Information Sciences 348 (2016) 377–393

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Generalized bucketization scheme for flexible privacy settings

Ke Wang[a,*], Peng Wang[a], Ada Waichee Fu[b], Raymond Chi-Wing Wong[c]

[a] School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
[b] Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong
[c] Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong

ARTICLE INFO

Article history:
Received 16 January 2015
Revised 4 November 2015
Accepted 27 January 2016
Available online 17 February 2016

Keywords:
Anonymity
Bucketization
Data publishing
Disclosure
Privacy

ABSTRACT

Bucketization is an anonymization technique for publishing sensitive data. The idea is to group records into small buckets to obscure the record-level association between sensitive information and identifying information. Compared to the traditional generalization technique, bucketization does not require a taxonomy of attribute values, so is applicable to more data sets. A drawback of previous bucketization schemes is the uniform privacy setting and uniform bucket size, which often results in a non-achievable privacy goal or excessive information loss if sensitive values have variable sensitivity.

In this work, we present a flexible bucketization scheme to address these issues. In the flexible scheme, each sensitive value can have its own privacy setting and buckets of different sizes can be formed. The challenge is to determine proper bucket sizes and group sensitive values into buckets so that the privacy setting of each sensitive value can be satisfied and overall information loss is minimized. We define the *bucket setting problem* to formalize this requirement. We present two efficient solutions to this problem. The first solution is optimal under the assumption that two different bucket sizes are allowed, and the second solution is heuristic without this assumption. We experimentally evaluate the effectiveness of this generalized bucketization scheme.

1. Introduction

1.1. Motivation

Privacy preserving data publishing is concerned with publishing sensitive data for data analysis while ensuring that no sensitive information about individuals is disclosed. The next example illustrates how sensitive information may be disclosed by answering count queries.

Example 1. A hospital provides online services for answering count queries on medical data T containing three attributes, Gender, Zipcode, and Disease. Disease is sensitive and must not be disclosed, and Gender and Zipcode are public. Suppose that the patient Alice has a record in the data and that an adversary tries to learn the value of Disease for Alice. Knowing that Alice's Zipcode is 61434, the adversary issues two queries Q_1 and Q_2:

* Corresponding author. Tel.: +1 7787824667.
E-mail addresses: wangk@cs.sfu.ca (K. Wang), pwa22@sfu.ca (P. Wang), adafu@cse.cuhk.edu.hk (A.W. Fu), raywong@cse.ust.hk (R.C.-W. Wong).

http://dx.doi.org/10.1016/j.ins.2016.01.100
0020-0255/

Q_1: SELECT COUNT(*) FROM T WHERE Gender=F AND Zipcode=61434
Q_2: SELECT COUNT(*) FROM T WHERE Gender=F AND Zipcode=61434 AND Disease=HIV

Let x and y be the answers for Q_1 and Q_2, i.e., the number of records matching the description in the WHERE clause. Further assume $x \neq 0$. Such answers are building blocks for many data analysis tasks such as constructing Naive Bayes classifiers. However, an adversary could also use these answers to learn that Alice has HIV with the probability $\frac{y}{x}$.

The above example illustrates a sensitive disclosure through *non-independent reasoning*, where Alice's disease information is learnt from *other* people who share the same Gender and Zipcode as Alice, under the assumption that the diseases of those people follow the same underlying distribution. Non-independent reasoning has a great success in many applications, including classification, prediction, direct marketing, and product recommendation. For example, in classification we learn the class information of a new case from a training data set. Preventing sensitive non-independent reasoning has been a focus of *syntactic models* in the literature, where the raw data is modified, usually by generalization and suppression, in order to bound the change of adversary's beliefs after accessing published information. Some representative syntactic models are k-anonymization [17], ρ_1–ρ_2 privacy [9], ℓ-diversity [16], t-closeness [14], β-likeness [3], Δ-growth [18], to name a few. See surveys [1,4,10] for more details.

As an alternative to syntactic models, the *differential privacy* approach [7,8] does not release the data set itself but releases the answers to user queries using the data set, and instead of preventing non-independent reasoning, it seeks to mask the impact of a single individual by releasing a noisy query answer $x + \Delta x$, where x is the true answer and Δx is the added noise such that the distribution of noisy answers changes little with and without the participation of a single individual. A recent study [11,20] showed that differential privacy is in the following dilemma: while a more restricted setting of ϵ-differential privacy (i.e., a small ϵ) helps protect privacy, the noisy answers have a poor utility for data analysis; while a less restricted setting provides a good utility for data analysis, this good utility also permits sensitive disclosures of non-independent reasoning. The root of this dilemma is that both data analysis and sensitive disclosures are making use of the same information, i.e., noisy answers. Wang et al. [20] shows that with the noises Δy and Δx generated by the ϵ-differential privacy mechanism for the query answers y and x in Example 1, $\frac{y+\Delta y}{x+\Delta x}$ arbitrarily approaches the noise-free probability $\frac{y}{x}$ as the data set grows arbitrarily large. The following quotes from two pioneering works on differential privacy also suggest that differential privacy does not address the privacy violation due to non-independent reasoning.

> Page 3 of [2]: "We explicitly consider nonindependent reasoning as a non-violation of privacy; information that can be learned about a row from sources other than the row itself is not information that the row could hope to keep private."
>
> Page 8 of [7]: "Note that a bad disclosure can still occur, but our guarantee assures the individual that it will not be the presence of her data that causes it, nor could the disclosure be avoided through any action or inaction on the part of the user."

According to [2] and [7], the main reason that a disclosure due to non-independent reasoning is not considered as privacy violation is that such disclosures cannot be avoided through any action or inaction on the part of the user. For this reason, the disclosure of Alice's diseases through $\frac{y+\Delta y}{x+\Delta x}$, when it approaches $\frac{y}{x}$, is not considered as privacy violation by differential privacy, though in practice Alice may not agree with this. Therefore, differential privacy is not suitable when non-independent reasoning does violate an individual's privacy.

In this work, we consider non-independent reasoning of sensitive information as a privacy violation. Under this assumption, syntactic methods through data suppression, generalization, and bucketization remain relevant. See surveys [1,4,10] for more discussions on syntactic methods. Data *suppression* has a large information loss for count queries because suppressed records or values cannot be counted by the query. Data *generalization* is applicable only when there is a taxonomy for each public attributes. Also, generalized data cannot be easily analyzed by standard methods. For example, to reduce the information loss in local recoding, the value "Engineer" could be generalized into the high level value "Professional" in some records while it remains unchanged in other records. Consequently, "Professional" and "Engineer" cannot be treated as two distinct values for counting, which makes it impossible to apply any standard counting based data mining methods such as Naive Beyes classifiers. Data *bucketization* does not have the above problems because it does not generalize domain values. In this work, we shall focus on the bucketization approach.

Unfortunately, previous bucketization schemes [22] have some major drawbacks. First, all previous bucketization schemes use a uniform privacy setting for all sensitive values. For example, the Anatomy algorithm in [22] uses ℓ-diversity to specify the privacy criterion that the frequency of HIV and Flu in a bucket is no more than $1/\ell$. Suppose that HIV is more sensitive than Flu, where Flu occurs more frequently than HIV. ℓ-diversity must be set according to the sensitivity of HIV, i.e., a large ℓ so that $1/\ell$ is small enough for HIV. However, often this setting is not satisfiable for Flu that has a higher frequency in the data. Another drawback of previous bucketization schemes is the uniform size for all buckets, i.e., either ℓ or $\ell + 1$ in [22]. In the example of HIV and Flu, ℓ that is set according to the most sensitive HIV can be very large. A larger bucket size means less association of a record with its original sensitive value in the bucket, thus, more information loss. More discussions about these points will be presented in Section 3.

Table 1
Notations.

T, $\|T\|$	The raw data and its cardinality
m	Domain size of SA
x_i	A sensitive value
o_i	Number of occurrence of x_i in T
f_i	$o_i/\|T\|$
f'_i	Privacy threshold for x_i
F'-privacy	A collection of f'_i for all x_i
$B_j(S_j, b_j)$	b_j buckets of size S_j
$s(B_j)$	Total size of buckets in B_j

1.2. Contributions

The goal of this paper is to address the above drawbacks of previous bucketization schemes. Our contributions are as follows.

Contribution 1. In Section 3, we formalize a generalized bucketization problem, called *bucket setting problem*, in which each sensitive value has its own privacy setting and buckets of different sizes can be formed. The goal is to find the optimal bucket setting, i.e., the minimum bucket size for each bucket that is necessary to provide the specified privacy settings for all the sensitive values within the bucket.

Contribution 2. We present an efficient *optimal* solution under the two-size assumption that two different bucket sizes (but many buckets) are allowed. This solution has two parts. In Section 4, the first part tests whether a *given* bucket setting has a valid record assignment to the bucket while satisfying the specified privacy setting. In Section 5, the second part searches for an optimal bucket setting.

Contribution 3. For the general case without the two-size assumption, we present two solutions in Section 6. The first solution is based on integer linear programming. While guaranteeing optimality, this solution works only for data of a small size. The second solution is heuristic and is built using the optimal solution to the two-size bucket setting problem in Contribution 2. This solution is scalable to a large data size.

Contribution 4. In Section 7, we evaluate the above solutions against the claim that the generalized bucketization scheme could enforce variable privacy settings with less information loss.

2. Related work

To our knowledge, most existing works consider a uniform privacy specification for all sensitive values with the exceptions of personalized privacy [23], confidence bounding [19], ℓ^+-diversity [15], and β-likeness [3]. Our privacy definition is similar to the flexible privacy settings proposed in these works. The difference is in the enforcement of a flexible privacy setting. In particular, [3,15,23] generalize the data on public attributes using a taxonomy on attributes, and [15,19] suppress values on sensitive attributes. As discussed earlier, the suppression approach incurs a large information loss for count queries and the generalization approach is not applicable if attribute taxonomies are not available. We consider the bucketization approach because this approach does not require any attribute taxonomy and bucketized data can be processed by standard data mining methods (after flattening bucketized data into a set of records). To our knowledge, this is the first bucketization scheme that enforces a flexible privacy setting using variable bucket sizes to reduce information loss.

One concern with syntactic models is that they are susceptible to several attacks such as the deFinetti attack [12] and a similar attack in [21]. Under certain conditions, such attacks permit the adversary to infer sensitive information with a probability higher than $1/\ell$ (for ℓ-diversity) by an analysis of the global distribution of all buckets. As shown in [6,12], the effectiveness of the deFinetti attack diminishes substantially when the size buckets grow. In this regard, our generalized bucketization scheme is more immune to the deFinetti attack because a larger bucket is used for a more sensitive value. The study in [5] argued that the deFinetti attack is not a solid argument to abandon syntactic privacy models in favor of differential privacy, and the study in [6] suggested that syntactic models are no more susceptible to the deFinetti attack than differential privacy. The bottom line is that syntactic privacy targets privacy violation of non-independent reasoning whereas differential privacy does not; if the data owner considers non-independent reasoning as privacy violation, syntactic privacy models are the only option.

