare ER cases (22 cases out of 20,000 experimental samples or 0.44%), for which α was less than six. For all the datasets that were *actually unmultiplied*, there were 3,508 FP cases out of 20,000 control samples (17.54%).

The results given in Figure 9 (a) might appear initially to be unimpressive, particularly for the TN cases, but we found that *we could achieve better results with a more careful selection of parameters* (i.e., lower bounds of α and the number of data entries in each dataset, n). This will be discussed next.

1) RESULTS WITH RESPECT TO THE LOWER BOUNDS OF α

The tables in Figures 9 (b) and 9 (c) give the results with respect to two different lower bounds of α. If the lower bound of α is 6, as shown in Figure 9 (b), all 17,523 experimental cases were correctly predicted and the ER cases decreased from the 0.44% in Figure 9 (a) to 0%. Moreover, the TP cases decreased from 17.54% in Figure 9 (a) to 2.74%.

As given in Figure 9 (c), both the ER and FP cases decreased to 0% when the lower bound was set to 16, which means that we can rely on Algorithm 1 with high confidence if α is greater than or equal to 16. The graph in Figure 10 (a) describes the occurrence ratio of the ER and the FP cases with respect to the lower bound of α. This graph is also consistent with the probability estimation in (11) (see Appendix A).

2) EFFECTS OF THE NUMBER OF DATA ENTRIES

A second point to note from our results is that the success rates of the predictions rise as the number of data entries

(a) overall		Prediction whether multiplied or not along with the rightmost interval I*			Total cases
		Positive (multiplied)		**Negative** (unmultiplied)	
		$\alpha \in I^*$ (correct interval)	$\alpha \notin I^*$ (incorrect interval)		
Truth	**Positive** (multiplied)	19912 (99.56%) TP	88 (0.44%) ER	0 (0%) FN	20000
	Negative (unmultiplied)	3508 (17.54%) FP		16492 (82.46%) TN	20000

(b) $\alpha \geq 6$		Prediction			Total cases
		P		**N**	
		$\alpha \in I^*$	$\alpha \notin I^*$		
Truth	**P**	17523 (100%)	0 (0%)	0 (0%)	17523
	N	547 (2.74%)		19453 (97.26%)	20000

(c) $\alpha \geq 16$		Prediction			Total cases
		P		**N**	
		$\alpha \in I^*$	$\alpha \notin I^*$		
Truth	**P**	14679 (100%)	0 (0%)	0 (0%)	14679
	N	0 (0%)		20000 (100%)	20000

(d) $n \geq 16$		Prediction			Total cases
		P		**N**	
		$\alpha \in I^*$	$\alpha \notin I^*$		
Truth	**P**	16000 (100%)	0 (0%)	0 (0%)	16000
	N	150 (0.94%)		15850 (99.6%)	16000

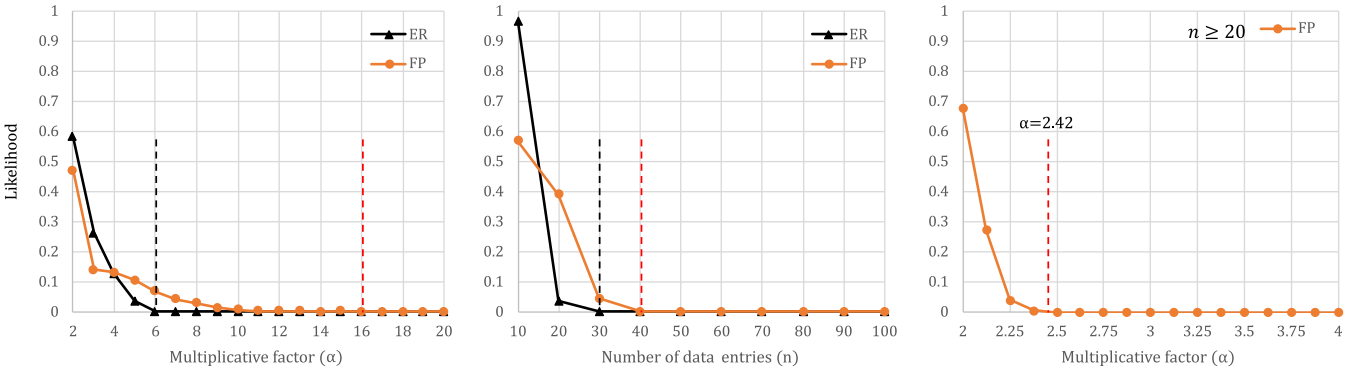
(e) $n \geq 20$, $\alpha \geq 2.42$		Prediction			Total cases
		P		**N**	
		$\alpha \in I^*$	$\alpha \notin I^*$		
Truth	**P**	15790 (100%)	0 (0%)	0 (0%)	15790
	N	0 (0%)		16000 (100%)	16000

