The VLDB Journal (2011) 20:59–81
DOI 10.1007/s00778-010-0191-9

REGULAR PAPER

SABRE: a Sensitive Attribute Bucketization and REdistribution framework for *t*-closeness

Jianneng Cao · Panagiotis Karras · Panos Kalnis · Kian-Lee Tan

Received: 1 July 2009 / Revised: 7 April 2010 / Accepted: 19 April 2010 / Published online: 19 May 2010

Abstract Today, the publication of microdata poses a privacy threat: anonymous personal records can be re-identified using third data sources. Past research has tried to develop a concept of *privacy guarantee* that an *anonymized* data set should satisfy before publication, culminating in the notion of t-closeness. To satisfy t-closeness, the records in a data set need to be grouped into Equivalence Classes (ECs), such that each EC contains records of indistinguishable *quasi-identifier* values, and its local distribution of *sensitive attribute* (SA) values conforms to the global table distribution of SA values. However, despite this progress, previous research has not offered an anonymization algorithm tailored for t-closeness. In this paper, we cover this gap with SABRE, a *SA* *B*ucketization and *RE*distribution framework for t-closeness. SABRE first *greedily* partitions a table into buckets of similar SA values and then redistributes the tuples of each bucket into dynamically determined ECs. This approach is facilitated by a property of the Earth Mover's Distance (*EMD*) that we employ as a measure of distribution closeness: If the tuples in an EC are picked *proportionally* to the sizes of the buckets they hail from, then the *EMD* of that EC is tightly upper-bounded using localized upper bounds derived for each bucket. We prove that if the t-closeness constraint is properly obeyed during partitioning, then it is obeyed by the derived ECs too. We develop two instantiations of SABRE and extend it to a streaming environment. Our extensive experimental evaluation demonstrates that SABRE achieves information quality superior to schemes that merely applied algorithms tailored for other models to t-closeness, and can be much faster as well.

J. Cao (✉) · P. Karras · K.-L. Tan
School of Computing, National University of Singapore,
Singapore, Republic of Singapore
e-mail: caojianneng@comp.nus.edu.sg

P. Karras
e-mail: karras@comp.nus.edu.sg

K.-L. Tan
e-mail: tankl@comp.nus.edu.sg

P. Kalnis
Division of Mathematical and Computer Sciences and Engineering,
King Abdullah University of Science and Technology,
Thuwal, Saudi Arabia
e-mail: panos.kalnis@kaust.edu.sa

Keywords t-closeness · Earth Mover's Distance

1 Introduction

Organizations such as ministries or hospitals regularly release microdata (e.g., census data or medical records) for purposes that serve the common good through the advancement of knowledge. Still, the same data can inadvertently reveal sensitive personal information to malicious adversaries. Research and practice have shown that merely concealing explicit identifying attributes, such as *name, address* and *telephone number*, is not sufficient to protect personal privacy. An attacker may still uncover the hidden identities by joining the released microdata attributes with other publicly available data. The set of attributes instrumental in that purpose, such as *gender, zipcode*, and *age*, are called *quasi-identifier* (QI). The *anonymization* problem calls for bringing the data to a publishable form that forestalls such *linking attacks* while preserving as much of the original information as possible.

The question of what *form* the data should be brought to is a subject of inquiry in itself. Past research has tried to formulate the *privacy guarantee* that a properly *anonymized* data set should satisfy in order to be safely published. The main highlights of this effort consist of the k-anonymity

[26,27], ℓ-diversity [20], and, most recently, the t-closeness [18] model. In particular, the k-anonymity model postulates that microdata be partitioned into a set of *equivalence classes* (ECs) of *at least* k tuples each, and all tuples within an EC be assigned the same, *generalized* value over each of their QI attributes [26,27]. In effect, the k-anonymity guarantee protects against identity disclosure by hiding each released tuple in a crowd of at least $k-1$ other tuples, all indistinguishable with respect to their QI. Still, k-anonymity does not heed the values of a non-QI *sensitive attribute* (SA), hence the privacy of such values may be compromised. The ℓ-diversity model addresses this limitation, requiring that at least ℓ different SA values be "well represented" within each EC [20]. However, even ℓ-diversity fails to protect against attacks arising from an adversary's unavoidable knowledge of the *overall* distribution of SA values in a released table [18]. Thus, a *skewness* attack may occur when the SA distribution in an EC differs substantially from that in the published table as a whole (e.g., in case the positive occurrence of AIDS in an EC is 99%, while in the whole table it is 1%). Furthermore, a *similarity* attack is made possible when the SA values in an EC are semantically similar (e.g., Table 2 is a 3-diverse version of Table 1, yet all tuples in EC 1 indicate a *respiratory problem*).

The t-closeness model aims to forestall the type of attacks outlined above, by requiring that the SA distribution in any EC differs from its overall distribution by *at most* a given threshold t, according to an appropriate distance metric. The value of t constrains the *additional* information an adversary *gains* after seeing a single EC, measured with respect to the information provided by the full released table. The t-closeness guarantee directly protects against a *skewness* attack, while it also provides defense against a *similarity* attack, depending on the extent to which semantic similarity exists among the SA values in the whole table [18].

Table 1 Patient records

ID	Name	Weight	Age	Disease
01	Mike	60	40	SARS
02	Alice	70	50	intestinal cancer
03	John	60	60	pneumonia
04	Bob	50	50	bronchitis
05	Beth	80	50	gastric flu
06	Carol	70	70	gastric ulcer

Table 2 3-diverse published table

EC	Weight	Age	Disease
1	[50–60]	[40–60]	SARS
	[50–60]	[40–60]	Pneumonia
	[50–60]	[40–60]	Bronchitis
2	[70–80]	[50–70]	Intestinal cancer
	[70–80]	[50–70]	Gastric Flu
	[70–80]	[50–70]	Gastric ulcer

The t-closeness model poses the problem of bringing a microdata table to a form that complies with it while compromising data quality as little as possible. This problem is *distinct* from those posed by other privacy models. Each model poses a particular *tradeoff* between privacy and information quality, which needs to be resolved in an effective and efficient manner. However, the two extant schemes for t-closeness [18,19] are extensions of algorithms designed for k-anonymity; they employ either the *Incognito* [14] or the *Mondrian* [15] technique for k-anonymization, merely adding to them the extra condition that the produced ECs should satisfy t-closeness. Still, a good t-closeness anonymization does not[1] necessarily derive from a good k-anonymization. Thus, unfortunately, the techniques in [18,19] limit the scope of the solutions they can achieve by building them on top of k-anonymizations, and fail in terms of efficiency by performing too many brute-force t-closeness satisfaction checks. The question of an algorithm tailored for t-closeness-abiding anonymization remains open.

In this paper, we provide SABRE, a *S*ensitive *A*ttribute *B*ucketization and *RE*distribution framework for t-closeness. SABRE operates in two phases. First, it partitions a table into buckets of similar SA values in a greedy fashion. Then, it redistributes tuples from each bucket into dynamically configured ECs. Following [18,19], we employ the Earth Mover's Distance (*EMD*) as a measure of closeness between distributions, and utilize a property of this measure to facilitate our approach. Namely, a tight *upper bound* for the *EMD* of the distribution in an EC from the overall distribution can be derived as a function of localized upper bounds for each bucket, provided that the tuples in the EC are picked *proportionally* to the sizes of the buckets they hail from. Furthermore, we prove that if the bucket partitioning obeys t-closeness, then the derived ECs also abide to t-closeness. We develop two SABRE instantiations. The former, SABRE-AK focuses on efficiency. The latter, SABRE-KNN trades some efficiency for information quality. Furthermore, we extend SABRE to the context of streaming data. Our extensive experimental evaluation demonstrates that both instantiations achieve information quality superior to schemes that extend algorithms customized for k-anonymity to t-closeness, while SABRE-AK is much faster than them as well.

The rest of this paper is organized as follows. In the next section, we review the related work. Section 3 introduces the background on which we build our framework. We present

[1] An analogous observation was made with respect to the particular problem posed by ℓ-diversity in [11].

the proposed SABRE framework and outline its two instantiations in Sect. 4, and extend it to the context of streaming data in Sect. 5. In Sect. 6, we present the results of an extensive performance study. We discuss our findings in Sect. 7 and conclude this article in Sect. 8.

2 Related work

The first suggested model by which one can anonymize data while preserving their integrity was the k-anonymity model [27]; it suggests grouping tuples into ECs of no less than k tuples, with indistinguishable QI values. Still, the problem of *optimal* (i.e., minimal-information-loss) k-anonymization is NP-hard [3,21] for $k \geq 3$ and more than one QI attribute. Thus, past research has proposed several heuristics for k-anonymization. Such schemes transform the data by *generalization* and/or *suppression*. A generalization replaces, or *recodes*, all values of a QI attribute in an EC by a single *range* that contains them. For example, QI `gender` with values *male* and *female* can be generalized to *person*, and QI *age* with values 20, 25 and 32 can be generalized to [20, 32]. Suppression is an extreme case of generalization that deletes some QI values or even tuples from the released table. Generalization for a *categorical* attribute is typically facilitated by a hierarchy over its values.