3. Problem statement

To define the generalized bucketization problem, we start with the basic bucketization scheme. Consider a microdata table of the form $T(QI, SA)$, where QI is the quasi-identifier consisting of a set of public attributes $\{A_1, \ldots, A_d\}$ and SA is a sensitive attribute with the domain $\{x_1, \ldots, x_m\}$. o_i denotes the number of records in T for x_i and $f_i = o_i/|T|$ is the frequency of x_i, where $|T|$ is the cardinality of T. For a record r in T, $r[QI]$ and $r[SA]$ denote the values of r on QI and SA. Table 1 lists the notations used in this paper.

Table 2
A table T.

Gender	Zipcode	Disease
Male	54321	Brain Tumor
Male	54322	Indigestion
Female	61234	Cancer
Female	61434	HIV

Table 3
A bucketization T^* satisfying 2-diversity.

Gender	Zipcode	BID
M	54321	1
M	54322	1
F	61234	2
F	61434	2

(a) *QIT*

BID	Disease
1	Brain Tumor
1	Indigestion
2	Cancer
2	HIV

(b) *SAT*

An adversary wants to learn the *SA* value of an individual t whose record is in T through the access to a published version of T, denoted T^*. The adversary knows the public attributes of t, $t[QI]$, as well as the algorithm that was used to produce T^*. The disclosure of learning a *SA* value, x_i, is measured by the probability that t is inferred to have x_i, given $t[QI]$ and T^*. This probability is denoted by $Pr(x_i|t, T^*)$.

3.1. The basic bucketization scheme

Xiao and Tao [22] presented a bucketization scheme to limit $Pr(x_i|t, T^*)$ based on the privacy criterion specified by l-diversity [16]. This scheme partitions records into buckets such that the records in the same bucket have a diverse set of *SA* values. In particular, T^* is represented by two tables: $QIT(QI, BID)$ and $SAT(BID, SA)$, where *BID* denotes the bucket id. If a record r belongs to a bucket with the bucket id g, *QIT* contains a record $(r[QI], g)$ and *SAT* contains a record $(g, r[SA])$ (duplicates are preserved). Suppose that $t[QI]$ is contained in a bucket g. $Pr(x_i|t, g) = |g, x_i|/|g|$ is the probability for t to have x_i in g, where $|g, x_i|$ is the number of occurrence of (g, x_i) in *SAT* and $|g|$ is the size of the bucket g. $Pr(x_i|t, T^*)$ is the maximum $Pr(x_i|t, g)$ for any bucket g containing $t[QI]$ [22].

The following is an instantiation of the ℓ-diversity principle in [16].

Definition 1. T^* satisfies ℓ-diversity if for every individual t with a record in T and for every x_i, $Pr(x_i|t, T^*) \leq 1/\ell$. (Eligibility condition) There is a T^* that satisfies ℓ-diversity if and only if the maximum frequency of any x_i in T, $max_i f_i$, is no more than $1/\ell$ [22].

The eligibility condition says that if some sensitive value x_i is "too frequent", i.e., $max_i f_i > 1/\ell$, ℓ-diversity cannot be satisfied, that is, no T^* can be published without violating ℓ-diversity. If the eligibility condition holds, the Anatomy algorithm in [22] can find an optimal T^* that satisfies ℓ-diversity. A consequence of this algorithm is that all buckets in their T^* have a uniform size that is either ℓ or $\ell + 1$.

Example 2. Consider the microdata T in Table 2 where Gender and Zipcode are the *QI* attributes and Disease is *SA*. Table 3 shows *QIT* and *SAT* for one bucketization. To learn the *SA* value of Alice with $QI = \langle F, 61434\rangle$, the adversary first locates the buckets that contain $\langle F, 61434\rangle$, i.e., bucket 2 identified by $BID = 2$. There are two diseases in this bucket, Cancer and HIV, each occurring once. So $Pr(x_i|Alice, BID = 2) = 50\%$, where x_i is either Cancer or HIV. Since $\langle F, 61434\rangle$ is contained only in bucket 2, $Pr(x_i|Alice, T^*) = 50\%$. Similarly, $Pr(x_i|t, T^*) = 50\%$ for every individual t in T. So T^* satisfies 2-diversity. Since the maximum frequency $max_i f_i$ is 1/4, the maximum ℓ such that ℓ-diversity can be satisfied is 4.

3.2. Utility metrics

Count queries. In this paper, we consider the utility of answering count queries such as in Example 1. Count queries are important building blocks for many data analysis tasks, such as contingent tables, correlation analysis, mining frequent itemsets, building decision tree or naive Bayes classifiers. Since our bucketization preserves exactly the counts for a count query involving only public attributes, information loss occurs only for a count query involving both public attributes and the sensitive attribute, which has the form [22]:

SELECT COUNT(∗) FROM T
WHERE $pred(A_1)$ AND ... AND $pred(A_{q_d})$ AND $pred(SA)$
$A_1, \ldots, A_{q_d}$ are *QI*-attributes. For any attribute A, $pred(A)$ has the form

$A = a_1$ OR ... OR $A = a_b$,

where a_i is a value from the domain of A. The answer *act* to a count query Q is the number of records in T that satisfy the condition in the WHERE clause.

We have to estimate the answer *act* using the published T^* given by a collection of buckets g_j stored in two tables *QIT* and *SAT*. Let $g_j(QIT)$ and $g_j(SAT)$ denote the set of records in *QIT* and the set of records in *SAT* for g_j. Let $c(g_j, pred(SA))$ be the number of records in $g_j(SAT)$ that satisfies $pred(SA)$ and let $c(g_j, pred(A_1), \ldots, pred(A_{q_d}))$ be the number of records in $g_j(QIT)$ that satisfies $pred(A_1)$ AND ... AND $pred(A_{q_d})$. We say that a bucket g_j matches a count query Q if $c(g_j, pred(SA)) > 0$ and $c(g_j, pred(A_1), \ldots, pred(A_{q_d})) > 0$. Let $g_1, \ldots, g_k$ be the matching buckets of Q. The estimated answer *est* is given by Xiao and Tao [22]

$$\sum_j c(g_j, pred(A_1), \ldots, pred(A_{q_d})) \times \frac{c(g_j, pred(SA))}{|g_j|}$$

This estimation assumes that each record in $g_j(QIT)$ has an equal chance, i.e., $\frac{c(g_j, pred(SA))}{|g_j|}$, to be associated with each occurrence of a *SA* value in $g_j(SAT)$. The *relative error* of *est* is $|act - est|/act$.

We need a loss metric to guide the search of T^* that aims to reduce the relative error *without knowing all the queries in advance*. There are several loss metrics in the literature. Our choice is based on the observation that the larger a bucket is, the more records are contained in the bucket and the less likely a record will be associated with its own x_i value in the bucket. Therefore, a simple heuristic of the loss metric is the sizes of buckets. The next definition formalizes this loss metric.

Definition 2. Let T^* contain buckets $g_1, \ldots, g_b$. The Mean Squared Bucket Size (MSBS) of T^* is defined by

$$MSBS(T^*) = \frac{\sum_{i=1}^{b}(|g_i| - 1)^2}{|T| - 1} \tag{1}$$

Intuitively, *MSBS* is the average bucket size minus one (per record). At one extreme, the raw data T can be considered as T^* in which each record itself forms a bucket of size one, and $MSBS = 0$. At the other extreme, for the single bucket containing all the records in T, $MSBS = |T| - 1$. *MSBS* is in the range $[0, |T| - 1]$. For the basic bucketization scheme satisfying ℓ-diversity, *MSBS* is between $\ell - 1$ and ℓ because a bucket has a size of ℓ or $\ell - 1$. Since each record in a bucket has an equal chance to be associated with each occurrence of a *SA* value in the bucket, a smaller *MSBS* value means that a record belongs to a smaller bucket, thus, has more chance to be associated with its own *SA* value when estimating the answer to a count query using buckets. For example, $MSBS = 0$ means that each record is always associated with its own *SA* value in a bucket because it is the only record in its bucket. Thus, minimizing *MSBS* would increase the chance of a record being associated with his own *SA* value in a bucket.

With $|T|$ being fixed, minimizing *MSBS* is equivalent to minimizing the following loss

$$Loss(T^*) = \sum_{i=1}^{b}(|g_i| - 1)^2 \tag{2}$$

Loss has the following additivity property: if T^* has a partition $\{T_1^*, T_2^*\}$, $Loss(T^*) = Loss(T_1^*) + Loss(T_2^*)$. Thus, an optimal solution T^* consists of an optimal solution T_1^* and an optimal solution T_2^*. This property helps decompose a larger problem into smaller ones. In the rest of the paper, we shall use *Loss* as our loss metric in the search of bucketization T^*.

One may note that *MSBS* is completely determined by the sizes of buckets and does not depend on the actual distribution of x_i values in a bucket. This decision is a tradeoff between the complexity of search and the accuracy of information loss because minimizing the bucket sizes is much simpler than minimizing other loss metrics that consider the actual distribution of x_i values in a bucket. In particular, we will show that, to minimize *MSBS* while satisfying a privacy criterion, it suffices to consider *even distribution* of the occurrences of each x_i value across buckets because such distributions most effectively remove privacy violation. This property significantly simplifies the search of an optimal bucketization. More details are presented in Lemma 2.

3.3. Privacy metrics

The ℓ-diversity in Definition 1 uses a single privacy setting $1/\ell$ for all sensitive values x_i. Typically, a different sensitive value may require a different privacy setting for two reasons. First, some sensitive values are more sensitive than others by nature. Second, some sensitive values are less common than others. For example, HIV is less common and more sensitive than Flu, thus, a smaller threshold $1/\ell$, i.e., a larger ℓ, is required as compared to Flu. In this case, the single setting $1/\ell$ must be specified according to the more sensitive HIV, which is too small for Flu that has a large frequency in the data. This leads to an unsatisfiable siltation. Even if the ℓ-diversity is satisfiable for both values, the single ℓ chosen according to HIV is likely very large, so the basic bucketization scheme [22] produces the buckets of large sizes ℓ or $\ell + 1$. To address this issue, we consider a flexible privacy setting specification defined as follows. Recall that f_i denotes the actual frequency of x_i in T, i.e., $o_i/|T|$, whereas f_i' is a *publisher-specified* threshold for $Pr(x_i|t, T^*)$.

Definition 3 (F'**-Privacy**). For each *SA* value x_i, f_i'*-privacy* specifies the requirement that $Pr(x_i|t, T^*) \le f_i'$, where f_i' is a real in the range (0,1]. F'*-privacy* is a collection of f_i'-privacy for all *SA* values x_i.