(f)	# of cases	Max. time (sec)	Min. time(sec)	Avg. time(sec)
TP	19912	7.35	0.00003	0.029
TN	16492	21914.43	0.000025	253.54
FP	3508	6607.47	0.00056	88.57
ER	88	0.69	0.00078	0.077
FN	0	NA	NA	NA

FIGURE 9. Prediction results.

n increases. The table in Figure 9 (d) shows that there are 16,000 datasets that have at least 16 data entries for both the experimental and control groups. For the experimental group, all samples were correctly predicted and there were no failures in finding correct α values (i.e., no ER cases). Only 150 cases out of the 16,000 control samples were wrongly predicted (FP). Figure 10 (b) shows how ER and FP change according to n. This shows that we can avoid any nonzero ER cases by limiting the samples to those with $n \geq 30$ and the nonzero FP cases with $n \geq 40$ while leaving other conditions unchanged. We can therefore say that Algorithm 1 predicts perfectly for datasets having at least 40 data entries ($n \geq 40$).

We can mitigate this limitation on n further by selecting appropriate combinations of α and n. Although using small α values (i.e., $\alpha < 6$) can negatively affect prediction rates, as shown in Figures 9 (b) and 9 (c), we obtain better prediction rates when we consider datasets containing at least 20 data entries, even for small α values. The table in Figure 9(e) shows the case where the lower bound of n is set to 20 and the

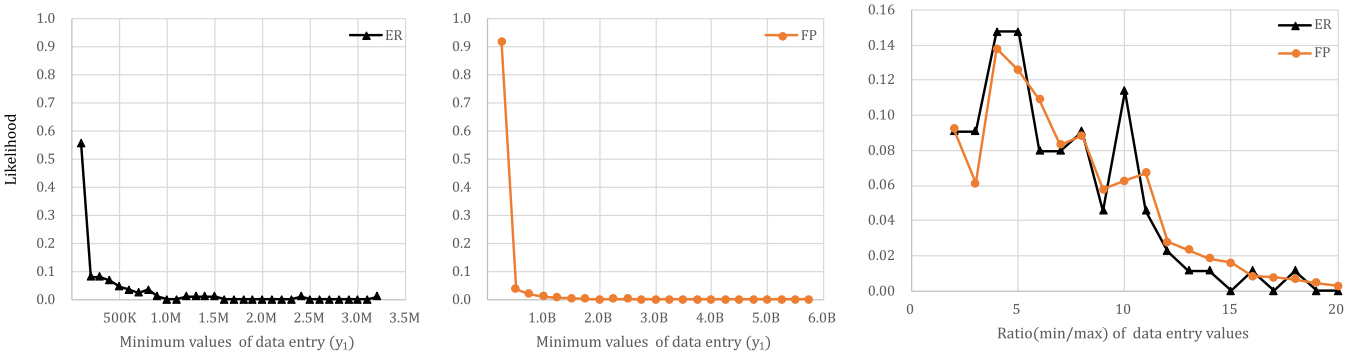


(a) Likelihood of prediction errors (ER and FP) on α

(b) Likelihood of prediction errors (ER and FP) on the number of data entries

(c) Likelihood of prediction error (FP) on α and the number of data entries

FIGURE 10. Prediction error rates with conditions on α and n.



(a) Likelihood of prediction error (ER) on minimum values of data entry

(b) Likelihood of prediction error (FP) on minimumvalues of data entry

(c) Likelihood of prediction error (ER and FP) on min/max ratio of data entry values

FIGURE 11. Prediction error rates considering the minimum values in disclosed datasets and the ratio between the minimum and the maximum values in disclosed datasets.

lower bound of α is set to 2.42. Here, Algorithm 1 predicted perfectly for 15,790 experimental samples and 16,000 control samples. In Figure 10 (c), where $n \geq 20$, the FP cases increase as α decreases below 2.42.