Generalization recodings can be classified as follows: A *global recoding* [4,9,12,14,27] maps all tuples with the same QI values to the same EC. On the other hand, a *local recoding* [2,6,10,32] allows tuples of the same QI values to be mapped to different generalized values (i.e., different ECs). Intuitively, ECs generated by a local recoding may, but those generated by a global recoding may *not*, *overlap* each other. The flexibility of local recoding allows for anonymizations of higher information quality [10,14,15]. Furthermore, a *single-dimensional* recoding considers the domain of each QI attribute independently of the others [14] (hence forms a *grid* over the combined QI domains); on the other hand, a *multidimensional* recoding freely defines ECs over the combined domains of all QI attributes [15].

Several recent works have developed k-anonymization schemes specialized for particular circumstances. Some approaches attempt to limit information loss with respect to predefined workloads; [9] creates ECs with a classification workload in mind; the approach of [16] is more general, catering to selected data mining tasks including classification. However, such schemes can only be applied if the intended workloads are known at the time of data publication. Another research direction explores k-anonymization in dynamic settings. Wang and Fung [28] solve the problem of k-anonymizing sequentially released views of the same underlying table. The works in [8,23] consistently anonymize multiple releases of a table that undergoes incremental updates. Cao et al. [6,7] present clustering-based schemes that anonymize streaming data on the fly and, at the same time, ensure the freshness of the anonymized data by satisfying specified delay constraint.

Still, the k-anonymity model suffers from a critical limitation. While the objective of anonymization is to conceal sensitive information about the subject involved, k-anonymity pays no attention to non-QI *sensitive* attributes (SAs). Thus, a k-anonymized table may contain ECs with so skewed a distribution of SA values, that an adversary can still infer the SA value of a record with high confidence. To address this limitation, Machanavajjhala et al. [20] extended k-anonymity to the ℓ-diversity model, which postulates that each EC contains at least ℓ "well represented" values. The requirement that values be "well represented" can be defined in diverse ways. Thus, by *entropy* ℓ-diversity, the entropy of SA values in each EC should be at least $\log \ell$; by *recursive* (c, ℓ)-diversity, it should hold that $r_1 < c(r_\ell + r_{\ell+1} + \cdots + r_m)$, where r_i is the number of occurrences of the ith most frequent SA value in a given EC, c a constant, and m the number of distinct sensitive values in that EC. Xiao and Tao [29] propose a third instantiation of ℓ-diversity, which requires that the most frequent sensitive value in any EC occurs in at most $1/\ell$ of its records.

The proposal of the ℓ-diversity model was not accompanied by an anonymization algorithm tailored for it. In response to this need, Ghinita et al. [10,11] provide a local-recoding ℓ-diversification framework that resolves the arising high-dimensional partitioning problem via a space-filling curve, such as the Hilbert curve [22]. Furthermore, Byun et al. [5] develop a diversity-aware data re-publication scheme that supports tuple insertion only. The model of m-invariance [30] supports re-publication of data undergoing both insertions and deletions of tuples.

The ℓ-diversity model is designed with a *categorical* SA in mind; it does not directly apply to the case of a *numerical* SA. Namely, a diversity of numerical SA values does not guarantee privacy when their *range* in an EC is narrow (i.e., the values are close to each other); such a narrow range can provide accurate enough information to an adversary. To address this deficiency, Zhang et al. [33] propose a model that requires the range of a numerical SA's values in an EC to be wider than a threshold. Still, an adversary may still be able to infer a numerical SA value with high confidence, if most numerical SA values in an EC are close, no matter how wide their total range is (i.e., the EC may simply contain a few outliers). Thus, Li et al. [17] propose a scheme requiring that $\frac{|g_c|}{|\mathcal{G}|} \leq 1/m$, where $\mathcal{G}$ is a given EC, g_c any group of close tuples in $\mathcal{G}$, and m a parameter.

The deficiency of ℓ-diversity outlined above is most conspicuous with numerical SAs, but not restricted to them only. It can also apply to *semantically* similar values of categorical

SA. In general, ℓ-diversity fails to guarantee privacy whenever the *distribution* of SA values within an EC differs substantially from their *overall* distribution in the released table, allowing *skewness* and *similarity* attacks. Thus, Li et al. [18] propose the t-closeness model, which requires that the difference, measured by an appropriate metric, *of* the SA distribution within any EC *from* the overall distribution of that SA be no more than a given threshold t. According to the t-closeness model, an adversary who knows the overall SA distribution in the published table gains only limited more information about an EC by seeing the SA distribution in it.

To our knowledge, three t-closeness-attaining techniques have been proposed to date. The first of them [18] extends the Incognito method for k-anonymization [14]. It operates in an iterative manner, employing a predefined generalization hierarchy over the domain of each QI attribute. In the first round, it determines the level in the generalization hierarchy of each *single* QI attribute above which t-closeness is met. In the second round, it uses the findings of the first round to establish those *combinations* of *two* QI attributes, generalized at different levels over their respective hierarchies, that achieve t-closeness (a lattice structure represents such combinations). The scheme proceeds in this manner, examining subsets of QI attributes of size increased by one at each iteration, until it establishes the valid generalizations over *all* QI attributes that satisfy t-closeness, and selects the best of those. Unfortunately, this approach shares the drawbacks of Incognito as an algorithm for k-anonymization: it is limited to single-dimensional global recoding. Thus, it achieves low information quality, while its worst-case time complexity is exponential in the number of QI attributes.

Likewise, the second t-closeness-obtaining scheme [19] extends the Mondrian k-anonymization method [15]. It recursively partitions the combined domain of all QI attributes, carrying out a split only if the resultant partitions obey t-closeness with respect to the overall distribution. While this method is more efficient than the Incognito-based one, it still fails in terms of information quality, as it does not cater to special features of t-closeness.

Recently, a scheme for t-closeness-like anonymization has been proposed [24]. Still, it uses perturbation (i.e., postrandomization [13]) and adds noise to anonymize the data; thus, it does not guarantee the *integrity* of the data, which is a basic common feature of the generalization-based techniques we examine in this article. Furthermore, [24] does not enforce the t threshold as a maximum difference constraint, but only as an *average* distance metric; it compares distributions measured over perturbed QI values (not over ECs) to that of the overall table; and it employs *KL-divergence* instead of EMD as a distance metric. Thus, the model of [24] does not provide the same worst-case privacy guarantees as t-closeness.

3 Background

Consider a database table $\mathcal{DB}(A_1, A_2, \dots, A_n)$. The *quasi-identifier* (QI) of $\mathcal{DB}$ is a subset of its attributes, $\{A_1, A_2, \dots, A_d\} \subseteq \{A_1, A_2, \dots, A_n\}$ that can, joined with an external database, reveal the identities of the tuples involved. An *equivalence class* (EC) is a group of published tuples that have the same (generalized) QI values. One of the attributes of $\mathcal{DB}$ is a *sensitive attribute* (SA). We say that $\mathcal{DB}$ satisfies t-closeness if and only if the difference *of* the distribution of its SA in any EC *from* its distribution in the whole table does not exceed a given threshold t. We measure the difference between two distributions by the Earth Mover's Distance[2] metric, outlined in the following.

3.1 The earth mover's distance metric

The *Earth Mover's Distance* (*EMD*) is suggested as a metric for quantifying the difference between distributions in [25]. Intuitively, it views one distribution as a mass of *earth* piles spread over a space, and the other as a collection of *holes*, in which the mass fits, over the same space. The *EMD* between the two is defined as the minimum *work* needed to fill the holes with earth, thereby *transforming* one distribution to the other.

Let $\mathcal{P} = (p_1, p_2, \dots, p_m)$ be the distribution of "holes", $\mathcal{Q} = (q_1, q_2, \dots, q_m)$ that of "earth", d_{ij} the *ground distance* of q_i from p_j, and $F = [f_{ij}]$, $f_{ij} \geq 0$ a flow of mass of earth moved from element q_i to p_j, $1 \leq i, j \leq m$. The *EMD* is the minimum value of the work required to transform $\mathcal{Q}$ to $\mathcal{P}$ by F:

$$\text{WORK}(\mathcal{P}, \mathcal{Q}, F) = \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} \times f_{ij}$$

For the paper to be self-contained, in the following, we present the *EMD* formulas given in [18].

In case of a numerical SA, let its *ordered* domain be $\{v_1, v_2, \dots, v_m\}$, where v_i is the ith smallest value ($\mathcal{P}$ and $\mathcal{Q}$ are distributions over these values). The distance between two values v_i, v_j in this domain is defined by the number of values between them in the total order, as $\frac{|i-j|}{m-1}$. Then the minimal work for transforming $\mathcal{Q}$ to $\mathcal{P}$ can be calculated by sequentially satisfying the *earth* needs of each *hole* element, moving earth from/to its immediate neighbor pile [18]. Thus, the *EMD* between $\mathcal{P}$ and $\mathcal{Q}$ is defined as:

$$EMD(\mathcal{P}, \mathcal{Q}) = \frac{1}{m-1} \sum_{i=1}^{m-1} \left| \sum_{j=1}^{i} (q_j - p_j) \right|$$

[2] According to [18], neither the Kullback–Leibler (KL) nor the variational distance is appropriate for evaluating the difference of two distributions, as they do not consider semantic relationships of SA values.