ℓ-diversity is the special case of $f'_i = 1/\ell$ for all x_i values. In general, the specification of f'_i depends on the sensitivity of x_i, the frequency f_i, and the user's trade-off between privacy and utility. In general, f'_i can be specified semi-automatically. For example, the publisher may set $f'_i = 1$ for those x_i's that are not sensitive at all (such as Flu), set f'_i manually to a small value for highly sensitive x_i (such as HIV), and set f'_i to some function of f_i for those x_i whose thresholds are related to their frequencies in T. An example is $f'_i = min\{1, \theta \times f_i + 0.02\}$ for some constant θ. This linear relationship models that a less frequently occurring x_i value tends to be more sensitive (thus, have a smaller privacy threshold f'_i). For example, suppose that HIV is less common than Flu, we would expect that HIV has a smaller privacy threshold than Flu. The coefficient θ specifies the rate of this relationship.

The next lemma gives a necessary and sufficient condition for the existence of T^* satisfying a given F'-privacy.

Lemma 1 (Eligibility condition for F'-privacy)**.** *Given a table T and a specification of F'-privacy, there exists T^* satisfying F'-privacy if and only if $f'_i \geq f_i$ for all x_i.*

Proof. "If" follows from the fact that the single bucket T^* containing all records always satisfies F'-privacy if $f'_i \geq f_i$ for all x_i. "Only if" follows from the fact that, for each x_i, there is some bucket g in which the frequency of x_i is at least f_i, that is, $Pr(x_i|t, g) \geq f_i$. □

Remark 1. F'-privacy has two advantages over ℓ-diversity. First, by allowing each x_i value a different setting f'_i, a smaller bucket can be used for the records of x_i with a larger f'_i and a larger bucket is used only for records of x_i with a smaller f'_i. In contrast, the *corresponding* ℓ-diversity that enforces the same F'-privacy requires $1/\ell \leq min_i f'_i$, or $\ell = \lceil 1/min_i f'_i \rceil$. This ℓ, chosen according to the minimum f'_i, yields large buckets in the basic bucketization scheme [22] where every bucket has a size either ℓ or $\ell + 1$. The second advantage is that F'-privacy is easier to satisfy than the corresponding ℓ-diversity. From Definition 1 and Lemma 1, F'-privacy has a solution if and only if $f'_i \geq f_i$ for all x_i, and the corresponding ℓ-diversity has a solution if and only if $1/\ell \geq max_i f_i$. With $1/\ell \leq min_i f'_i$, $1/\ell \geq max_i f_i$ implies $min_i f'_i \geq max_i f_i$. The last inequality is much harder to satisfy than $f'_i \geq f_i$ for all x_i, the eligibility condition for F'-privacy. This is especially true when the distribution x_i is skewed where $max_i f_i$ is large, or when the sensitivity of x_i is skewed where $min_i f'_i$ is small.

3.4. Generalized bucketization problems

Given a F'-privacy specification and T, we want to find a bucketization T^* such that $Pr(x_i|t, T^*) \leq f'_i$ for every target individual t and every *SA* value x_i, and $Loss(T^*)$ is as small as possible. The key step is to determine the smallest size for each bucket so that all records can be assigned to some bucket and f'_i-privacy is satisfied in each bucket. In the rest of the paper, the term *bucket setting* refers to the specification of the size for each bucket and is written $\langle B_1, \ldots, B_q \rangle$ or $\cup B_j$, where B_j is a pair (S_j, b_j) and specifies b_j buckets of the size S_j, $0 < S_1 < \cdots < S_q$ and $b_j > 0$, $j = 1, \ldots, q$. $s(B_j) = b_j S_j$ denotes the capacity of the buckets specified by B_j, $Loss(\cup B_j)$ denotes the loss of the bucket setting, i.e., $\sum_{j=1}^{q} b_j \times (S_j - 1)^2$, as defined in Eq. (2). We want to find a bucket setting $\langle B_1, \ldots, B_q \rangle$ such that all of the following conditions are satisfied:

1. $\sum_j s(B_j) = |T|$, i.e., the buckets could hold all records in T,
2. there is an assignment of records to the buckets such that, for each x_i and each bucket g, the frequency of x_i in g is no more than f'_i, and
3. $Loss(\cup B_j)$ is minimized.

We say that a bucket setting satisfying 1) is *feasible*; a bucket setting satisfying 1) and 2) is *valid*, and in this case the record assignment in 2) is valid; a bucket setting satisfying all of 1), 2) and 3) is *optimal*. Here are the main problems we will study:

Definition 4. *The bucket setting validation problem*: Given a feasible bucket setting $\cup B_j$, test if it is valid w.r.t. $|T|$ and F'-privacy. *The bucket assignment problem*: Given a valid bucket setting $\cup B_j$, find a valid assignment of records in T to the buckets in $\cup B_j$. *The bucket setting problem*: Given T and F'-privacy, find an optimal bucket setting $\langle B_1, \ldots, B_q \rangle$.

The *one-size bucket setting problem* is the special case where $q = 1$, and the *two-size bucket setting problem* is the special case where $q = 2$. Even for the two-size bucket setting problem, finding an optimal solution is challenging because the number of feasible bucket settings is huge and considering all is prohibitive. For example, suppose that the sizes S_1 and S_2 are chosen from the range of [3, 20], and $|T| = 1,000,000$, there are a total of 2,077,869 feasible bucket settings of the form $\langle (S_1, b_1), (S_2, b_2) \rangle$. This number is much larger if $q > 2$.

4. Validating two-size bucket setting

This section considers the problem of testing if a given two-size bucket setting B is valid wrt T and F', that is, if there is an assignment of records in T to the buckets specified such that F'-privacy is satisfied. Let the function $Valid(B, T, F')$ return "true" if and only if B is valid wrt T and F'. In Section 4.1, we consider the case for one-size bucket setting B, and in Section 4.2, we consider the case for two-size bucket setting B. Section 4.3 presents an algorithm for assigning records to buckets for a valid two-size bucket setting B.

Table 4
RRA: each row represents a bucket and each integer i represents one record for x_i.

1	3	4	5	6	7	8	8
1	3	4	5	6	7	8	8
2	4	5	6	7	7	8	8
2	4	5	6	7	7	8	8

4.1. One-size validation

Consider b buckets $B = \{g_0, \ldots, g_{b-1}\}$ that have the *same* bucket size S. The following even distribution, called *Round-Robin Assignment (RRA)*, of records for x_i to all buckets $g_0, \ldots, g_{b-1}$ will minimize the frequency of x_i in a bucket: for each value x_i, $1 \le i \le m$, assign all the records for x_i to the buckets $g_0, \ldots, g_{b-1}$ in a round-robin manner. Essentially, RRA distributes the records for each x_i evenly across the buckets in order to minimize the frequency of x_i in a bucket. Therefore, to test if B is valid, it suffices to consider RRA because if this assignment cannot satisfy F'-privacy, no assignment can.

Table 4 shows an example of RRA that assigns 32 records to 4 buckets of size 8. Each row represents one bucket and an integer i represents one record for x_i. RRA evenly distributes the records for each x_i across buckets. It is easy to see that for b buckets of the size S each, the number of records for x_i assigned to a bucket is either $\lfloor |o_i|/b \rfloor$ or $\lceil |o_i|/b \rceil$, and the frequency of x_i in a bucket of size S is at most $\frac{\lceil o_i/b \rceil}{S}$. This observation gives rise to a sufficient and necessary condition for $Valid(B, T, F') = true$ (i.e., conditions 3) and 4)) in the next lemma.

Lemma 2 (One-size validation). *Let $B = \{g_0, \ldots, g_{b-1}\}$ be a set of buckets of size S such that $|T| = s(B)$ (note $s(B) = bS$). The following conditions are equivalent: (1) $Valid(B, T, F') = true$. (2) There is a valid RRA from T to B. (3) For each x_i, $\frac{\lceil o_i/b \rceil}{S} \le f'_i$. 4) For each x_i, $o_i \le \lfloor f'_i S \rfloor b$.*

Proof. We show 4) $\Rightarrow$ 3) $\Rightarrow$ 2) $\Rightarrow$ 1) $\Rightarrow$ 4). Let r be a real and i be an integer. $r \le i$ if and only if $\lceil r \rceil \le i$, and $i \le r$ if and only if $i \le \lfloor r \rfloor$. Therefore, $\frac{\lceil o_i/b \rceil}{S} \le f'_i \Leftrightarrow \lceil o_i/b \rceil \le f'_i S \Leftrightarrow \lceil o_i/b \rceil \le \lfloor f'_i S \rfloor \Leftrightarrow o_i/b \le \lfloor f'_i S \rfloor \Leftrightarrow o_i \le \lfloor f'_i S \rfloor b$. This implies 4) $\Rightarrow$ 3). To see 3) $\Rightarrow$ 2), observe that $\frac{\lceil o_i/b \rceil}{S}$ is the maximum frequency of x_i in a bucket generated by RRA. So 3) implies that this assignment is valid. 2) $\Rightarrow$ 1) follows because every valid RRA is a valid assignment. To see 1) $\Rightarrow$ 4), observe that F'-privacy implies that the number of occurrence of x_i in a bucket of size S is at most $\lfloor f'_i S \rfloor$. Thus for any valid assignment, the total number of occurrence o_i in the b buckets of size S is no more than $\lfloor f'_i S \rfloor b$. □

Lemmas 2 – 4 gives a test for $Valid(B, T, F') = true$, which can be evaluated efficiently given T, F', b, and S. In addition, if this condition holds, RRA gives a valid record assignment.

4.2. Two-size validation

For two-size bucket setting $B_1 \cup B_2$, where B_1 denotes a set of buckets of size S_1 and B_2 denotes a set of buckets of size S_2, $Valid(B_1 \cup B_2, T, F') = true$ if and only if there is a partition of T, $\{T_1, T_2\}$, such that $Valid(B_j, T_j, F') = true$, where T_j is the set of records assigned to the buckets in B_j, $j = 1, 2$. We present the algorithm for computing $Valid(B_1 \cup B_2, T, F')$. First, we define some notations.

Definition 5 (u_{ij} and a_{ij}). For each x_i and for $j = 1, 2$, we define $u_{ij} = \lfloor f'_i S_j \rfloor b_j$ and $a_{ij} = min\{u_{ij}, o_i\}$.

From Lemmas 2–4, if $Valid(B_j, T_j, F') = true$, u_{ij} is an upper bound on the number of records for x_i that can be allocated to the buckets in B_j. a_{ij} is an upper bound after considering the *maximum supply* of the x_i records, i.e., o_i. The next theorem gives an efficient algorithm for computing $Valid(B_1 \cup B_2, T, F')$.