3) EFFECTS OF THE MINIMUM AND THE MAXIMUM VALUES OF THE DISCLOSED DATASET

As defined in Section III-A, y_1 and y_n indicate the minimum and the maximum values of the disclosed dataset $\{y_i\}$, respectively.

The graphs in Figures 11(a) and 11 (b) show that both the ER and FP error rates decrease as y_1 increases. We can understand this tendency from the conditional probability in (10), which describes the possibility that any arbitrary α values we are considering would be located erroneously within I^* (8). That is, we would expect the ER and FP error rates to be higher when using a high upper limit for the conditional probability in (10).

Therefore, as y_1 increases, the ratio $y_i/\Delta y$ also increases, and the upper bound of the conditional probability in (10) decreases, which implies a lower likelihood of prediction errors related to the ER and FP cases. The graph in Figure 11 (c) shows similar tendencies for the ER and FP cases with respect to the ratio y_n/y_1.

In summary, we can obtain more accurate prediction results with higher y_i and y_n/y_1.

D. PROCESSING TIME

The table in Figure 9 (f) gives the maximum, minimum, and average processing times for each of the five cases TP, TN, FP, ER, and FN. The maximum time (21914.43 sec) involved one of the TN cases. Among all the TN cases, around 25% of cases (4,118 data samples out of 16,492) required more than 60 seconds. This was because the full set of iterative calculations within nested loops were required (Lines 8 and 14 in Algorithm 1) to determine that the dataset had not been modified (i.e., the TN case). Because this is the worst case, the computational complexity of Algorithm 1 can be given as $O(n\Delta y)$. Similarly, more processing time for the FP cases was required when α approached 2 because α_s and α_t approach 2 only for the final iterations of Line 8 in Algorithm 1. Among all the FP cases, around 11.35% of cases (398 out of 3,508) required more than 60 seconds. This issue could be a limitation of Algorithm 1 for some applications but we believe this issue could be addressed adequately by scaling up the GPU-based parallel processing.

VI. CONCLUSION

In this paper, we formalize and analyze an integer data manipulation problem inspired by a recent voting scandal where ballot counts were fabricated by multiplication with a large nonintegral real number [2]. Because the multiplied numbers are rounded to integers, recovering the multiplicative factor and the original data is not easy. Based on our mathematical model and analysis, we develop an algorithm that can determine, with high probability, if a given integer dataset has been manipulated by multiplication.

If the integer dataset has actually been multiplied, Algorithm 1 reports this fact and returns a narrow interval containing the multiplied nonintegral real number. With this information, the original data can be restored accurately using Algorithm 2. Conversely, if the integer dataset is genuine, our algorithm also reports that fact with high probability.

To check the validity of our algorithm, we tested 20,000 multiplied integer datasets (the experimental group) and 20,000 genuine integer datasets (the control group), which were randomly generated within parameter value settings for the multiplicative factor α, the number of data entries n, and the range of data values. In these experiments, our algorithm found correct answers perfectly under conservative assumptions such as $\alpha \geq 16$ and $n \geq 40$. To the best of our knowledge, our mathematical analysis and Algorithm 1 represent the first attempt to address the manipulation process discussed in this paper.

APPENDIX A
PROOF OF LEMMA 1

Proof: First, suppose that α is a proper multiplicative factor with respect to y_i. We first claim that

$$z_i = x_i$$

for $i = 1, 2, \ldots, n$. Recalling that $\widetilde{y}_i = \alpha x_i$, Definitions (1) and (6) give

$$|z_i - x_i| = \left|\frac{y_i}{\alpha} \pm \frac{1}{2} - \frac{\widetilde{y}_i}{\alpha}\right| \leq \frac{|y_i - \widetilde{y}_i|}{\alpha} + \frac{1}{2} \leq \frac{1}{2\alpha} + \frac{1}{2} < 1,$$

where $a = b \pm \varepsilon$ means $b - \varepsilon \leq a \leq b + \varepsilon$. Therefore,

$$\widetilde{z}_i - z_i = \widetilde{z}_i - x_i = \frac{y_i - \widetilde{y}_i}{\alpha},$$

and (7) therefore holds.