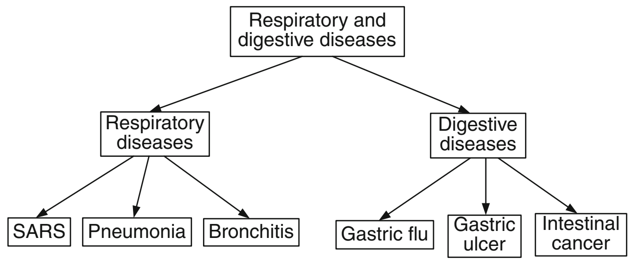

Fig. 1 The hierarchy for disease

In case of a categorical SA, we assume a generalization hierarchy $\mathcal{H}$ over its domain. For example, Fig. 1 depicts a hierarchy of *respiratory and digestive diseases*. The distance between two (leaf) values v_i and v_j is defined as $\frac{h(v_i,v_j)}{h(\mathcal{H})}$, where $h(\mathcal{H})$ is the height of $\mathcal{H}$, and $h(v_i, v_j)$ that of the lowest common ancestor of v_i and v_j in $\mathcal{H}$. To define *EMD*, we first define the following recursive function of the collective *extra earth* residing among the leaves under node n in $\mathcal{H}$.

$$extra(n) = \begin{cases} q_i - p_i, & \text{if } n \text{ is a leaf } v_i \\ \sum\limits_{c \in child(n)} extra(c), & \text{otherwise} \end{cases}$$

The value of $extra(n)$ denotes the exact amount of earth that should be moved in/out of node n. Furthermore, we define the accumulated amount of earth to be moved inwards and outwards for an *internal* node of $\mathcal{H}$:

$$nege(n) = \sum_{c \in child(n) \wedge extra(c) < 0} |extra(c)|$$

$$pose(n) = \sum_{c \in child(n) \wedge extra(c) > 0} extra(c)$$

Then the minimum of the above quantities signifies the cost of all *pending* earth movements *among* the leaves under node n, after their *cumulative* earth excess/deficit has been corrected:

$$\text{cost}(n) = \frac{h(n)}{h(\mathcal{H})} \min(pos_e(n), neg_e(n))$$

Then, the total *EMD* between $\mathcal{P}$ and $\mathcal{Q}$ is:

$$EMD(\mathcal{P}, \mathcal{Q}) = \sum_n \text{cost}(n)$$

where n is a non-leaf node in $\mathcal{H}$.

Example 1 Assume Table 1 is the input table, `{weight, age}` the QI, `disease` the SA, and Table 2 the published table. Let R, D and RD represent *respiratory diseases*, *digestive diseases*, and *respiratory and digestive diseases* respectively. Accordingly, the $\mathcal{SA}$ distribution in Table 1 is $\mathcal{P} = \left(\frac{1}{6}, \frac{1}{6}, \ldots, \frac{1}{6}\right)$, while that in EC 1 of Table 2 is $\mathcal{Q} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0\right)$. Then $extra(SARS) = extra(pneumonia) = extra(bronchitis) = \frac{1}{6}$. Thus $extra(R) = \frac{1}{2}$, $pos_e(R) = \frac{1}{2}$, and $neg_e(RD) = 0$, hence $\text{cost}(R) = 0$. Likewise, $extra(D) = -\frac{1}{2}$ and $\text{cost}(D) = 0$. In effect, $extra(RD) = 0$, and $pos_e(RD) = neg_e(RD) = \frac{1}{2}$. Thus, $\text{cost}(RD) = 1 \times \min(pos_e(RD), neg_e(RD)) = \frac{1}{2}$, and $EMD(\mathcal{P}, \mathcal{Q}) = \text{cost}(R) + \text{cost}(D) + \text{cost}(RD) = 0.5$.

3.2 Information loss metrics

The t-closeness-attaining problem calls for the enforcement of the t constraint by sacrificing as little of the accuracy in the data as possible. To quantify the information quality compromised for the sake of privacy, we need an appropriate information loss metric. Past literature has proposed various metrics, such as the *Classification Metric* [12] and the *Discernibility Metric* [4]. The best metric to use depends on the intended use of the data. We assume that the anonymized data is to be used for multiple purposes, which may not be known in advance; hence we adopt a General Loss Metric (GLM) [7,10,12,32].

Let $QI = \{A_1, A_2, \ldots, A_d\}$ and $\mathcal{G}$ be an EC. For a numerical attribute $NA \in QI$, let $[L_{NA}, U_{NA}]$ be its domain range and $[l^{\mathcal{G}}_{NA}, u^{\mathcal{G}}_{NA}]$ the generalized range of its values in $\mathcal{G}$; then the information loss with respect to NA in $\mathcal{G}$ is defined as:

$$IL_{NA}(\mathcal{G}) = \frac{u^{\mathcal{G}}_{NA} - l^{\mathcal{G}}_{NA}}{U_{NA} - L_{NA}}$$

For a categorical attribute CA, we assume a generalization hierarchy $\mathcal{H}_{CA}$ over its domain (as in Fig. 1). If a is the lowest common ancestor of all CA values in $\mathcal{G}$, then the information loss with respect to CA in $\mathcal{G}$ is defined as:

$$IL_{CA}(\mathcal{G}) = \begin{cases} 0, & Leaves(a) = 1 \\ \frac{Leaves(a)}{Leaves(\mathcal{H}_{CA})}, & \text{otherwise} \end{cases}$$

where $Leaves(a)$ is the number of leaves under the subtree of $\mathcal{H}_{CA}$ rooted at a, and $Leaves(\mathcal{H}_{CA})$ is the total number of leaves in $\mathcal{H}_{CA}$. The total information loss of $\mathcal{G}$ is then:

$$IL(\mathcal{G}) = \sum_{i=1}^{d} w_i \times IL_{A_i}(\mathcal{G})$$

where w_i is the weight of A_i and $\sum_{i=1}^{d} w_i = 1$. In our experiments, we treat each A_i as equally important, hence assign $w_i = 1/d$. The total information loss on a database table $\mathcal{DB}$, partitioned into a set $S_{\mathcal{G}}$ of ECs, is defined as:

$$AIL(S_{\mathcal{G}}) = \frac{\sum_{\mathcal{G} \in S_{\mathcal{G}}} |\mathcal{G}| \times IL(\mathcal{G})}{|\mathcal{DB}|}$$

The goal of SABRE is to keep the value of $AIL(S_{\mathcal{G}})$ low, while enforcing t-closeness on $\mathcal{DB}$.

4 The SABRE framework

In this section, we present SABRE, our framework for attaining t-closeness. We first discuss the main challenges we face in the development of SABRE, and then outline the approach itself. Table 3 gathers together the notations we employ.

4.1 Observations and challenges

SABRE consists of two phases. In the first phase, *bucketization*, it partitions $\mathcal{DB}$ into a set of buckets, such that each SA value appears in only one bucket, defined as follows.

Definition 1 (*bucket partition*) Given a table $\mathcal{DB}$, sensitive attribute SA, we say that a set of buckets φ forms a *bucket partition* of $\mathcal{DB}$ if and only if $\bigcup_{\forall \mathcal{B} \in \varphi} \mathcal{B} = \mathcal{DB}$ and each $\mathcal{SA}$ value appears in *exactly one* bucket.

In the second phase, *redistribution*, SABRE reallocates tuples from buckets to ECs. For the sake of exposition, we first consider the requirement that the number of tuples assigned to an EC from a certain bucket is proportional to that bucket's size. This *proportionality requirement* is defined as follows.

Definition 2 (*proportionality requirement*) Given a table $\mathcal{DB}$ and a bucket partition thereof φ, assume that an EC, $\mathcal{G}$, is formed with x_i tuples from bucket $\mathcal{B}_i \in \varphi, i = 1, 2, \dots, |\varphi|$. $\mathcal{G}$ abides to the *proportionality requirement* with respect to φ, if and only if the sizes of x_i are proportional to those of $\mathcal{B}_i$, i.e., $|x_1| : |x_2| : \cdots : |x_{|\varphi|}| = |\mathcal{B}_1| : |\mathcal{B}_2| : \cdots : |\mathcal{B}_{|\varphi|}|$.

Assume we create a partitioning $\varphi' = \{b_1, b_2, \dots, b_m\}$, in which bucket b_i includes those and only those tuples in $\mathcal{DB}$ that have SA value v_i. Then we select x_i tuples from bucket $b_i, i = 1, 2, \dots, |\varphi|$, to form an EC $\mathcal{G}$. In this case, if $\mathcal{G}$ follows the proportional requirement with respect to φ', then it also holds that $|x_1| : |x_2| : \cdots : |x_m| = N_1 : N_2 : \cdots : N_m$, where $N_i = |b_i|$ is the number of tuples with SA value v_i in $\mathcal{DB}$. In effect, $\mathcal{G}$ is 0-close to $\mathcal{DB}$; thus, an adversary gains no extra information by seeing $\mathcal{G}$. Still, a complete enforcement of 0-closeness for all ECs would severely degrade information quality. This is not what we aim for; we wish to allow for some loss of privacy, delimited by the t-closeness constraint, in order to preserve more in terms of information quality. Thus, we need to opt for a more flexible arrangement in our scheme.