Theorem 1 (Two-size validation). *$Valid(B_1 \cup B_2, T, F') = true$ if and only if all of the following hold:*

$$Privacy\ Constraint\ (PC) : \forall i : a_{i1} + a_{i2} \ge o_i \tag{3}$$

$$Fill\ Constraint\ (FC) : j = 1, 2, \textstyle\sum_i a_{ij} \ge b_j S_j \tag{4}$$

$$Capacity\ Constraint\ (CC) : |T| = b_1 S_1 + b_2 S_2 \tag{5}$$

Proof. Eq. (3) says that the total number of occurrences of x_i should not exceed the upper bound $a_{i1} + a_{i2}$, where a_{ij} is an upper bound imposed by F'-privacy for T_j, $j = 1, 2$. Eq. (4) says that under the upper bound constraint of a_{ij} it should be possible to fill up all the buckets in B_j. Eq. (5) says that the total bucket capacity matches the data cardinality. Clearly, all these conditions are necessary for a valid record assignment to the buckets in $B_2 \cup B_2$. The sufficiency proof follows from an algorithm in the next section that actually finds a partition $\{T_1, T_2\}$ of T such that $Valid(B_j, T_j, F') = true$, $j = 1, 2$, assuming that the above conditions hold. □

Thus, given T, F'-privacy, b_j, and S_j, $j = 1, 2,$ we can test efficiently whether there is a valid record assignment from T to the b_j buckets of size S_j *without* enumerating any actual assignment.

4.3. Record partition

The next question is how to produce a partition $\{T_1, T_2\}$ of T such that $Valid(B_1, T_1, F') = true$ and $Valid(B_2, T_2, F') = true$, assuming that the conditions in Theorem 1 hold. The answer to this question serves two purposes: it provides the sufficiency proof for Theorem 1 and it provides an algorithm for producing a valid record assignment for a valid two-size bucket setting $B_1 \cup B_2$, i.e., by first finding the partition $\{T_1, T_2\}$ and then assigning the records in T_j to the buckets in B_j by applying RRA to (B_j, T_j), $j = 1, 2$. The answer to the above question is given by Algorithm 1. Recall that o_i is the number of records for x_i in T and that a_{ij} and u_{ij}, $j = 1, 2,$ are defined in Definition 5. This algorithm produces $\{T_1, T_2\}$ such that for $j = 1, 2$, $o_{ij} \le u_{ij}$ for all x_i and $\sum_i o_{ij} = b_j S_j$, where o_{ij} denotes the number of records for x_i in T_j. This condition implies that the condition in Lemmas 2–4 holds on (B_j, T_j) for $j = 1, 2,$ therefore, $Valid(B_1, T_1, F') = true$ and $Valid(B_2, T_2, F') = true$.

Algorithm 1 Record Partition.

Input: T, B_1, B_2, F' satisfying all the conditions (i.e., PC, FC, and CC) in Theorem 1
Output: the partition $\{T_1, T_2\}$ of T such that, for $j = 1, 2$, $o_{ij} \le u_{ij}$ for all x_i and $|T_j| = b_j S_j$, where o_{ij} denotes the number of records for x_i in T_j and $|T_j| = \sum_i o_{ij}$

1: **for all** *SA* values x_i **do**
2: let T_1 contain any a_{i1} records for x_i in T and let T_2 contain the remaining $o_i - a_{i1}$ records for x_i in T
3: **end for**
4: **while** $|T_1| > b_1 S_1$ **do**
5: move one record for x_i from T_1, such that $o_{i2} < u_{i2}$, to T_2
6: **end while**
7: return $\{T_1, T_2\}$

Let us see why the output satisfies the specified condition. Initially, for each x_i, T_1 contains any a_{i1} records for x_i in T and T_2 contains the remaining $o_i - a_{i1}$ records for x_i. So $o_{i1} = a_{i1}$ and $o_{i2} = o_i - a_{i1}$. $o_{i1} = a_{i1}$ and $a_{i1} \le u_{i1}$ imply that Lemmas 2–4 holds on (B_1, T_1). From PC, $o_i - a_{i1} \le a_{i2}$, thus, $o_{i2} \le u_{i2}$, so Lemmas 2–4 holds on (B_2, T_2). It remains to see $|T_j| = b_j S_j$, $j = 1, 2$. Note $|T_1| = \sum_i a_{i1}$. FC implies $|T_1| \ge b_1 S_1$. If $|T_1| = b_1 S_1$, then $|T_2| = b_2 S_2$ because of CC, so $\{T_1, T_2\}$ is a partition specified in the output.

Let us assume $|T_1| > b_1 S_1$, thus $|T_2| < b_2 S_2$. To satisfy $|T_j| = b_j S_j$, $j = 1, 2$, Lines 3 and 4 move records from T_1 to T_2 while preserving $o_{i1} \le u_{i1}$ and $o_{i2} \le u_{i2}$. Clearly, moving records out of T_1 preserves $o_{i1} \le u_{i1}$. We claim that, as long as $|T_2| < b_2 S_2$, there must exist some x_i such that $o_{i2} < u_{i2}$, therefore, moving one record for x_i from T_1 into T_2 preserves $o_{i2} \le u_{i2}$. Suppose no such x_i exists, we must have $u_{i2} = o_{i2}$ for all x_i, thus, $\Sigma_i u_{i2} = |T_2|$. Then $u_{i2} \ge a_{i2}$ and FC imply $|T_2| \ge b_2 S_2$, which contradicts $|T_2| < b_2 S_2$. This shows the above claim. From this claim, as long as $|T_1| > b_1 S_1$ (thus, $|T_2| < b_2 S_2$), we can always move a record for some x_i from T_1 to T_2 while preserving $o_{i2} \le u_{i2}$, until $|T_2| = b_2 S_2$. This move is performed by the while-loop in Algorithm 1.

Corollary 1. *$Valid(B_1 \cup B_2, T, F') = true$ can be tested in $O(m + |T| log |T|)$ time.*

5. Optimal solution to two-size bucket setting

We now present an optimal solution for the two-size bucket setting problem: given T and F'-privacy, find the valid bucket setting of the form $\langle (S_1, b_1), (S_2, b_2) \rangle$, where $b_j \ge 0$ and $S_1 < S_2$, such that

$$Loss(B_1 \cup B_2) = b_1 (S_1 - 1)^2 + b_2 (S_2 - 1)^2 \tag{6}$$

is minimized. f'_i-privacy implies that a record for x_i must be placed in a bucket of size at least $\lceil 1/f'_i \rceil$; therefore, the minimum value for S_1 and S_2 is $M = min_i \{\lceil 1/f'_i \rceil\}$. The maximum value for S_1 and S_2 is constrained by the information loss allowed. We assume that this maximum value M' is given and we search for the optimal bucket setting over all the size pairs (S_1, S_2) in the range $M \le S_1 < S_2 \le M'$. Note that there may not be a valid bucket setting in this range, but if it does, we want to find an optimal bucket setting in this range.

As noted in Section 3.4, it is prohibitive to enumerate all feasible bucket settings. Our approach is pruning unpromising bucket settings without examining them. A bucket setting is unpromising if either it is not valid or it does not have the minimum loss. First, we need to organize all feasible bucket settings for each (S_1, S_2) according to *Loss*. We say that a candidate (b_1, b_2) is a *feasible* wrt (S_1, S_2) if $\langle (S_1, b_1), (S_2, b_2) \rangle$ is feasible, i.e., $|T| = S_1 b_1 + S_2 b_2$; (b_1, b_2) is a *valid* wrt (S_1, S_2) if $\langle (S_1, b_1), (S_2, b_2) \rangle$ is valid; (b_1, b_2) is an *optimal* wrt (S_1, S_2) if it is valid and if $Loss(B_1 \cup B_2)$ is minimum among all valid (b_1, b_2) wrt (S_1, S_2), where $B_j = (S_j, b_j)$, $j = 1, 2$.

5.1. Ordering candidates

Recall that $Loss(B_1 \cup B_2) = b_1(S_1 - 1)^2 + b_2(S_2 - 1)^2$. Given a size pair (S_1, S_2), where $S_1 < S_2$, since $S_1b_1 + S_2b_2 = |T|$, (b_1, b_2) has a smaller loss than (b'_1, b'_2) if and only if $b_1 > b'_1$ (in which case $b_2 < b'_2$). Therefore, if all feasible candidates (b_1, b_2) wrt (S_1, S_2) are listed in the descending order of b_1, they are in the ascending order of *Los*. Let $\Gamma(S_1, S_2)$ denote this list for (S_1, S_2).

Lemma 3. *If (b_1, b_2) precedes (b'_1, b'_2) in $\Gamma(S_1, S_2)$, $Loss(B_1 \cup B_2) < Loss(B'_1 \cup B'_2)$, where B_j contains b_j buckets of size S_j, and B'_j contains b'_j buckets of size S_j, $j = 1, 2$. Therefore, the first valid candidate (b_1, b_2) in $\Gamma(S_1, S_2)$ is optimal wrt (S_1, S_2).*

The beauty of $\Gamma(S_1, S_2)$ is that its ith candidate can be generated based on the index i, which means that there is no need to store $\Gamma(S_1, S_2)$. The first candidate (b_1, b_2) has the largest possible b_1 such that $S_1b_1 + S_2b_2 = |T|$. This (b_1, b_2) is the solution to the integer linear program:

$$min\{b_2 \mid S_1b_1 + S_2b_2 = |T|\}, \tag{7}$$

where b_1 and b_2 are variables of non-negative integers and S_1, S_2, $|T|$ are constants. Let (b_1^0, b_2^0) denote this first candidate in $\Gamma(S_1, S_2)$.

To determine the ith candidate, consider two consecutive candidates (b_1, b_2) and $(b_1 - \Delta_1, b_2 + \Delta_2)$ in $\Gamma(S_1, S_2)$. Note that $S_1b_1 + S_2b_2 = |T|$ and $S_1(b_1 - \Delta_1) + S_2(b_2 + \Delta_2) = |T|$, and Δ_1 and Δ_2 are the smallest positive integers such that these equalities hold. Therefore, $S_1\Delta_1 = S_2\Delta_2$, and $S_2\Delta_2$ are the least common multiple of S_1 and S_2 (i.e., the smallest positive integer that is a multiple of both S_1 and S_2). Let $lcm(S_1, S_2)$ denote the least common multiple of S_1 and S_2 . Then Δ_2 and Δ_1 are given by

$$\begin{aligned} \Delta_1 &= lcm(S_1, S_2)/S_1 \\ \Delta_2 &= lcm(S_1, S_2)/S_2 \end{aligned} \tag{8}$$

The ith candidate in $\Gamma(S_1, S_2)$, where $i \geq 0$, has the form $(b_1^0 - i * \Delta_1, b_2^0 + i * \Delta_2)$. The last candidate has the smallest possible $b_1^0 - i * \Delta_1$, that is, $0 \leq b_1^0 - i * \Delta_1 < \Delta_1$. The integer i satisfying this condition is given by

$$k = \lfloor b_1^0/\Delta_1 \rfloor \tag{9}$$

Lemma 4. *Let $b_1^0, b_2^0, \Delta_1, \Delta_2, k$ be defined in Eqs. (7)–(9). For $0 \leq i \leq k$, the ith candidate in $\Gamma(S_1, S_2)$ has the form*

$$(b_1^0 - i * \Delta_1, b_2^0 + i * \Delta_2) \tag{10}$$

Example 3. Let $|T| = 28, S_1 = 2, S_2 = 4$. $lcm(S_1, S_2) = 4$. $\Delta_2 = 4/4 = 1$ and $\Delta_1 = 4/2 = 2$. $b_1^0 = 14, b_2^0 = 0$. $k = \lfloor 14/2 \rfloor = 7$. $\Gamma(S_1, S_2)$ is (14,0), (12,1), (10,2), (8,3), (6,4), (4,5), (2,6), (0,7).