From the other direction, suppose that (7) holds for $1 \leq i \leq n$. If $x_i = z_i,$, then

$$\alpha z_i = \alpha x_i = \widetilde{y}_i.$$

The definition $\alpha\widetilde{z}_i = y_i$ gives $\widetilde{y}_i - y_i = \alpha(z_i - \widetilde{z}_i)$. Applying the condition (7) gives

$$-\frac{1}{2} \leq \widetilde{y}_i - y_i < \frac{1}{2},$$

(i.e., $y_i = round(\widetilde{y}_i) = round(\alpha x_i)$). Following this, y_i is obtained by multiplication of the factor α by the original data $x_i = z_i$, whereby α is proper with respect to y_i. ■

Figure 12 is an intuitive presentation of this argument.

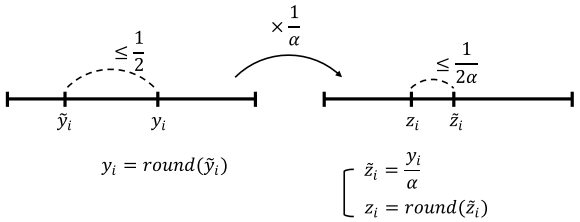


FIGURE 12. Change of intervals by multiplying with $1/\alpha$.

We can shed light on how hard it is to satisfy condition (7) for $1 \leq i \leq n$ in cases where the real numbers $\widetilde{z}_i$ are independently and randomly chosen from a uniform distribution of $[1, N]$ for a sufficiently large integer N. Clearly, $\lceil\widetilde{z}_i - z_i\rceil$ is uniform within $[0, 1/2]$, and it follows that

$$\mathbb{P}\lceil\widetilde{z}_i - z_i\rceil \leq \frac{1}{2\alpha} = \frac{1}{\alpha}$$

for $i = 1, 2, \ldots, n$. Because the $\widetilde{z}_i$ values are independent, we have

$$\mathbb{P}\lceil\widetilde{z}_i - z_i\rceil \leq \frac{1}{2\alpha} \quad \text{for } 1 \leq i \leq n = \frac{1}{\alpha^n} \quad \text{for independently random } \widetilde{z}_i. \tag{11}$$

This shows that for randomly chosen data, it becomes harder to satisfy condition (7) for $1 \leq i \leq n$ in Lemma 1 as α and n increase. This tendency is consistent with the experimental results described in Section V-C1.

APPENDIX B
PROOF OF LEMMA 2

Proof: Two consecutive intervals of the form $\left[\frac{y_i-0.5}{j}, \frac{y_i+0.5}{j}\right]$ can be written as

$$\left[\frac{y_i - 0.5}{j}, \frac{y_i + 0.5}{j}\right] \quad \text{and} \quad \left[\frac{y_i - 0.5}{j-1}, \frac{y_i + 0.5}{j-1}\right].$$

Assuming

$$\alpha_{j+} = \frac{y_i + 0.5}{j} \geq 2,$$
$$y_i + 0.5 - j \geq j > 0,$$

it follows that

$$\frac{y_i + 0.5}{j} - \frac{y_i - 0.5}{j-1} = \frac{-y_i - 0.5 + j}{j(j-1)} < 0,$$

and we can infer that two consecutive intervals of the form $\left[\frac{y_i-0.5}{j}, \frac{y_i+0.5}{j}\right]$ do not overlap. ■

APPENDIX C
INITIAL ESTIMATE FOR INEQUALITY (10)

Supposing that the y_i values are not multiplied, we can form an initial estimate for the value

$$\sum_{1 \leq \ell \leq \frac{\Delta y - 1}{2}} \mathbb{P}\alpha \in I_\ell^* \mid \alpha \in I_\ell$$

for an ideal situation. We start with

$$|I_\ell| = \frac{2}{\ell} \quad \text{and } |I^*(i, \ell)| = \sum_{j \in J(i,\ell)} \frac{1}{j}.$$