To that end, we can start out with buckets of *more than one* distinct SA value. We slice $\mathcal{DB}$ into an alternative bucket partition $\varphi = \{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_{|\varphi|}\}$, in which each bucket $\mathcal{B}_i$ may contain multiple *semantically close* SA values. In this case, an EC $\mathcal{G}$ that satisfies the proportionality requirement with respect to the buckets in φ does *not* necessarily obey the relationship $|z_1| : |z_2| : \cdots : |z_m| = N_1 : N_2 : \cdots : N_m$, where z_i is the number of tuples with SA value v_i in $\mathcal{G}, i = 1, 2, \dots, m$. After all, when we pick tuples from a bucket $\mathcal{B}_i$ to form $\mathcal{G}$, we do *not* discriminate between different $\mathcal{SA}$ values. The following example illustrates the two EC compositions described above.

Example 2 Consider Table 1, where `{weight, age}` is the QI, and `disease` is the SA. Figure 2 shows the QI-space and the distribution of tuples from Table 1 (each QI attribute corresponds to a dimension). A bucket partition φ' of this table could consist of six buckets of one tuple each, $b_1, b_2, \dots, b_6$, with SA values *SARS*, *pneumonia*, *bronchitis*, *intestinal cancer*, *gastric flu*, and *gastric ulcer*, respectively. Taking one tuple from each of these buckets, we could build a single EC of 0-close privacy. Still, such an EC covers the entire QI-space, incurring high information loss. Another bucket partition could consist of two buckets of three semantically similar tuples each, $\varphi = \{\mathcal{B}_1, \mathcal{B}_2\}$, with *SARS*, *pneumonia*, and *bronchitis* in bucket $\mathcal{B}_1$ and the rest in $\mathcal{B}_2$. We can then build three ECs, by taking one tuple from each of these buckets, as shown in Fig. 2. Tuples in the same EC are labeled by the same number in the figure. This EC partitioning achieves better information quality, as the minimum bounding boxes of ECs in QI-space are smaller.

An equivalence class $\mathcal{G}$ constructed from a looser bucket partition achieves higher information quality, but is no longer 0-close. Still, it suffices to construct it in a manner that obeys t-closeness for a given t. In Example 2, all three ECs generated from φ are $\frac{1}{3}$-close with respect to the distribution of `disease` in Table 1, hence satisfying t-closeness for $t \geq \frac{1}{3}$.

Following the aforementioned observations, SABRE first partitions tuples according to their SA values, and then redistributes the tuples to ECs. In order to ensure t-closeness and

Table 3 Employed notations

Notation	Denotation
$\mathcal{DB}$	A microdata table (original table)
SA	The sensitive attribute in $\mathcal{DB}$
$\mathcal{V} = \{v_1, v_2, \dots, v_m\}$	The domain of SA
N_i	The number of tuples with v_i in $\mathcal{DB}$
$p_i = N_i/\|\mathcal{DB}\|$	The distribution of value v_i in $\mathcal{DB}$
$\mathcal{P} = (p_1, p_2, \dots, p_m)$	Overall distribution of SA in $\mathcal{DB}$
$\mathcal{G}$	An equivalence class
$\mathcal{Q} = (q_1, q_2, \dots, q_m)$	The distribution of SA in $\mathcal{G}$

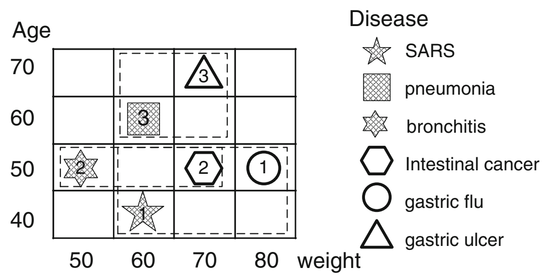

Fig. 2 Information quality under SABRE

good information quality, we need to address the following questions:

1. How should we partition SA values into buckets? How many buckets should we generate to ensure t-closeness?
2. How many ECs should we generate? How should we choose tuples from each bucket to form an EC?

Next we present our approaches to these questions.

4.2 SABRE's bucketization scheme

We commence the presentation of our bucketization scheme with a property of the proportionality requirement.

Consider a categorical SA with the domain hierarchy in Fig. 1. Assume that a table $\mathcal{DB}$ contains 50 tuples with *SARS*, 30 with *Pneumonia*, 20 with *Bronchitis*, 40 with *Gastric flu*, 20 with *Gastric ulcer*, and 20 with *Intestinal cancer*. Suppose a bucket partition $\varphi = \{\mathcal{B}_1, \mathcal{B}_2\}$ of $\mathcal{DB}$, where $\mathcal{B}_1$ contains all tuples with *SARS*, *Pneumonia* and *Bronchitis*, and $\mathcal{B}_2$ includes tuples with the remaining three SA values. The overall SA distribution is $\mathcal{P} = \left(\frac{5}{18}, \frac{3}{18}, \frac{2}{18}, \frac{4}{18}, \frac{2}{18}, \frac{2}{18}\right)$. Then an EC, $\mathcal{G}$, with 10 tuples from $\mathcal{B}_1$ and 8 tuples from $\mathcal{B}_2$ satisfies the proportionality requirement with respect to φ. For instance, $\mathcal{G}$ may comprise 4 tuples with *SARS*, 2 with *Pneumonia*, 4 with *Bronchitis*, 0 with *Gastric flu*, 4 with *Gastric ulcer*, and 4 with *Intestinal cancer*. Then the $\mathcal{SA}$ distribution in $\mathcal{G}$ is $\mathcal{Q} = \left(\frac{4}{18}, \frac{2}{18}, \frac{4}{18}, 0, \frac{4}{18}, \frac{4}{18}\right)$.

Now, we can divide the elements (i.e., holes) of $\mathcal{P}$ in two subsets, $H_1 = \left\{\frac{5}{18}, \frac{3}{18}, \frac{2}{18}\right\}$ and $H_2 = \left\{\frac{4}{18}, \frac{2}{18}, \frac{2}{18}\right\}$, and, likewise, the piles of earth in $\mathcal{Q}$ in $E_1 = \left\{\frac{4}{18}, \frac{2}{18}, \frac{4}{18}\right\}$ and $E_2 = \left\{0, \frac{4}{18}, \frac{4}{18}\right\}$, corresponding to the division of SA values into respiratory diseases in $\mathcal{B}_1$ and digestive diseases in $\mathcal{B}_2$. Then the volume of holes in H_1 is $\frac{5}{18} + \frac{3}{18} + \frac{2}{18} = \frac{10}{18}$, equal to the volume of earth in E_1, $\frac{4}{18} + \frac{2}{18} + \frac{4}{18} = \frac{10}{18}$. Likewise, the volume of holes in H_2 equals that of earth in E_2. In effect, the transformation from $\mathcal{Q}$ to $\mathcal{P}$ can be decomposed in two *independent* subtasks: filling the holes in H_1 with earth from E_1, and those in H_2 with earth from E_2. We name such a subtask *earth transportation in a bucket*, defined as follows.

Definition 3 (*earth transportation in a bucket*) Assume a bucket partition φ of a table $\mathcal{DB}$ with sensitive attribute SA, and any equivalence class $\mathcal{G}$ that follows the proportionality requirement with respect to φ. Without loss of generality, assume bucket $\mathcal{B} \in \varphi$ contains the $\mathcal{SA}$ values $v_1, v_2, \ldots, v_j$. Then, *earth transportation in $\mathcal{B}$ with regard to $\mathcal{G}$* is the transformation from $(q_1, q_2, \ldots, q_j)$ to $(p_1, p_2, \ldots, p_j)$, where q_i is the distribution of v_i in $\mathcal{G}$ and p_i is the distribution of v_i in $\mathcal{DB}$, $i = 1, 2, \ldots, j$.

We denote the $\underline{c}$ost of this $\underline{e}$arth $\underline{t}$ransportation in bucket $\mathcal{B}$ with regard to EC $\mathcal{G}$ as $CET(\mathcal{B}, \mathcal{G})$. Once a table $\mathcal{DB}$ is given, $p_1, p_2, \ldots, p_j$ are fixed. But the values of $q_1, q_2, \ldots, q_j$ depend on the EC $\mathcal{G}$ at hand. For instance, in our running scenario, the distribution of the 10 tuples from $\mathcal{B}_1$ in $\mathcal{G}$ among values *SARS*, *Pneumonia*, and *Bronchitis* is $q_1 = \frac{4}{18}$, $q_2 = \frac{2}{18}$, and $q_3 = \frac{4}{18}$. If the 10 tuples are all *Bronchitis*, then $q_1 = q_2 = 0$ and $q_3 = \frac{10}{18}$. Actually, they could be any 10 tuples from $\mathcal{B}_1$. Still, we are interested in the worst-case value of $CET(\mathcal{B}, \mathcal{G})$ over all possible ECs following the proportionality requirement to φ. Thus, we define an upper bound of $CET(\mathcal{B}, \mathcal{G})$ as follows.