We now return to the problem of finding the optimal bucket setting $\langle (S_1, b_1), (S_2, b_2) \rangle$, where $M \leq S_1 < S_2 \leq M'$. We propose two pruning strategies to prune unpromising candidates in $\Gamma(S_1, S_2)$. The first strategy prunes candidates that do not have a minimum loss and the second strategy prunes the candidates that are not valid.

5.2. Loss-based pruning

We consider all size pairs (S_1, S_2), where $M \leq S_1 < S_2 \leq M'$, in some order. Let $Best_{loss}$ be the minimum loss of the valid bucket settings examined so far and let (S_1, S_2) be the next size pair to consider. Lemma 3 implies that all the candidates in $\Gamma(S_1, S_2)$ that have *Loss* less than $Best_{loss}$ are contained in a prefix of $\Gamma(S_1, S_2)$. Let (b_1^*, b_2^*) be the last candidate in this prefix. We have $b_1^* = b_1^0 - k^* * \Delta_1$ and $b_2^* = b_2^0 + k^* * \Delta_2$, where k^* is the maximum integer satisfying $b_1^*(S_1 - 1)^2 + b_2^*(S_2 - 1)^2 < Best_{loss}$. Solving this inequality gives

$$k^* = max\left\{0, \left\lfloor \frac{Best_{loss} - b_1^0(S_1 - 1)^2 - b_2^0(S_2 - 1)^2}{\Delta_2(S_2 - 1)^2 - \Delta_1(S_1 - 1)^2} \right\rfloor\right\} \tag{11}$$

Define

$$k' = min\{k, k^*\} \tag{12}$$

where k is given by Eq. (9).

From Lemma 3, any bucket setting in $\Gamma(S_1, S_2)$ that has a smaller loss than $Best_{loss}$ occurs in the first $k' + 1$ candidates in $\Gamma(S_1, S_2)$, and the first valid bucket setting in the first $k' + 1$ candidates is optimal wrt (S_1, S_2). In other words, it suffices to consider the first $k' + 1$ candidates in $\Gamma(S_1, S_2)$, instead of the entire $\Gamma(S_1, S_2)$. This loss-based pruning is stated in the next lemma and the algorithm based on this strategy is given in Algorithm 2.

Lemma 5. *Let $\Gamma'(S_1, S_2)$ contain the first $k' + 1$ candidates in $\Gamma(S_1, S_2)$. The first valid candidate in $\Gamma'(S_1, S_2)$ is optimal wrt (S_1, S_2).*

The input to Algorithm 2 consists of a table T, privacy parameter F', and the minimum and maximum bucket sizes M and M'. The output is an optimal two-size bucket setting wrt F' if one exists. Lines 1 and 2 initialize $Best_{loss}$ and $Best_{setting}$.

Algorithm 2 TwoSizeBucketing.

Input: T, F', M, M'
Output: optimal bucket setting $\langle (S_1, b_1), (S_2, b_2) \rangle$

```
1:  Best_loss ← ∞
2:  Best_setting ← NULL
3:  for all {S_1 = M; S_1 ≤ M' − 1; S_1 ++} do
4:    for all {S_2 = S_1 + 1; S_2 ≤ M'; S_2 ++} do
5:      compute k' using Eq. (12)
6:      let Γ'(S_1, S_2) be the first k' + 1 candidates in Γ(S_1, S_2)
7:      find the first valid candidate (b_1, b_2) in Γ'(S_1, S_2)
8:      if (b_1, b_2) is found then
9:        let B_j be the set of b_j buckets of size S_j, j = 1, 2
10:       if Best_loss > Loss(B_1 ∪ B_2) then
11:         Best_setting ← ⟨B_1, B_2⟩
12:         Best_loss ← Loss(B_1 ∪ B_2)
13:       end if
14:     end if
15:   end for
16: end for
17: return Best_setting
```

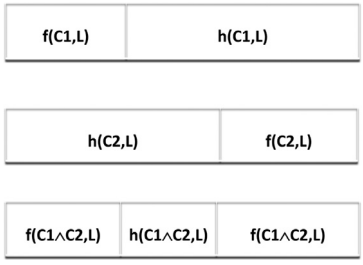


Fig. 1. (a) Monotone C_1. (b) Anti-monotone C_2. (c) $C_1 \wedge C_2$.

Lines 3 and 4 iterate all size pairs (S_1, S_2) with $M \le S_1 < S_2 \le M'$. For each (S_1, S_2), Lines 7–12 find the first valid candidate in $\Gamma'(S_1, S_2)$ and update $Best_{loss}$ and $Best_{setting}$ if necessary. Line 13 returns $Best_{setting}$. The loss-based pruning is implemented by focusing on the prefix $\Gamma'(S_1, S_2)$ derived from $Best_{loss}$. Though we did not specify the order of considering the sizes (S_1, S_2), to tighten up the above pruning, Lines 3 and 4 should examine smaller sizes (S_1, S_2) before examining larger ones, to ensure that small $Best_{loss}$ is generated as early as possible.

Algorithm 2 still needs to scan the prefix $\Gamma'(S_1, S_2)$ to find the first valid candidate. If there are many invalid candidates before the first valid candidate, this sequential scan is still expensive. In the next section, we present a binary search to locate the first valid candidate in $\Gamma'(S_1, S_2)$ by exploiting certain "monotonicity properties" of valid candidates.

5.3. Privacy-based pruning

Consider a sublist L of Γ' and a Boolean condition C defined on an candidate in L. Let $h(C, L)$ denote the set of all candidates in L on which C holds, and let $f(C, L)$ denote the set of all candidates in L on which C fails. We say that C is *monotone* in L if whenever a candidate in L is in $h(C, L)$, all later candidates in L are in $h(C, L)$, and we say that C is *anti-monotone* in L if whenever a candidate in L is in $f(C, L)$, all later candidates in L are in $f(C, L)$. A monotone C splits L into those in $f(C, L)$ on the left and those in $h(C, L)$ on the right, and we assume that $f(C, L)$ and $h(C, L)$ are sublists of L that preserve the order of candidates in L. An anti-monotone C splits L into those in $h(C, L)$ on the left and those in $f(C, L)$ on the right, and similarly, we assume that $f(C, L)$ and $h(C, L)$ are sublists of L.

Fig. 1 (a), (b) and (c) show $h(C_1, L)$, $h(C_2, L)$ and $h(C_1 \wedge C_2, L)$ for a monotone C_1 and anti-monotone C_2. An important point is that if C is monotone, we can find the first satisfying candidate in L by a binary search over L by testing C for at most $log_2|L|$ candidates, and if C is anti-monotone, we can find the first satisfying in L by testing the first candidate in L. More generally, if C_i, $1 \le i \le l$, are monotone in L and that C'_j, $1 \le j \le l'$, are anti-monotone in L, we can find the first candidate in L that satisfies $(\wedge_i C_i) \wedge (\wedge_j C'_j)$ by a binary search over L. The initial round starts with the full list L. At each round, we test if $\wedge_i C_i$ holds at the mid-point μ. If so, we drop the second half and search the first half. If not, there are two cases: if $\wedge_j C'_j$ holds at μ, we drop the first half and search the second half; if $\wedge_j C'_j$ fails at μ, there is no candidate in L satisfying $(\wedge_i C_i) \wedge (\wedge_j C'_j)$. At every mid-point μ, we test at most $l + l'$ conditions. In total, we test at most $(l + l')log_2|L|$ conditions.

Lemma 6. *Suppose that C_i, $1 \le i \le l$, are monotone in L and that C'_j, $1 \le j \le l'$, are anti-monotone in L, we can find the first candidate in L that satisfies $(\wedge_i C_i) \wedge (\wedge_j C'_j)$ by a binary search over L.*

In the discussion below, Γ' refers to $\Gamma'(S_1, S_2)$ when S_1 and S_2 are clear from the context. From Lemma 5, the first valid candidate in Γ' is optimal wrt (S_1, S_2), where a candidate in Γ' is valid if and only if it satisfies the conditions FC and PC in Theorem 1 (note that CC holds since all candidates in Γ' are feasible). We show that FC and PC can be broken into several monotone C_i and anti-monotone C'_j, therefore, we can find the first valid candidate in Γ' by a binary search over Γ'. In Theorem 1, let FC_{S_1} denote FC for $j = 1$ and FC_{S_2} denote FC for $j = 2$, and let PC_{x_i} denote PC for x_i. The set of valid candidates in Γ' is given by

$$\Gamma^* = h(FC_{S_1} \wedge FC_{S_2} \wedge (\wedge_i PC_{x_i}), \Gamma') \tag{13}$$

Below, we show that $FC_{S_1}, FC_{S_2}, PC_{x_i}$ are either monotone or anti-monotone.

5.3.1. Monotonicity of FC

Lemma 7. *FC_{S_1} is monotone in Γ' and FC_{S_2} is anti-monotone in Γ'.*

Proof. Rewrite FC_{S_1} and FC_{S_2} into

$$\sum_i min_i\{\lfloor f'_i S_1 \rfloor b_1, o_i\} \ge S_1 b_1 \tag{14}$$

$$\sum_i min_i\{\lfloor f'_i S_2 \rfloor b_2, o_i\} \ge S_2 b_2 \tag{15}$$

Let (b_1, b_2) precede (b'_1, b'_2) in Γ'. $b_1 > b'_1$ and $b_2 < b'_2$. Since o_i is a constant, decreasing b_1 to b'_1 preserves the satisfaction of Eq. (14), so Eq. (14) is monotone on Γ'. Similarly, increasing b_2 to b'_2 preserves the failure of Eq. (15), so Eq. (15) is anti-monotone on Γ'. □

The first valid candidate in Γ' is contained in the sublist $h(FC_{S_1} \wedge FC_{S_2}, \Gamma')$. In the following discussion, L_{FC} denotes this sublist. The boundaries of L_{FC} can be found by a binary search over Γ' using FC_{S_1} and FC_{S_2} because these conditions are monotone or anti-monotone (Lemma 7).

5.3.2. Monotonicity of PC

The monotonicity of PC_{x_i} is a bit more complicated. Let us rewrite PC_{x_i} into

$$min\{\lfloor f'_i S_1 \rfloor b_1, o_i\} + min\{\lfloor f'_i S_2 \rfloor b_2, o_i\} \ge o_i \tag{16}$$

Since b_1 is decreasing and b_2 is increasing in L, $\lfloor f'_i S_1 \rfloor b_1 \ge o_i$ is anti-monotone and $\lfloor f'_i S_2 \rfloor b_2 \ge o_i$ is monotone in L. Therefore, Eq. (16) holds in three cases:

Case1: $\lfloor f'_i S_1 \rfloor b_1 \ge o_i$,
Case2: $\lfloor f'_i S_2 \rfloor b_2 \ge o_i$,
Case3: $(\lfloor f'_i S_1 \rfloor b_1 < o_i) \wedge (\lfloor f'_i S_2 \rfloor b_2 < o_i)$.