We first estimate $|I^*(i,\ell)|$ for each $1 \le i \le n$. Recalling that $j \in J(i,\ell)$ is equivalent to $\frac{y_i-0.5}{\alpha_t} \le j < \frac{y_i+0.5}{\alpha_s}$, we have

$$\begin{aligned}\mathbb{E}\left(|J(i,\ell)|\right) &= \frac{y_i+0.5}{\alpha_s} - \frac{y_i-0.5}{\alpha_t} \\ &= \frac{\ell(y_i+0.5)}{\Delta y-1} - \frac{\ell(y_i-0.5)}{\Delta y+1} \\ &= \frac{\ell(2y_i+\Delta y)}{(\Delta y-1)(\Delta y+1)}.\end{aligned}$$

Consequently,

$$\begin{aligned}|I^*(i,\ell)| &= \sum_j \frac{1}{j} \le \frac{\ell(2y_i+\Delta y)}{(\Delta y-1)(\Delta y+1)} \cdot \frac{\Delta y+1}{\ell(y_i-0.5)} \\ &= \frac{2(2y_i+\Delta y)}{(\Delta y-1)(2y_i-1)}.\end{aligned}$$

Therefore,

$$\mathbb{P}\alpha \in I^*(i,\ell) \mid \alpha \in I_\ell = \frac{|I^*(i,\ell)|}{|I_\ell|} \le \frac{\ell(2y_i+\Delta y)}{(\Delta y-1)(2y_i-1)}.$$

To refine this estimate, suppose that $I^*(i,\ell)$ is independent of i. Recalling that $I^*_\ell := \bigcap_i I^*(i,\ell)$, we have

$$\begin{aligned}\mathbb{P}\alpha \in I^*_\ell \;\big|\; \alpha \in I_\ell &= \prod_{1\le i\le n} \mathbb{P}\alpha \in I^*(i,\ell) \mid \alpha \in I_\ell \\ &\le \prod_{1\le i\le n} \frac{\ell(2y_i+\Delta y)}{(\Delta y-1)(2y_i-1)} \\ &= \left(\frac{\ell}{\Delta y-1}\right)^n \prod_{1\le i\le n} \frac{2y_i+\Delta y}{2y_i-1}.\end{aligned}$$

Therefore, with disjoint I_ℓ, we have

$$\begin{aligned}\sum_{1\le \ell\le \frac{\Delta y-1}{2}} \mathbb{P}\alpha \in I^*_\ell \;\big|\; \alpha \in I_\ell &\le \left(\frac{1}{\Delta y-1}\right)^n \prod_{1\le i\le n} \frac{2y_i+\Delta y}{2y_i-1} \\ &\quad \cdot \sum_{1\le \ell\le \frac{\Delta y-1}{2}} \ell^n \\ &\le \left(\frac{1}{\Delta y-1}\right)^n \prod_{1\le i\le n} \frac{2y_i+\Delta y}{2y_i-1} \cdot \int_0^{\Delta y/2} x^n dx \\ &\le \frac{(\Delta y)^{n+1}}{(n+1)2^{n+1}(\Delta y-1)^n} \cdot \prod_{1\le i\le n} \frac{2y_i+\Delta y}{2y_i-1} \\ &\le \frac{(\Delta y)^{n+1}}{(n+1)2^{n+1}(\Delta y-1)^n} \cdot \prod_{1\le i\le n} \frac{2+\frac{\Delta y}{y_i}}{2-\frac{1}{y_i}}.\end{aligned}$$

ACKNOWLEDGMENT

The authors would like to thank Taehwan Yoon, Jaehee Kim, Jee Yong Chung, Moony Lee, Woo-Hyun Chung, Eric Choi, and Mark Siggers for their valuable comments.

REFERENCES

[1] *Marshall van Alstyne (MIT) at Emerce eDay 2013*, EMERCE, Amsterdam, The Netherlands, 2013. [Online]. Available: https://www.youtube.com/watch?v=vYxplaWong8

[2] *Mnet Vote Manipulation Investigation*, Wikipedia, Nov. 2020. [Online]. Available: https://en.wikipedia.org/wiki/Mnet_vote_manipulation_investigation

[3] R. Aragão and L. Linsi, "Many shades of wrong: What governments do when they manipulate statistics," *Rev. Int. Political Economy*, May 2020. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/09692290.2020.1769704, doi: 10.1080/09692290.2020.1769704.

[4] S. A. Ross, "The economic theory of agency: The principal's problem," *Amer. Econ. Rev.*, vol. 63, no. 2, pp. 134–139, May 1973.