Definition 4 (*upper-bound cost in a bucket*) Assume a bu-cket partition φ of a table $\mathcal{DB}$ with sensitive attribute SA, and a bucket $\mathcal{B} \in \varphi$. Then, we define $CET^U_{\mathcal{B}}$, the *upper-bound cost of earth transportation in $\mathcal{B}$*, as the highest possible value of $CET(\mathcal{B}, \mathcal{G})$ over *all possible* equivalence classes $\mathcal{G}$ that follow the proportionality requirement to φ.

$$CET^U_{\mathcal{B}} = \max_{\forall \mathcal{G}}\{CET(\mathcal{B}, \mathcal{G})\}$$

In the following, we present three theorems that form the foundation of SABRE. Let φ be a bucket partition of table $\mathcal{DB}$, and $\mathcal{G}$ be an EC following proportionality requirement with respect to φ. Theorem 1 formalizes the intuition gained from the above scenario—it essentially tells us that earth transportation in a bucket can be independent from that of any other bucket. Given a bucket $B \in \varphi$, Theorem 2 determines $CET^U_{\mathcal{B}}$. Based on the above two, Theorem 3 states that we can compute the cost of transforming the $\mathcal{SA}$ distribution in $\mathcal{G}$ to that in $\mathcal{DB}$, by the summation of the upper bounds related with all buckets in φ.

Theorem 1 (Independence) *Let $\mathcal{G}$ be an EC that follows the proportionality requirement with respect to a bucket partition φ of $\mathcal{DB}$ with sensitive attribute SA. Given any bucket $\mathcal{B} \in \varphi$, the earth transportation in $\mathcal{B}$ with regard to $\mathcal{G}$ is independent from buckets in $\varphi \setminus \{\mathcal{B}\}$.*

Proof Without loss of generality, assume that $\mathcal{B}$ contains tuples with $\mathcal{SA}$ values of $v_1, v_2, \ldots, v_j$, and let z_i be the set of tuples in $\mathcal{G}$ with $\mathcal{SA}$ value of v_i, $i = 1, 2, \ldots, j$. We consider $\{p_1, p_2, \ldots, p_j\}$, $p_i = \mathcal{N}_i/|\mathcal{DB}|$, as the set of holes, and $\{q_1, q_2, \ldots, q_j\}$, $q_i = |z_i|/|\mathcal{G}|$, as the piles of earth. Given that $\mathcal{G}$ follows the proportionality requirement with respect to φ, the number of tuples from $\mathcal{B}$ assigned to $\mathcal{G}$ is $\sum_{i=1}^{j} |z_i| = \frac{|\mathcal{B}|}{|\mathcal{DB}|} \cdot |\mathcal{G}|$. Then,

$$\sum_{i=1}^{j} q_i = \frac{1}{|\mathcal{G}|} \sum_{i=1}^{j} |z_i| = \frac{|\mathcal{B}|}{|\mathcal{DB}|} = \frac{1}{|\mathcal{DB}|} \sum_{i=1}^{j} N_i = \sum_{i=1}^{j} p_i$$

Thus, the volume of earth equals the volume of holes, hence earth transportation between them can be done locally, i.e., the transformation from $(q_1, q_2, \ldots, q_j)$ to $(p_1, p_2, \ldots, p_j)$ can be independent from the earth transportation of any other bucket in $\varphi \setminus \{\mathcal{B}\}$. □

Theorem 2 (Upper bound) *Let φ be a bucket partition of $\mathcal{DB}$ with sensitive attribute SA; assume that bucket $\mathcal{B} \in \varphi$ contains $\mathcal{SA}$ values of $v_1, v_2, \ldots, v_j$. Then $CET^U_{\mathcal{B}}$, the upper-bound cost of earth transportation in $\mathcal{B}$, is determined as follows:*

– *For a categorical SA,*

$$CET^U_{\mathcal{B}} = \frac{h(n)}{h(\mathcal{H})} \cdot \left(\sum_{i=1}^{j} p_i - \min\{p_1, p_2, \ldots, p_j\} \right),$$

where $\mathcal{H}$ is the domain hierarchy of SA and n is the lowest common ancestor of $v_1, \ldots, v_j$.

– *For a numerical SA,*

$$CET^U_{\mathcal{B}} = \max_{\ell=1,2,\ldots,j} \left\{ \sum_{i=1}^{j} d_{\ell i} \times p_i \right\},$$

where $d_{\ell i}$ is the distance between v_ℓ and v_i.

Proof Again, we consider $\{p_1, p_2, \ldots, p_j\}$ as a collection of holes, and $\{q_1, q_2, \ldots, q_j\}$ as piles of earth, where q_i is the distribution of v_i in $\mathcal{G}$ (an EC following proportionality requirement with respect to φ). By Theorem 1, $\sum_{i=1}^{j} p_i = \sum_{i=1}^{j} q_i$.

Categorical SA. We divide the set of holes in two subsets: The subset of holes "missing earth", $H_1 = \{p_\ell | p_\ell > q_\ell, 1 \leq \ell \leq j\}$, and that of holes "in excess of earth", $H_2 = \{p_\ell | p_\ell \leq q_\ell, 1 \leq \ell \leq j\}$. Likewise, we separate the set of earth-piles in two corresponding subsets: that of "deficient" piles, $E_1 = \{q_\ell | p_\ell > q_\ell, 1 \leq \ell \leq j\}$, and that of "superfluous" piles, $E_2 = \{q_\ell | p_\ell \leq q_\ell, 1 \leq \ell \leq j\}$. The earth transportation in $\mathcal{B}$ involving $\mathcal{G}$ is done by two steps.

In the first step, we fill up hole p_i with earth q_i, $i = 1, 2, \ldots, j$. Since the distance between p_i and q_i is 0, the cost of this step is 0. Still, all the earth of deficient piles in E_1 is used up; thus, in order to fill up the holes missing earth in H_1, we need extra earth of $neg_e(n) = \sum_{\forall q_\ell \in E_1} (p_\ell - q_\ell)$. Symmetrically, the holes in excess of earth in H_2 are completely filled, and the superfluous earth-piles in E_2 have extra earth of exactly $pos_e(n) = \sum_{\forall q_\ell \in E_2} (q_\ell - p_\ell)$. From the independence of earth transportation within $\mathcal{B}$, i.e., from $\sum_{i=1}^{j} p_i = \sum_{i=1}^{j} q_i$, it follows that $neg_e(n) = pos_e(n)$, as we would expect.

In the second step, we have to move $neg_e(n)$ earth from the superfluous piles in E_2 to the holes missing earth in H_1. Since the distance between any two elements in $\{v_1, \ldots, v_j\}$ is at most $\frac{h(n)}{h(\mathcal{H})}$, the cost of the whole earth movement is at most $\frac{h(n)}{h(\mathcal{H})} \cdot neg_e(n)$. However, $neg_e(n) \leq \sum_{\forall p_\ell \in H_1} p_\ell$. Besides, because there is at least one hole in H_2 (i.e., at least one superfluous pile), it follows that $\sum_{\forall p_\ell \in H_1} p_\ell \leq \sum_{i=1}^{j} p_i - \min\{p_1, p_2, \ldots, p_j\}$. Putting it all together, we get $CET(\mathcal{B}, \mathcal{G}) \leq \frac{h(n)}{h(\mathcal{H})} \cdot \left(\sum_{i=1}^{j} p_i - \min\{p_1, p_2, \ldots, p_j\}\right)$.

Numerical SA. We scan the holes $\{p_1, p_2, \ldots, p_j\}$ sequentially, and fill up every hole in need of earth that we encounter. For each such hole, we use earth from its *nearest* pile, resolving ties arbitrarily. If the nearest pile is used up, we move earth from its second nearest pile. This continues until the hole is filled up. Thus, we transfer earth in the most affordable way. Let q_ℓ, $1 \leq \ell \leq j$ be the last pile that the process takes earth from. After q_ℓ is used up, all the holes are filled. Using q_ℓ, we divide the holes $\{p_1, p_2, \ldots, p_j\}$ into two groups: H_1, holes that are entirely filled by earth hailing from q_ℓ, and H_2, the rest. We name the above as *sequential process*, and denote its cost by seq_c. The following situation, which has all earth concentrated in the ℓth pile, corresponds to distribution $(\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_j)$, where $\tilde{q}_\ell = \sum_{i=1}^{j} p_i$ and $\tilde{q}_i = 0$ if $i \neq \ell$. The cost of the given situation (by transforming $(\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_j)$ to $(p_1, p_2, \ldots, p_j)$) is an upper bound of seq_c. That is because filling a hole in H_1 sequential process has the same cost as the given situation; on the other hand, filling a hole in H_2 the former costs less than the latter, because in the former case some earth is brought to the hole from a pile that is, by the definition of H_2, closer to it than pile q_ℓ. The cost of the given situation is $\sum_{i=1}^{j} (d_{\ell i} \times p_i)$. We cannot know which pile q_ℓ is; thus, we consider the worst-case scenario, i.e., the maximum out of all ℓ. In effect, the cost to transform $\{q_1, q_2, \ldots, q_j\}$ to $\{p_1, p_2, \ldots, p_j\}$ is upper-bounded by $\max_{\ell=1,2,\ldots,j} \left\{ \sum_{i=1}^{j} d_{\ell i} \times p_i \right\}$.