If both Case1 and Case2 fail, we have Case3. In this case, Eq. (16) degenerates into

$$\lfloor f'_i S_1 \rfloor b_1 + \lfloor f'_i S_2 \rfloor b_2 \ge o_i \tag{17}$$

Consider any two consecutive candidates (b_1, b_2) and $(b_1 - \Delta_1, b_2 + \Delta_2)$ in L. If

$$\lfloor f'_i S_2 \rfloor \Delta_2 \ge \lfloor f'_i S_1 \rfloor \Delta_1 \tag{18}$$

holds, Eq. (17) holding on (b_1, b_2) implies that it holds on $(b_1 - \Delta_1, b_2 + \Delta_2)$, therefore, Eq. (17) is monotone; if Eq. (18) fails, Eq. (17) failing on (b_1, b_2) implies that it fails on $(b_1 - \Delta_1, b_2 + \Delta_2)$, therefore, Eq. (17) is anti-monotone. Note that Eq. (18) does not depend on (b_1, b_2) and it has a fixed value for each (S_1, S_2). The next corollary summarizes the above observation.

Corollary 2. *L_{FC} is divided into three sublists L_1, L_2, L_3 wrt PC_{x_i} corresponding to the following cases:*

- *Case1: $\lfloor f'_i S_1 \rfloor b_1 \ge o_i$ holds.*
- *Case2: $\lfloor f'_i S_2 \rfloor b_2 \ge o_i$ holds.*
- *Case3: Both Case1 and Case2 fail. In this case, Eq. (16) degenerates into Eq. (17). If Eq. (18) holds, Eq. (17) is monotone, and if Eq. (18) fails, Eq. (17) is anti-monotone.*

Fig. 2 shows the relative relationships of the three sublists L_1, L_2, L_3 of L_{FC}. Any of the sublists can be empty and L_1 and L_2 may overlap, in which case L_3 is empty. The boundaries, Boundary 1 and Boundary 2, of these sublists can be found by a binary search over L_{FC}. At each testing point, we test the conditions defining the three cases. In particular, if the condition for Case1 holds at the testing point, Boundary 1 will be in the right half, otherwise, Boundary 1 is in the left half; if the condition for Case2 holds at the testing point, Boundary 2 will be in the left half, otherwise, Boundary 2 is in the right half.

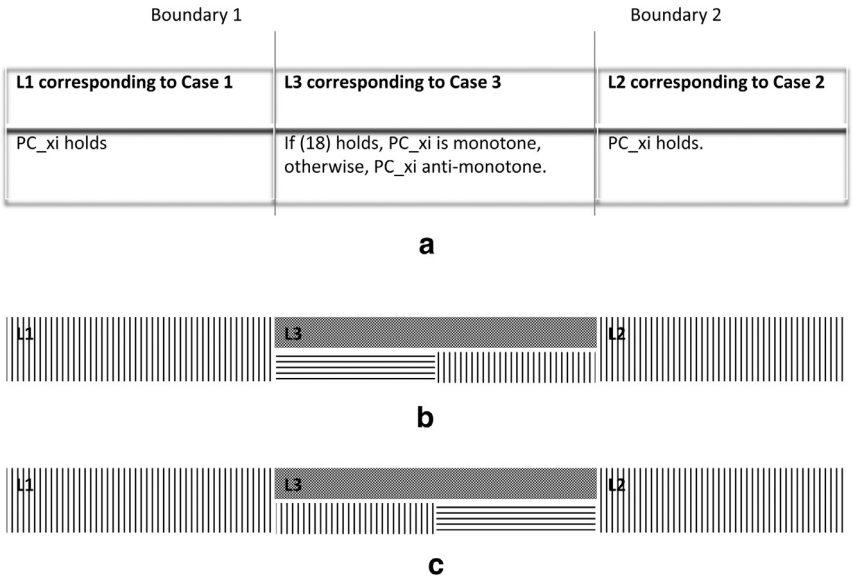


Fig. 2. (a) L_{FC} consists three sublists L_1, L_2, L_3, corresponding to Case1, Case2, and Case3. In L_1 and L_2, PC_{x_i} holds. In L_3, if Eq. (18) holds, PC_{x_i} is monotone, and if Eq. (18) fails, PC_{x_i} is anti-monotone. (b) Eq. (18) holds. (c) Eq. (18) fails. Vertically shaded areas refer to $h(PC_{x_i}, L_{FC})$ and horizontally shaded areas refer to $f(PC_{x_i}, L_{FC})$.

5.4. Algorithms

Now we combine all pruning strategies into the complete algorithm. From Lemma 5, the first valid candidate in Γ' is optimal wrt (S_1, S_2), and from Theorem 1, the set of valid candidates in Γ' is given by Γ^* in Eq. (13).

$$\Gamma^* = h(FC_{S_1} \wedge FC_{S_2} \wedge (\wedge_i PC_{x_i}), \Gamma') \tag{19}$$

Therefore, the first valid candidate in Γ' (if any) is the optimal candidate wrt (S_1, S_2). Algorithm 3 searches for the first valid candidate in Γ'. The algorithm is similar to Algorithm 2. The difference is that the search over Γ' is based on the sublist $L_{CF} = h(FC_{S_1} \wedge FC_{S_2}, \Gamma')$. As discussed above, L_{CF} is divided into three sublists L_1, L_2, L_3 such that PC_{x_i} holds in L_1 and L_2 and is either monotone or anti-monotone in L_3, depend on Eq. (18). The boundaries of these sublists can be found by a binary search. The first candidate satisfying PC_{x_i} in L_{FC} is the left most candidate in $\{c_1, c_3, c_2\}$, where c_1 and c_2 are the first candidate in L_1 and L_2, and c_3 is the first candidate satisfying PC_{x_i} in L_3. c_3 can be found by a binary search over L_3 because PC_{x_i} is either monotone or anti-monotone in L_3. Note that any of c_1, c_2, c_3 can be missing. Let C_i denote the left most candidate in $\{c_1, c_3, c_2\}$ for x_i. If C_i is found for all $1 \le i \le m$, the left most candidate in $\{C_1, \ldots, C_m\}$ is the first candidate in Γ^* in Eq. (13), thus, is the optimal candidate wrt (S_1, S_2), otherwise, there is no optimal candidate wrt (S_1, S_2).

Since Γ' is at most $|T|$ in length, the binary search takes at most time $O(m\log|T|)$.

Theorem 2. *Given o_i for all x_i, $1 \le i \le m$, finding the first candidate in Γ^* at Line 6 in Algorithm 3 takes time $O(m\log|T|)$.*

6. General bucket setting

In this section, we consider the general bucket setting problem where there is no restriction on the number of different bucket sizes.

6.1. ILP based solution

The first solution is optimal based on integer linear programming and works only for a small domain size m of SA. Let M and M' denote the minimum and maximum bucket sizes. Let $n = M' - M + 1$. Define $S_j = M + j - 1$, where $j = 1, \ldots, n$. Notice $S_1 = M$, $S_n = M'$, and $S_{j+1} = S_j + 1$ for $1 \le j \le n-1$. For $j = 1, \ldots, n$, let b_j be the variables representing the number of S_j-sized buckets, let B_j be the set of such buckets, with $s(B_j) = b_j S_j$. Let $o_1, \ldots, o_m$ be the numbers of occurrences of $x_1, \ldots, x_m$, respectively. For $1 \le i \le m$ and $1 \le j \le n$, let v_{ij} denote the variables for the number of occurrences of x_i in the buckets in B_j. Let $u_{ij} = \lfloor f_i' S_j \rfloor b_j$, as defined in Definition 5. Our objective is to minimize $Loss$, i.e., min $\sum_{j=1}^{n} b_j (S_j - 1)^2$, under the following constraints: (1) $\forall i : \sum_{j=1}^{n} v_{ij} = o_i$, (2) $\forall j : \sum_{i=1}^{m} v_{ij} = S_j b_j$, (3) $\forall i, j : v_{ij} \le u_{ij}$, (4) $\sum_{j=1}^{n} S_j b_j = |T|$. v_{ij} and b_j are variables of non-negative integers and S_j, o_i, u_{ij} are constants. The first constraint says that the total occurrences of a value x_i in all buckets is equal to o_i. The second constraint says that the total occurrences of all values x_i assigned to a bucket is equal to the capacity of that bucket. The third one ensures the upper bound imposed by F'-privacy as in Definition 5. From

Algorithm 3 TwoSizeBucketing.

Input: T, F', M, M'
Output: optimal bucket setting $\langle (S_1, b_1), (S_2, b_2) \rangle$

```
1: Best_loss ← ∞
2: Best_setting ← NULL
3: for all {S_1 = M; S_1 ≤ M' − 1; S_1 ++} do
4:   for all {S_2 = S_1 + 1; S_2 ≤ M'; S_2 ++} do
5:     let Γ* be defined in Eq. (13)
6:     let (b_1, b_2) be the first candidate in Γ* (computed by binary search over Γ' described)
7:     if (b_1, b_2) is found then
8:       let B_j be the set of b_j buckets of size S_j, j = 1, 2
9:       if Best_loss > Loss(B_1 ∪ B_2) then
10:        Best_setting ← ⟨B_1, B_2⟩
11:        Best_loss ← Loss(B_1 ∪ B_2)
12:        update Γ' by Best_loss using Eq. (12)
13:      end if
14:    end if
15:  end for
16: end for
17: return Best_setting
```

Lemma 2, this condition ensures F'-privacy. The last constraint says that the cardinality of T is equal to the total capability of all buckets.

6.2. Heuristic solution

The second solution is based on the optimal solution to the two-size bucket setting problem presented in Section 5.4. This solution, called *MultiSizeBucketing*, is given in Algorithm 4. The input consists of T, *Buckets*, F', M, M', where T is a set of records and *Buckets* is a set of buckets of the same size such that $s(Buckets) = |T|$. The algorithm calls *TwoSizeBucketing* to find the optimal two-size bucket setting $\langle B_1, B_2 \rangle$ for T (Line 1). If $Loss(B_1 \cup B_2) < Loss(Buckets)$, *RecordPartition*($T$, B_1, B_2) partitions the records in T into T_1 and T_2 for B_1 and B_2, as discussed in Section 4.3. Lines 4 and 5 recur on each of (T_1, B_1) and (T_2, B_2). If $Loss(B_1 \cup B_2) \geq Loss(Buckets)$, Line 7 returns the current (T, *Buckets*) without further recursion. The main call is *MultiSizeBucketing*(T, B, F', M, M'), where B is a single bucket of size $|T|$.

Algorithm 4 MultiSizeBucketing.