[5] B. M. Mitnick, "Fiduciary rationality and public policy: The theory of agency and some consequences," in *Proc. Annu. Meeting Amer. Political Sci. Assoc.*, New Orleans, LA, USA, 1973, doi: 10.2139/ssrn.1020859.

[6] J. L. Wallace, "Juking the stats? Authoritarian information problems in China," *Brit. J. Political Sci.*, vol. 46, no. 1, pp. 11–29, Jan. 2016.

[7] C. Lyu, K. Wang, F. Zhang, and X. Zhang, "GDP management to meet or beat growth targets," *J. Accounting Econ.*, vol. 66, no. 1, pp. 318–338, Aug. 2018.

[8] J. E. Mosimann, C. V. Wiseman, and R. E. Edelman, "Data fabrication: Can people generate random digits?" *Accountability Res.*, vol. 4, no. 1, pp. 31–55, 1995.

[9] S. Newcomb, "Note on the frequency of use of the different digits in natural numbers," *Amer. J. Math.*, vol. 4, nos. 1–4, p. 39, 1881.

[10] T. Grammatikos and N. I. Papanikolaou, "Applying Benford's law to detect accounting data manipulation in the banking industry," *J. Financial Services Res.*, vol. 59, nos. 1–2, pp. 115–142, May 2020.

[11] J. C. Collins, "Using excel and benford's law to detect fraud: Learn the formulas, functions, and techniques that enable efficient benford analysis of data sets," *J. Accountancy*, vol. 223, no. 4, p. 44, Apr. 2017.

[12] M. Jerven, *Poor Numbers: How we are Misled by African Development Statistics and What to do About it*. Ithaca, NY, USA: Cornell Univ. Press, 2013.

[13] M. C. Jensen and W. H. Meckling, "Theory of the firm: Managerial behavior, agency costs and ownership structure," *J. Financial Econ.*, vol. 3, no. 4, pp. 305–360, Oct. 1976.

[14] S. D. Young and H. Sherman, "Where financial reporting still falls short," *Harvard Bus. Rev.*, vol. 94, no. 7, p. 17, Jul. 2016.

[15] M. Schreyer, T. Sattarov, B. Reimer, and D. Borth, "Adversarial learning of deepfakes in accounting," Oct. 2019, *arXiv:1910.03810*. [Online]. Available: http://arxiv.org/abs/1910.03810

[16] P. Klimek, Y. Yegorov, R. Hanel, and S. Thurner, "Statistical detection of systematic election irregularities," *Proc. Nat. Acad. Sci. USA*, vol. 109, no. 41, pp. 16469–16473, Oct. 2012.

[17] P. Klimek, R. Jiménez, M. Hidalgo, A. Hinteregger, and S. Thurner, "Forensic analysis of turkish elections in 2017–2018," *PLoS ONE*, vol. 13, no. 10, Oct. 2018, Art. no. e0204975.

[18] R. Jiménez and M. Hidalgo, "Forensic analysis of venezuelan elections during the Chávez presidency," *PLoS ONE*, vol. 9, no. 6, Jun. 2014, Art. no. e100884.

[19] J. Deckert, M. Myagkov, and P. C. Ordeshook, "Benford's law and the detection of election fraud," *Political Anal.*, vol. 19, no. 3, pp. 245–268, 2011.

[20] I. Levin, J. Pomares, and R. M. Alvarez, "Using machine learning algorithms to detect election fraud," in *Computational Social Science: Discovery and Prediction* (Analytical Methods for Social Research), R. M. Alvarez, Ed. Cambridge, U.K.: Cambridge Univ. Press, 2016, pp. 266–294.

[21] M. Zhang, R. M. Alvarez, and I. Levin, "Election forensics: Using machine learning and synthetic data for possible election anomaly detection," *PLoS ONE*, vol. 14, no. 10, Oct. 2019, Art. no. e0223950.

[22] P. Fischer, "New research misconduct policies," Nat. Sci. Found., Sep. 2012. [Online]. Available: https://www.nsf.gov/oig/_pdf/presentations/session.pdf

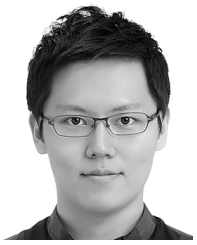
[23] J. E. Dahlberg and N. M. Davidian, "Scientific forensics: How the office of research integrity can assist institutional investigations of research misconduct during oversight review," *Sci. Eng. Ethics*, vol. 16, no. 4, pp. 713–735, Dec. 2010.