Both upper bounds are tight. The bounds are reached when the tuples from $\mathcal{B}$ assigned to $\mathcal{G}$ all have the same SA value v_ℓ. For categorial SA, v_ℓ is the least frequent value among $v_1, v_2, \ldots, v_j$, with $p_\ell = \min\{p_1, p_2, \ldots, p_j\}$. For numerical SA, v_ℓ is the value for which $\sum_{i=1}^{j} (d_{\ell i} \times p_i)$ is maximized. □

Theorem 3 (Additivity) *Let $\mathcal{G}$ be any EC that follows the proportionality requirement with respect to a bucket partition φ of table $\mathcal{DB}$ with sensitive attribute SA. Then the EMD of transforming the SA distribution $\mathcal{Q}$ in $\mathcal{G}$ to the distribution $\mathcal{P}$ in $\mathcal{DB}$ is upper bounded by $\sum_{\forall \mathcal{B}\in\varphi} CET^U_{\mathcal{B}}$.*

Proof In order to transform $\mathcal{Q}$ to $\mathcal{P}$, we need to carry out the earth transportation in each bucket $\mathcal{B} \in \varphi$. Since these transportations do not affect each other (Theorem 1), the *EMD* required to transform $\mathcal{Q}$ to $\mathcal{P}$ is the sum of their costs, and the upper bound of this sum is the sum of the upper bounds to the individual costs. □

Example 3 Consider once again our running scenario. We still assume that *SARS*, *pneumonia*, and *bronchitis* are in $\mathcal{B}_1$, and $\mathcal{B}_2$ comprises tuples of the remaining three SA values. We select 10 tuples from $\mathcal{B}_1$ and 8 tuples from $\mathcal{B}_2$ to form $\mathcal{G}$. If the 10 tuples from $\mathcal{B}_1$ all have SA value *bronchitis*, then $(q_1, q_2, q_3) = \left(0, 0, \frac{10}{18}\right)$ and the cost of earth transportation in $\mathcal{B}_1$ reaches its upper bound $\frac{1}{2} \cdot [(p_1 + p_2 + p_3) - \min\{p_1, p_2, p_3\}] = \frac{2}{9}$, where $(p_1, p_2, p_3) = \left(\frac{5}{18}, \frac{3}{18}, \frac{2}{18}\right)$. This upper bound is realized by moving $\frac{5}{18}$ earth from q_3 to p_1 (cost is $\frac{1}{2} \times \frac{5}{18}$), $\frac{3}{18}$ earth from q_3 to p_2 (cost is $\frac{1}{2} \times \frac{3}{18}$), and $\frac{2}{18}$ earth from q_3 to p_3 (cost is $0 \times \frac{2}{18}$). Likewise, if the 8 tuples from $\mathcal{B}_2$ are all with *gastric ulcer* (or *intestinal cancer*), then the cost of earth transportation in $\mathcal{B}_2$ reaches its upper bound $\frac{1}{2} \cdot [(p_4 + p_5 + p_6) - \min\{p_4, p_5, p_6\}] = \frac{1}{6}$. With 10 tuples of *bronchitis* and 8 of *gastric ulcer*, the SA distribution in $\mathcal{G}$ is $\mathcal{Q} = (0, 0, \frac{10}{18}, 0, \frac{8}{18}, 0)$. The overall $\mathcal{SA}$ distribution in $\mathcal{DB}$ is $\mathcal{P} = (\frac{5}{18}, \frac{3}{18}, \frac{2}{18}, \frac{4}{18}, \frac{2}{18}, \frac{2}{18})$. After the earth transportations in B_1 and B_2, $\mathcal{Q}$ is transformed to $\mathcal{P}$ with a cost upper bounded by $CET^U_{\mathcal{B}_1} + CET^U_{\mathcal{B}_2} = \frac{2}{9} + \frac{1}{6} = \frac{7}{18}$.

After the above foundations, we can now discuss the generation of buckets. SABRE partitions $\mathcal{DB}$ hierarchically, based on the SA values of its tuples, forming a *bucketization tree*. Each node of this tree denotes a bucket containing tuples having a certain subset of SA values. The leaf nodes of the tree are the buckets that correspond to the actual bucket partition of $\mathcal{DB}$. The tree starts with a single node—the root—which corresponds to the entire table with the whole domain of $\mathcal{SA}$. Then the tree grows in a top-down manner by recursively splitting leaf nodes. In each iteration, we can compute the upper bounded cost of each node/bucket (based on Theorem 2). By Theorem 3, we determine $\mathcal{U}$, the summation of all the upper bounds. In this way, we select the node that contributes to the largest reduction of $\mathcal{U}$ as the node to be further split. This process terminates when $\mathcal{U}$ becomes smaller than the closeness threshold t. By Theorem 3, this termination condition guarantees that the SA distribution in any EC formed from the final buckets according to the *proportionality requirement* will not differ from that in $\mathcal{DB}$ by more than t.

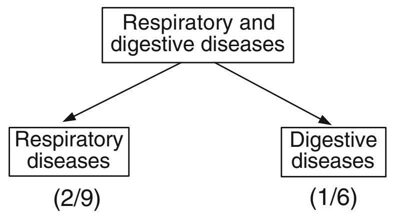

Fig. 3 Bucketization of Disease at *RDD*

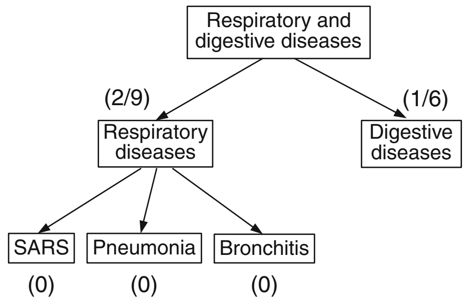

Fig. 4 Bucketization of Disease at *RD*

We now elaborate on the way we split a leaf bucket/node n. For a categorical SA, let $\mathcal{H}$ be the domain hierarchy of $\mathcal{SA}$. Then each node in the bucketization tree has a corresponding node in $\mathcal{H}$. The root of the tree matches the root of $\mathcal{H}$. When splitting n, its new children are the children of its corresponding node in $\mathcal{H}$.

Example 4 Let `disease` be a categorical SA with the domain hierarchy of Fig. 1. Assume a table $\mathcal{DB}$, containing 5 tuples with *SARS*, 3 with *Pneumonia*, 2 with *Bronchitis*, 4 with *Gastric flu*, 2 with *Gastric ulcer*, and 2 with *Intestinal cancer*. The resultant distribution is $\left(\frac{5}{18}, \frac{3}{18}, \frac{2}{18}, \frac{4}{18}, \frac{2}{18}, \frac{2}{18}\right)$. Assume a threshold $t = 0.2$. We build the bucketization tree in a top-down fashion as follows. At the beginning, the tree comprises the root, *respiratory and digestive diseases*. The upper bound cost of the root is calculated as $1 \times \left(\left(\frac{5}{18} + \cdots + \frac{2}{18} + \frac{2}{18}\right) - \min\{\frac{5}{18}, \ldots, \frac{2}{18}, \frac{2}{18}\}\right) = 1 \times \left(1 - \frac{2}{18}\right) = 0.889$. Since $0.889 > 0.2$, we split it. We add its two children *respiratory diseases, digestive diseases* to the tree, as in Fig. 3. Now the upper bound cost of node *respiratory diseases* is $\frac{1}{2} \times \left(\frac{10}{18} - \frac{2}{18}\right) = \frac{2}{9}$, and that of *digestive diseases* is $\frac{1}{2} \times \left(\frac{8}{18} - \frac{2}{18}\right) = \frac{1}{6}$. Since $\frac{2}{9} + \frac{1}{6}$ is still larger than t, we further split the tree. Splitting *respiratory diseases* would reduce $\mathcal{U}$ by $\frac{2}{9}$, whereas splitting *digestive diseases* would reduce it by $\frac{1}{6}$; thus, we opt for the former. We add the three children of node *respiratory diseases* to the tree. Now the sum of upper bounds of all leaf nodes is $\frac{1}{6} < t$. Figure 4 shows the final tree.

On the other hand, for a numerical SA, the children of n are dynamically determined. Let lv be the set of SA values included in n, sorted in ascending order. We split lv to a

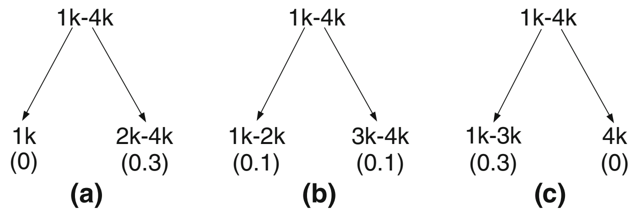

Fig. 5 Bucketization of salary at 1 k−4 k

left child c_1, containing values in lv to the left of the splitting point, and a right child c_2 with the remaining ones. The splitting point is the one that minimizes $CET^U_{c_1} + CET^U_{c_2}$.