Input: T, *Buckets*, F', M, M'
Output: a bucket setting $\langle B_1, \ldots, B_q \rangle$ and $T_1, \ldots, T_q$, where T_j is a set of records for B_j, $1 \leq j \leq q$

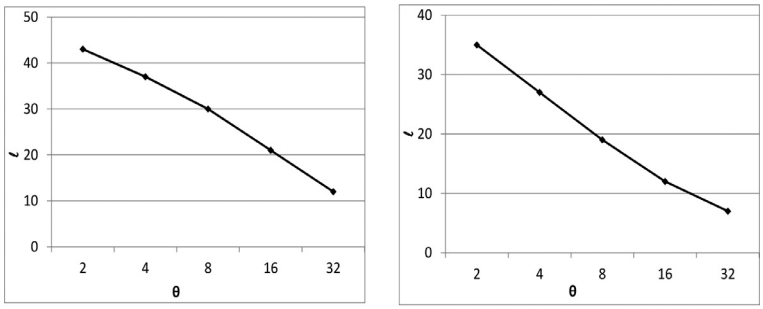
```
1: ⟨B_1, B_2⟩ ← TwoSizeBucketing(T, F', M, M')
2: if Loss(B_1 ∪ B_2) < Loss(Buckets) then
3:   (T_1, T_2) ← RecordPartition(T, B_1, B_2) (Section 4.3)
4:   MultiSizeBucketing(T_1, B_1, F', M, M')
5:   MultiSizeBucketing(T_2, B_2, F', M, M')
6: else
7:   return(T, Buckets)
8: end if
```

7. Empirical studies

This section evaluated the generalized bucketization solutions proposed in Sections 4 and 6. We used the real life CENSUS data set for 500K American adults, previously used in [13,16,22]. There are eight discrete attributes: Age (76), Gender (2), Education (14), Marital (6), Race (9), Work-Class (10), Country (83), and Occupation (50), where the integers in the brackets are the domain sizes. We derived two base tables from CENSUS. The first table denoted OCC has Occupation as *SA* and the remaining attributes as the *QI*-attributes. The second table denoted EDU has Education as *SA* and the remaining attributes as the *QI*-attributes. OCC-n and EDU-n denote the data sets of OCC and EDU of the cardinality n. OCC represents a more balanced distribution of *SA* values with the maximum $f_i = 7.5\%$, and EDU represents a more skewed distribution of *SA* values with the maximum $f_i = 27.3\%$.

Table 5
Parameter settings.

Parameters	Settings
Cardinality $\|T\|$	100k, 200k, **300k**, 400k, 500k
f_i'-privacy for x_i	$f_i' = min\{1, \theta \times f_i + 0.02\}$
θ	2, 4, **8**, 16, 32
Minimum bucket size M	$min_i\{\lceil 1/f_i' \rceil\}$
Maximum bucket size M'	50



(a) OCC (b) EDU

Fig. 3. ℓ (y-axis) vs θ (x-axis). ℓ is the parameter of the corresponding ℓ-diversity for F'-privacy specified by $f_i' = min\{1, \theta \times f_i + 0.02\}$. For most values of θ, a large ℓ, i.e., more than 10, is required for the corresponding ℓ-diversity. Such ℓ-diversities often cannot be enforced for the OCC and EDU data sets because $1/\ell$ is less than the maximum frequency of SA values in these data sets, which are 27.3% for EDU and 7.5% for OCC.

Table 5 summarizes the parameter settings in the subsequent experiments with the default settings in bold face. We consider F'-privacy specified by $f_i' = min\{1, \theta \times f_i + 0.02\}$ for each SA value x_i, $1 \le i \le m$, where f_i is the frequency of x_i in the data set and θ is the coefficient chosen from {2, 4, 8, 16, 32}. The small constant 0.02 is the base tolerance in inference probability for all SA values. On top of this, $\theta \times f_i$ models an additional tolerance specific to each SA value, that is, a less frequently occurring x_i value tend to be more sensitive and have a smaller privacy threshold. For example, suppose that HIV is less common than Flu, we expect that HIV has a smaller privacy threshold than Flu. θ specifies the rate of this relationship. The choices of θ from {2, 4, 8, 16, 32} represent a wide range of F'-privacy settings. For all of these choices, $f_i' \ge f_i$, so the eligibility condition for having a F'-Privacy solution (Lemma 1) is satisfied.

As discussed in Section 5, the minimum bucket size should be set to $M = min\{\lceil 1/f_i' \rceil\}$ because any bucket of a size smaller than this minimum size cannot satisfy the F'-privacy. We set the maximum bucket size to $M' = 50$.

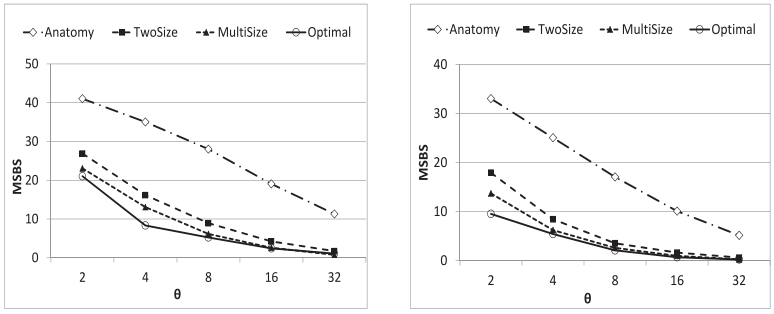
We consider three evaluation criteria: ability for handling skewed sensitivity and distribution of sensitive values, data utility, and scalability.

7.1. Criterion 1: handling skewed sensitivity and distribution of SA values

One claimed objective of the generalized bucketization scheme and F'-privacy is dealing with skewed sensitivity and skewed distribution of sensitive values. To evaluate how this objective is achieved, we consider how often F'-privacy has a satisfying solution but the corresponding ℓ-diversity that enforces the same F'-privacy has no solution, where $\ell = \lceil 1/min_i f_i' \rceil$ (see Remark 1). According to Lemma 1, F'-privacy has a satisfying solution T^* if and only if $f_i' \ge f_i$ holds for all x_i. This condition is satisfied for all θ settings considered.

For the corresponding ℓ-diversity, $\ell = \lceil 1/min_i f_i' \rceil$. From Remark 1, this ℓ-diversity has a satisfying solution if and only if $1/\ell \ge max_i f_i$. For the OCC-300K that has a more balanced f_i where $max_i f_i = 7.5\%$, $1/\ell \ge max_i f_i$ is violated for all $\ell \ge 14$. In particular, for all $\theta \in \{2, 4, 8, 16, 32\}$, Fig. 3 shows the value ℓ for the corresponding ℓ-diversity. Except for $\theta = 32$, we have $\ell \ge 14$. This means that the corresponding ℓ-diversity for most F'-privacy considered does not have a satisfying solution. This situation gets even worse for the EDU-300K that has $max_i f_i = 27.3\%$: the eligibility condition $1/\ell \ge max_i f_i$ is violated for all $\ell \ge 4$, and from Fig. 3, this means that for all the F'-privacy considered, the corresponding ℓ-diversity does not have a satisfying solution.

The eligibility condition of the corresponding ℓ-diversity implies $min_i f_i' \ge max_i f_i$ (see Remark 1), which can be violated for two reasons. The first reason is that $max_i f_i$ is large, and this occurs for a skewed distribution of x_i values such as EDU-300K. The second reason is that $min_i f_i'$ is small. Importantly, these reasons are inherent to the uniform privacy setting of ℓ-diversity, independent of the methods for achieving ℓ-diversity. Therefore, all methods for achieving ℓ-diversity cannot escape the above drawback. In contrast, the eligibility condition for F'-privacy compares f_i' with f_i, i.e., $f_i' \ge f_i$. This condition is much easier to satisfy than $min_i f_i' \ge max_i f_i$, especially when x_i has a skewed distribution, which has a large $max_i f_i$, or a skewed sensitivity, which has a small $min_i f_i'$.



(a) OCC (b) EDU

Fig. 4. *MSBS* (y-axis) vs θ (x-axis). "Anatomy" has a much larger *MSBS* than "TwoSize", "MultiSize", and "Optimal". The closeness of the last three suggests that "TwoSize" and "MultiSize" are good approximation of the optimal solution in the general setting.

7.2. Criterion 2: data utility

We created a pool of 5000 count queries defined in Section 3.2 following the same procedure described in [22]. For each query Q in the pool, we compute the estimated answer *est* using T^* as described in Section 3.2. The relative error (RE) of Q is defined by $RE = |act - est|/act$, where *act* is the actual answer. We report the average *RE* over all queries in the pool.

We also evaluate the *Mean Squared Bucket Size (MSBS)* of T^*, which was the search criterion used by our algorithm. We would like to see if there is a correlation between *MSBS* and *RE* because such correlation would confirm the effectiveness of *MSBS* as a search criterion. There is another reason for evaluating *MSBS*. As observed in Section 7.1, the corresponding ℓ-diversity that enforces a given F'-privacy often does not have a satisfying solution. In such cases, no data is published, so *RE* cannot be collected, but we can still compare *MSBS* because *MSBS* depends only on the bucket sizes, which are either ℓ or $\ell - 1$ for the basic bucketization scheme "Anatomy" proposed in [22].

We evaluate the following methods. **Optimal** is the optimal solution for the general bucket setting problem obtained from the integer linear programming in Section 6.1. This method provides the lower bound on *MSBS* but is exponential in the domain size $|SA|$. **TwoSize** is the optimal solution for the two-size bucket setting problem produced by Algorithm 3. **MultiSize** is the heuristic solution for the general bucket setting problem produced by the solution to the two-size bucket setting problem produced by Algorithm 4. **Anatomy** is the "Anatomy" algorithm in [22] run with the corresponding ℓ-diversity for a given F'-privacy.

7.2.1. MSBS

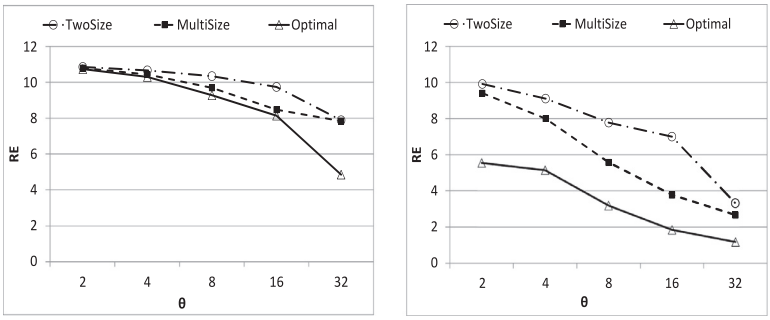
Fig. 4 shows *MSBS* vs various specification of F'-privacy in terms of the coefficient θ on the default OCC-300K and EDU-300K. "Anatomy" has a significantly higher *MSBS* than the other methods across all settings of θ. This is because the buckets produced by "Anatomy" have a size of either ℓ or $\ell + 1$, where $\ell = \lceil 1/min_i f'_i \rceil$. If any x_i has a small f'_i, ℓ is large, so is *MSBS*. "TwoSize" has a slightly higher *MSBS* than "MultiSize", which has a slightly higher *MSBS* than "Optimal". The closeness of *MSBS* for the last three methods suggests that both the solution to the two-size bucket settings and the heuristic solution are good approximations to the optimal solution for the general bucket setting problem.