[24] D. Fanelli, "How many scientists fabricate and falsify research? A systematic review and meta-analysis of survey data," *PLoS ONE*, vol. 4, no. 5, p. e5738, May 2009.

[25] R. M. Fleming, M. R. Fleming, and T. K. Chaudhuri, "Establishing data validity: Statistically determining if data is fabricated, falsified or plagiarized," *Acta Sci. Med. Sci.*, vol. 3, no. 8, pp. 169–191, Jul. 2019.

[26] N. Persaud, "Humans can consciously generate random number sequences: A possible test for artificial intelligence," *Med. Hypotheses*, vol. 65, no. 2, pp. 211–214, Jan. 2005.
[27] M. Figurska, M. Stańczyk, and K. Kulesza, "Humans cannot consciously generate random numbers sequences: Polemic study," *Med. Hypotheses*, vol. 70, no. 1, pp. 182–185, Jan. 2008.
[28] J. Mosimann, J. Dahlberg, N. Davidian, and J. Krueger, "Terminal digits and the examination of questioned data," *Accountability Res.*, vol. 9, no. 2, pp. 75–92, Apr. 2002, doi: 10.1080/08989620212969.
[29] B. Beber and A. Scacco, "What the numbers say: A digit-based test for election fraud," *Political Anal.*, vol. 20, no. 2, pp. 211–234, 2012.
[30] M. L. Overton, *Numerical Computing With IEEE Floating Point Arithmetic*, 1st ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, Apr. 2001.
[31] *IEEE Standard for Interval Arithmetic (Simplified)*, Standard IEEE 1788.1-2017, 2017.
[32] N. Revol, "Influence of the condition number on interval computations: Illustration on some examples," in *Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy etc. Methods and Their Applications* (Studies in Computational Intelligence), vol. 835, O. Kosheleva, S. P. Shary, G. Xiang, and R. Zapatrin, Eds. Cham, Switzerland: Springer, 2020, pp. 359–373.
[33] R. M. Layer and A. R. Quinlan, "A parallel algorithm for N-way interval set intersection," *Proc. IEEE. Inst. Electr. Electron. Eng.*, vol. 105, no. 3, pp. 542–551, Mar. 2017.
[34] *Boost C++ Library Class Template Interval_Set*, Boost, 2020. [Online]. Available: https://www.boost.org/doc/libs/1_49_0/libs/icl/doc/html/boost/icl/interval_set.html

TAEJUNG PARK received the B.A. and M.S. degrees in electrical engineering and the Ph.D. degree for 3-D mesh compression from Seoul National University, in 1997, 1999, and 2006, respectively. He designed a data structure for a direct matrix solver for FEM simulation for semiconductor devices for his master's degree. He has worked with two small start-up technology businesses in South Korea, in 1999 and 2002. He has been an Associate Professor with the Department of Cybersecurity/IT Media, Duksung Women's University, since 2013. His current research interests include parallel numerical simulation techniques and nonlinear interpolation methods from the viewpoint of information technology.



HYUNJOO SONG received the B.S. degree in computer science and engineering and the M.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, Korea, in 2009, 2011, and 2016, respectively. He was an Assistant Professor with the Department of IT Media, Duksung Women's University, from 2017 to 2020. He has been an Assistant Professor with the School of Computer Science and Engineering, Soongsil University, Seoul, since 2020. His research interests include human–computer interaction, information visualization, and gaze tracking.

SANG JUNE LEE received the B.S. degree in mathematics from Seoul National University, South Korea, in 2002, and the Ph.D. degree in mathematics from Emory University, Atlanta, GA, in 2012. From 2012 to 2014, he was the Visiting Scientist and a Postdoctoral Fellow with the Korea Institute for Advanced Study (KIAS) and ASARC, Korea Advanced Institute of Science and Technology (KAIST), respectively. From 2014 to 2019, he was an Assistant Professor with the Department of Mathematics, Duksung Women's University, South Korea. Since 2019, he has been an Associate Professor with the Department of Mathematics, Kyung Hee University, South Korea. His research interests include extremal combinatorics and probabilistic combinatorics about graph theory, and combinatorial number theory and set theory.