Example 5 Let `salary` be a numerical SA with values $1k, 2k, 3k, 4k$. Assume that a table $\mathcal{DB}$ contains 2 tuples with $1k$, 3 with $2k$, 3 with $3k$, and 2 with $4k$. Then the salary distribution is $(0.2, 0.3, 0.3, 0.2)$. We label the four values from 1 to 4 sequentially. The upper bound cost for a bucketization tree composed only of the root $(1k - 4k)$ is $d_{11} \times 0.2 + d_{12} \times 0.3 + d_{13} \times 0.3 + d_{14} \times 0.2 = 0 + \frac{1}{3} \times 0.3 + \frac{2}{3} \times 0.3 + 1 \times 0.2 = 0.5$. If we set the splitting point at $1k$, then the left child of $1k - 4k$ will be $1k$, and its right child $2k - 4k$. The upper bound cost of $1k$ is 0, and that of $2k - 4k$ is $d_{42} \times 0.3 + d_{43} \times 0.3 + d_{44} \times 0.2 = 0.3$, as shown in Fig. 5a. Figure 5b, c depict the corresponding trees for splitting along $2k$ and $3k$. The minimum sum of upper bounds is that in Fig. 5b. Thus, for a threshold $t = 0.25$, the tree in Fig. 5b is the final tree.

Function bucketCat ($\mathcal{H}$, $\mathcal{VP}$)

```
1  Let $\mathcal{L}$ be the set of leaf nodes in the bucketization tree;
2  Initialize $\mathcal{L}$ to be empty;
3  Let $r$ be the root of $\mathcal{H}$;
4  Calculate $CET^U_r$;
5  foreach child $c$ of $r$ in $\mathcal{H}$ do
6      calculate $CET^U_c$;
7  $dv_r = CET^U_r - \sum_{c \in child(r)} CET^U_c$;
8  Add $r$ to $\mathcal{L}$ and initialize $\mathcal{U} = CET^U_r$;
9  while $\mathcal{U} \geq t$ do
10     Let $n$ be the node in $\mathcal{L}$ with the maximum $dv$ value;
11     Remove $n$ from $\mathcal{L}$;
12     foreach child $c$ of $n$ in $\mathcal{H}$ do
13         Add $c$ to $\mathcal{L}$;
14     $\mathcal{U} = \mathcal{U} - dv_n$;
15     if $\mathcal{U} < t$ then
16         break;
17     foreach child $c$ of $n$ do
18         foreach child $gc$ of $c$ in $\mathcal{H}$ do
19             Calculate $CET^U_{gc}$;
20         $dv_c = CET^U_c - \sum_{gc \in child(c)} CET^U_{gc}$;
21 Let $\varphi$ be the set of buckets related with nodes in $\mathcal{L}$;
22 Return $\varphi$;
```

Function bucketCat generates buckets for a categorical SA. Input parameter $\mathcal{H}$ is the domain hierarchy of SA; $\mathcal{VP}$ is the list of (v_i, p_i) pairs, the SA value and its frequency in the whole table, $i = 1, 2, \ldots, m$. $\mathcal{L}$ stores all the leaf nodes of the bucketization tree (steps 1–2). We use a node to represent its corresponding bucket. Step 4 calculates the upper bound cost of the root r. Steps 5–7 compute the potential cost reduction after splitting r (dv denotes the decreased value). The root r is the first node added to $\mathcal{L}.\mathcal{U}$, the sum of all upper bounds of the nodes in $\mathcal{L}$, is initialized to CET^U_r (step 8). The leaf node n whose splitting reduces $\mathcal{U}$ at most is split (step 10), n is replaced by its children in $\mathcal{L}$ (steps 11–13), and $\mathcal{U}$ is accordingly reduced (step 14). Steps 17–20 calculate how much $\mathcal{U}$ can be deduced if n's child is split. This process continues iteratively (steps 9–20) until $\mathcal{U} < t$ (steps 15–16). Eventually, each node in $\mathcal{L}$ is associated to a bucket in φ, and φ is returned (steps 21–22).

Function bucketNum ($\mathcal{VP}$)

```
1  Let $\mathcal{L}$ be the set of leaf nodes in the bucketization tree;
2  Initialize $\mathcal{L}$ to be empty;
3  Create root $r$ and initialize its $\mathcal{SA}$ values $lv_r$ to be $\mathcal{VP}$;
4  Calculate $CET^U_r$;
5  Create $c_1$ and $c_2$, the left and right children of $r$;
6  split ($r, c_1, c_2$);
7  $dv_r = CET^U_r - CET^U_{c_1} - CET^U_{c_2}$;
8  Add $r$ to $\mathcal{L}$ and initialize $\mathcal{U} = CET^U_r$;
9  while $\mathcal{U} \geq t$ do
10     Find the node $n$ in $\mathcal{L}$ with the maximum value of $dv$;
11     Remove $n$ from $\mathcal{L}$;
12     foreach child $c$ of $n$ do
13         Add $c$ to $\mathcal{L}$;
14     $\mathcal{U} = \mathcal{U} - dv_n$;
15     if $\mathcal{U} < t$ then
16         break;
17     foreach child $c$ of $n$ do
18         Create $g_1$ and $g_2$, the left and right children of $c$;
19         split ( $c, g_1, g_2$ );
20         $dv_c = CET^U_c - CET^U_{g_1} - CET^U_{g_2}$;
21 Let $\varphi$ be the set of buckets related with nodes in $\mathcal{L}$;
22 Return $\varphi$;
```

Similarly, Function bucketNum generates buckets of a numerical SA. Input parameter $\mathcal{VP}$ is the list of SA values and their frequencies in the whole table, sorted in ascending order of SA values. Each node in the bucketization tree has a container lv that records all SA values covered by that node. The root r has lv equal to $\mathcal{VP}$, i.e. r covers all $\mathcal{SA}$ values (step 3). Procedure split calculates the best gain of dividing r into two child nodes (step 6) and r is added as the first node to $\mathcal{L}$ (step 8). CET^U_r (step 4) is assigned as the initial value of $\mathcal{U}$ (step 8). Then, nodes in $\mathcal{L}$ are split iteratively until $\mathcal{U}$ becomes smaller than the closeness threshold t (steps 9–20). In each round, the most cost-reducing node n is chosen from $\mathcal{L}$ to be split (step 10), and replaced by its two children (steps 11–13). The gain of splitting the children of n is also calculated (steps 17–20). The best grandchildren for n are dynamically determined by Procedure split (step 19). Eventually, each node in $\mathcal{L}$ is associated to a bucket, and the set of all buckets φ is returned (steps 21–22).

Procedure split (n, c_1, c_2)

```
1  $u_1 = 1.0, u_2 = 1.0$ ;
2  foreach split point $SP$ in $lv_n$ do
3      Clear $lv_{c_1}$ and $lv_{c_2}$;
4      Push all elements in $lv_n$ to the left of $SP$ into $lv_{c_1}$;
5      Allocate all remaining elements of $lv_n$ to $lv_{c_2}$;
6      Calculate $CET^U_{c_1}$ and $CET^U_{c_2}$;
7      if $CET^U_{c_1} + CET^U_{c_2} < u_1 + u_2$ then
8          $u_1 = CET^U_{c_1}$;
9          $u_2 = CET^U_{c_2}$;
10         $bestP = SP$;
11 Clear $lv_{c_1}$ and $lv_{c_2}$;
12 Push all elements in $lv_n$ to the left of $bestP$ into $lv_{c_1}$;
13 Allocate all the remaining elements of $lv_n$ to $lv_{c_2}$;
14 $CET^U_{c_1} = u_1$;
15 $CET^U_{c_2} = u_2$;
```

Procedure split dynamically divides a node n into a left child c_1 and a right child c_2. Each possible splitting point is tested, and the one that minimizes $CET^U_{c_1} + CET^U_{c_2}$ is selected (steps 2–10). The elements in lv_n are appropriately assigned to the two children (steps 12–13).

4.3 SABRE's redistribution scheme

The bucketization phase delivers a set of buckets φ, such that $\sum_{\forall \mathcal{B} \in \varphi} CET^U_{\mathcal{B}} < t$. To generate an equivalence class $\mathcal{G}$ conforming to the *proportionality requirement*, we need to select $|\mathcal{G}| \cdot \frac{|\mathcal{B}_i|}{|\mathcal{DB}|}$ tuples from bucket $\mathcal{B}_i$, $i = 1, 2, \ldots, |\varphi|$. However, $|\mathcal{G}| \cdot \frac{|\mathcal{B}_i|}{|\mathcal{DB}|}$ may not be an integer for some sizes of $\mathcal{G}$ and some $i \in \{1, 2, \ldots, |\varphi|\}$. Setting a constraint to the size of $\mathcal{G}$ so that each $|\mathcal{G}| \cdot \frac{|\mathcal{B}_i|}{|\mathcal{DB}|}$ be an integer may severely limit the allowed EC size, hence defeat the purpose of our study, which is to provide a flexible and quality-aware scheme for t-closeness. For example, assume that $|\mathcal{DB}| = 50,000$ and $\frac{|\mathcal{B}_i|}{|\mathcal{DB}|} = 0.1333$ for some $i \in \{1, 2, \ldots, |\varphi|\}$. Then each EC should have a size of at least $10,000$. Such large ECs generally incur high information loss.