7.2.2. Relative Error (RE)

The results on *RE* are summarized in Fig. 5. "Anatomy" is missing because we cannot evaluate *RE* without a satisfying solution for the corresponding ℓ-diversity, as discussed earlier. The maximum *RE* is slightly over 10%, but most time *RE* is below 10%. Similar to *MSBS*, *RE* for "TwoSize", "MultiSize", and "Optimal" are relatively close to each other: "MultiSize" loses no more than 2% accuracy compared to "Optimal", and "TwoSize" loses no more than 2% accuracy compared to "MultiSize", while this closeness is more observed on the balanced OCC data than on the skewed EDU data. The small *RE* in these algorithms suggested that *MSBS* as a search criterion for utility indeed helps preserve the utility for count queries.

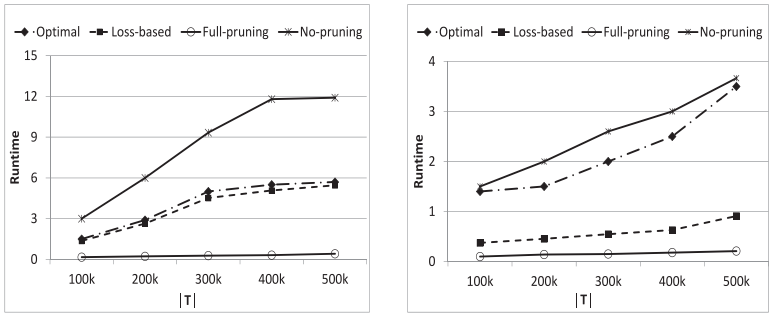
7.3. Criterion 3: scalability

We evaluated the effectiveness of several pruning settings for the two-size bucket setting problem in Section 5. **No-pruning** denotes the sequential search of the full list Γ without any pruning; **Loss-pruning** denotes the sequential search of the prefix Γ' presented in Algorithm 2, which exploits only loss-based pruning; **Full-pruning** denotes the binary search of Γ' presented in Algorithm 3, which exploits both loss-based pruning and privacy-based pruning. **Optimal** denotes the integer linear programming solution applied to the two-size bucket setting problem. The default setting $\theta = 8$ is used for specifying F'-privacy. All algorithms were implemented in C++ with ILP being implemented in Matlab and all experiments were run on a Windows 64 bits Platform with CPU of 2.53 GHz and memory size of 12GB. Each algorithm was run 100 times and the average time is reported here.



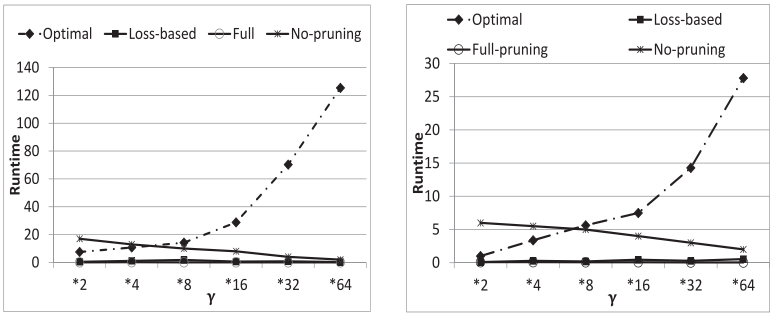
(a) OCC (b) EDU

Fig. 5. *RE* (%) (*y*-axis) vs θ (*x*-axis). The small differences among *RE*'s for "TwoSize", "MultiSize", and "Optimal" suggest "TwoSize" and "MultiSize" are good approximation of the optimal solution in the general setting.



(a) OCC (b) EDU

Fig. 6. Runtime in seconds (*y*-axis) vs |*T*| (*x*-axis). The pruning strategies effectively reduce runtime.



(a) OCC (b) EDU

Fig. 7. Runtime in seconds (*y*-axis) vs the scale factor γ of |*SA*| (*x*-axis). As the scale factor increases, Runtime of "Optimal" increases exponentially while those of other methods increase slowly.

7.3.1. Scalability with |T|

The runtime vs the data cardinality |*T*| is summarized in Fig. 6. "Loss-pruning" significantly reduces the time compared to "No-pruning", but has an increasing trend in runtime as |*T*| increases because of the sequential search of the list Γ'. "Full-pruning" takes the least time and remains constant as |*T*| increases because the length of Γ' has little effect on the binary search over Γ'. "Optimal" takes less time than "No-pruning" because the domain size m of *SA* is relatively small for both data sets. The next experiment shows that the comparison will be reversed as we scale up the domain size m.

7.3.2. Scalability with |SA|

We scale up the domain size $m = |SA|$ for OCC-500K and EDU-500K by a factor γ, where γ is ranged over 2, 4, 8, 16, 32 and 64. To create a new domain of *SA* of the size $m \times \gamma$, for each record t in T, we replace the value $t[SA]$ in t with another value $\gamma \times t[SA] + r$, where r is an integer selected randomly from the range $[0, \gamma - 1]$ with equal probability. With $t[SA]$ having the domain $\{0, 1, \ldots, m-1\}$, the new value $\gamma \times t[SA] + r$ has the domain $\{0, 1, \ldots, \gamma \times m - 1\}$.

Fig. 7 summarizes Runtime vs γ. As γ increases, the integer linear programming based "Optimal" takes significantly longer time due to the exponential growth of runtime in the domain size of *SA*. Runtime of the other algorithms increases little because their complexity is linear in the domain size of *SA*. In fact, as the domain size of *SA* increases, Runtime of "No-pruning" decreases since more *SA* values mean smaller f_i and f_i', thus, larger minimum bucket size $M = min_i\{\lceil 1/f_i' \rceil\}$. This leads to a shorter Γ list to search.

8. Conclusion

In this work, we presented a generalized bucketization scheme to limit sensitive non-independent reasoning, with two important differences from previous bucketization schemes: it allows a flexible privacy setting and it allows flexible bucket sizes. Our study shows that these flexibilities help publishing more data in the case of skewed privacy settings or a skewed distribution of *SA* values. Our bucketization scheme does not require a taxonomy on attribute values, thus, is applicable to more data sets.

Acknowledgments

The first author's work is partially supported by a Discovery Grant of the Natural Sciences and Engineering Research Council of Canada Grant No. 201406027.

References

[1] N.R. Adam, J.C. Wortmann, Security-control methods for statistical databases: A comparative study, ACM Comput. Surv. 21 (1989) 515–556.
[2] A. Blum, C. Dwork, F. McSherry, K. Nissim, Practical privacy: The SuLQ framework, in: Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), 2005, pp. 128–138.
[3] J. Cao, P. Karras, Publishing microdata with a robust privacy guarantee, Proc. VLDB Endow. 5 (11) (2012) 1388–1399.
[4] B. Chen, D. Kifer, K. LeFevre, A. Machanavajjhala, Privacy-preserving data publishing, Found. Trends Databases 2 (1-2) (2009) 1–167.
[5] C. Clifton, T. Tassa, On syntactic anonymity and differential privacy, in: Proceedings of the 2013 IEEE Twenty-ninth International Conference on Data Engineering Workshops (ICDEW), vol. 0, 2013, pp. 88–93.(Also, Transactions on data privacy, 6 (2013) 161–183).
[6] G. Cormode, Personal privacy vs population privacy: learning to attack anonymization, in: Proceedings of the Seventeenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2011, pp. 1253–1261.
[7] C. Dwork, Differential privacy, in: Proceedings of the 2006 International Colloquium on Automata, Languages and Programming (ICALP), 2006, pp. 1–12.
[8] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in: S. Halevi, T. Rabin (Eds.), Theory of Cryptography, Lecture Notes in Computer Science, vol. 3876, 2006.
[9] A. Evfimievski, J. Gehrke, R. Srikant, Limiting privacy breaches in privacy preserving data mining, in: Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), 2003, pp. 211–222.
[10] B. Fung, K. Wang, R. Chen, P. Yu, Privacy-preserving data publishing: A survey of recent developments, ACM Comput. Surv. 42 (4) (2010) 14:1–14:53.
[11] C. Han, K. Wang, Sensitive disclosures under differential privacy guarantees, in: Proceedings of the 2015 IEEE International Congress on Big Data (BigData Congress), 2015.
[12] D. Kifer, Attacks on privacy and deFinetti's theorem, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD), 2009, pp. 127–138.
[13] K. LeFevre, D.J. DeWitt, R. Ramakrishnan, Incognito: Efficient full-domain k-anonymity, in: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD), 2005, pp. 49–60.
[14] N. Li, T. Li, S. Venkatasubramanian, t-closeness: Privacy beyond k-anonymity and l-diversity, in: Proceedings of the Twenty-third International Conference on Data Engineering (ICDE), 2007, pp. 106–115.
[15] J. Liu, K. Wang, On optimal anonymization for l+-diversity, in: Proceedings of the IEEE Twenty-sixth International Conference on Data Engineering (ICDE), 2010, pp. 213–224.
[16] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, l-diversity: Privacy beyond k-anonymity, in: Proceedings of the Twenty-second International Conference on Data Engineering (ICDE), 2006.
[17] L. Sweeney, k-anonymity: A model for protecting privacy, Int. J. Uncertain. Fuzziness Knowl. Based Syst. 10 (5) (2002) 557–570.
[18] Y. Tao, X. Xiao, J. Li, D. Zhang, On anti-corruption privacy preserving publication, in: Proceedings of the 2008 IEEE Twenty-fourth International Conference on Data Engineering (ICDE), 2008, pp. 725–734.
[19] K. Wang, B.C.M. Fung, P.S. Yu, Handicapping attacker's confidence: An alternative to k-anonymization, Knowl. Inf. Syst. Int. J. 11 (3) (2007) 345–368.
[20] K. Wang, C. Han, A.W. Fu, R.C. Wing, P.S. Yu, Reconstruction privacy: Enabling statistical learning, in: Proceedings of the Eighteenth International Conference on Extending Database Technology (EDBT), 2015, pp. 469–480.
[21] R. Wong, A. Fu, K. Wang, P. Yu, J. Pei, Can the utility of anonymized data be used for privacy breaches? ACM Transactions on Knowledge Discovery from Data (TKDD) 5 (3) (2011).
[22] X. Xiao, Y. Tao, Anatomy: Simple and effective privacy preservation, in: Proceedings of the Thirty-second International Conference on Very Large Data Bases (VLDB), 2006, pp. 139–150.
[23] X. Xiao, Y. Tao, Personalized privacy preservation, in: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD), 2006, pp. 229–240.