We conclude that we should better relax the proportionality requirement: it suffices that the number of tuples from each bucket $\mathcal{B}_i$ in an EC $\mathcal{G}$ be *approximately* proportional to the size of the bucket. To ensure t-closeness, we determine the size of each EC dynamically. Theorem 4 establishes that this is always possible. Before presenting it, we introduce some auxiliary concepts and notations.

Definition 5 (*φ distribution in an EC*) Let $\mathcal{G}$ be an EC from table $\mathcal{DB}$ with bucket partition φ. The φ distribution in $\mathcal{G}$, denoted by $d(\mathcal{G}, \varphi)$, is $\left(\frac{|x_1|}{|\mathcal{G}|}, \frac{|x_2|}{|\mathcal{G}|}, \ldots, \frac{|x_{|\varphi|}|}{|\mathcal{G}|}\right)$, where x_i is the set of tuples from bucket $\mathcal{B}_i \in \varphi$ in $\mathcal{G}$ and $\bigcup_{i=1}^{|\varphi|} x_i = \mathcal{G}$.

The φ distribution of the whole table $\mathcal{DB}$ as a single EC is $d(\mathcal{DB}, \varphi) = \left(\frac{|\mathcal{B}_1|}{|\mathcal{DB}|}, \frac{|\mathcal{B}_2|}{|\mathcal{DB}|}, \ldots, \frac{|\mathcal{B}_{|\varphi|}|}{|\mathcal{DB}|}\right)$. Furthermore, if $\mathcal{G}$ conforms to the proportionality requirement with respect to φ, then $d(\mathcal{G}, \varphi) = d(\mathcal{DB}, \varphi)$. Given two φ distributions, we *define* d^u_{ij} to be the distance between element i of the former and element j of the latter. Let d_{yz} be the *ground distance* between two SA values v_y and v_z, and $\mathcal{V}_i$ be the set of SA values in bucket $\mathcal{B}_i \in \varphi$, then we define our d^u_{ij} metric as follows.

$$d^u_{ij} = \begin{cases} \max\{d_{yz} | v_y \in \mathcal{V}_i, v_z \in \mathcal{V}_j\}, & i \neq j \\ 0, & i = j \end{cases}$$

We can transform the SA distribution $\mathcal{Q}$ in an EC $\mathcal{G}$ to that in the whole table, $\mathcal{P}$ in two steps: first, we transform $\mathcal{Q}$ to $\mathcal{Q}'$, the $\mathcal{SA}$ distribution of an EC $\mathcal{G}'$ that follows the proportionality requirement; then, we transform $\mathcal{Q}'$ to $\mathcal{P}$. Lemma 1 proves that $\mathcal{Q}'$ exists and gives the upper bound cost of the transformation from $\mathcal{Q}$ to $\mathcal{Q}'$. Theorem 4 builds on Lemma 1 and specifies the conditions for EC sizes that satisfy t-closeness.

Lemma 1 *Let $\mathcal{G}$ be an EC from table $\mathcal{DB}$ with bucket partition φ, and $\mathcal{Q} = (q_1, q_2, \ldots, q_m)$ be the SA distribution in $\mathcal{G}$. Then there exists an SA distribution $\mathcal{Q}' = (q'_1, q'_2, \ldots, q'_m)$ of an EC following the proportionality requirement with respect to φ, such that the cost of transforming $\mathcal{Q}$ to $\mathcal{Q}'$ is upper bounded by $\mathcal{D} = EMD(d(\mathcal{G}, \varphi), d(\mathcal{DB}, \varphi))$.*

Proof Let $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ be the domain of SA. Consider the partition of $\mathcal{V}$, $\overline{\mathcal{V}} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_{|\varphi|}\}$, where $\mathcal{V}_i$ is the set of SA values in $\mathcal{B}_i \in \varphi$. Given $\overline{\mathcal{V}}$, we derive $\overline{\mathcal{Q}} = (\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_{|\varphi|})$ and $\overline{\mathcal{Q}'} = (\mathcal{Q}'_1, \mathcal{Q}'_2, \ldots, \mathcal{Q}'_{|\varphi|})$, where $q_y \in \mathcal{Q}_i (q'_y \in \mathcal{Q}'_i)$ if and only if $v_y \in \mathcal{V}_i$. Let x_i be the set of tuples from bucket $\mathcal{B}_i$ in $\mathcal{G}$. Then, $\sum_{\forall q_y \in \mathcal{Q}_i} q_y = \frac{|x_i|}{|\mathcal{G}|}$. The φ distribution in $\mathcal{G}$ is $d(\mathcal{G}, \varphi) = \left(\frac{|x_1|}{|\mathcal{G}|}, \frac{|x_2|}{|\mathcal{G}|}, \ldots, \frac{|x_{|\varphi|}|}{|\mathcal{G}|}\right)$, and that in $\mathcal{DB}$ is $d(\mathcal{DB}, \varphi) = \left(\frac{|\mathcal{B}_1|}{|\mathcal{DB}|}, \frac{|\mathcal{B}_2|}{|\mathcal{DB}|}, \ldots, \frac{|\mathcal{B}_{|\varphi|}|}{|\mathcal{DB}|}\right)$. We need to find a distribution $\mathcal{Q}' = (q'_1, q'_2, \ldots, q'_m)$ so that $\frac{|\mathcal{B}_j|}{|\mathcal{DB}|} = \sum_{\forall q'_z \in \mathcal{Q}'_j} q'_z$, $j = 1, 2, \ldots, |\varphi|$. Initially, we set $q'_z = 0$, $z = 1, 2, \ldots, m$. We see $d(\mathcal{DB}, \varphi)$ as a collection of holes to be filled by piles of earth transported from $d(\mathcal{G}, \varphi)$. Moreover, we see $\mathcal{Q}$ and $\mathcal{Q}'$ as piles of earth. The elements $\mathcal{Q}_i \in \overline{\mathcal{Q}}$ ($\mathcal{Q}'_i \in \overline{\mathcal{Q}'}$) can be seen as *clusters* of piles of earth from $\mathcal{Q}(\mathcal{Q}')$. During the earth transportation–from $d(\mathcal{G}, \varphi)$ to $d(\mathcal{DB}, \varphi)$–whenever δ earth is moved from $\frac{|x_i|}{|\mathcal{G}|}$ to $\frac{|\mathcal{B}_j|}{|\mathcal{DB}|}$ (costing by definition $d^u_{ij} \cdot \delta$), then we also move δ earth from (a pile in) the corresponding cluster $\mathcal{Q}_i$ to (a pile in) cluster $\mathcal{Q}'_j$. Then, after all holes in $d(\mathcal{DB}, \varphi)$ are filled with earth from $d(\mathcal{G}, \varphi)$, the volume of earth in cluster $\mathcal{Q}'_j$ is equal to the volume of hole $\frac{|\mathcal{B}_j|}{|\mathcal{DB}|}$, i.e. $\frac{|\mathcal{B}_j|}{|\mathcal{DB}|} = \sum_{\forall q'_z \in \mathcal{Q}'_j} q'_z$, $j = 1, 2, \ldots, |\varphi|$. In other words, $\mathcal{Q}$ is transformed to an SA distribution $\mathcal{Q}'$ that follows the proportionality requirement with respect to φ. Concerning the cost, we distinguish two cases during the above earth transportation operation:

Case 1: If $i \neq j$, then the cost of earth transportation from (a pile in) cluster $\mathcal{Q}_i$ to (a pile in) cluster $\mathcal{Q}'_j$ is *at most* $d^u_{ij} \cdot \delta$, since d^u_{ij} is, by its definition, the maximum ground distance between *any* pile $q_y \in \mathcal{Q}_i$ and *any* pile $q'_z \in \mathcal{Q}'_j$.

Case 2: If $i = j$, then, at the transportation from (a pile in) cluster $\mathcal{Q}_i$ to (a pile in) cluster $\mathcal{Q}'_i$, we simply require that earth from pile $q_y \in \mathcal{Q}_i$ only goes to the corresponding pile $q'_y \in \mathcal{Q}'_i$; the ground distance between these piles (q_y and q'_y) is 0, hence the cost of earth transportation is 0 too.

Based on the above two cases, we conclude that the cost of transforming $\mathcal{Q}$ to $\mathcal{Q}'$ is upper-bounded by $\mathcal{D}$. □

Example 6 We return to Example 4. Assume that $t = 0.45$, so that Fig. 3 shows the final bucketization tree. $\mathcal{B}_1$ relates to *Respiratory diseases* with $|\mathcal{B}_1| = 10$. $\mathcal{B}_2$ relates to *Digestive diseases*, with $|\mathcal{B}_2| = 8$. $\varphi = \{\mathcal{B}_1, \mathcal{B}_2\}$ is a bucket partition of $\mathcal{DB}$ with $d^u_{12} = 1$. Suppose that $\mathcal{G}$ contains 3 tuples with *SARS* from $\mathcal{B}_1$ and 2 tuples with *Gastric flu* from $\mathcal{B}_2$. The disease distribution in $\mathcal{G}$ then is $\mathcal{Q} = (q_1, \ldots, q_6) = \left(\frac{3}{5}, 0, 0, \frac{2}{5}, 0, 0\right)$. Initially, $\mathcal{Q}' = (q'_1, \ldots, q'_6) = (0, \ldots, 0)$. Thus, $\overline{\mathcal{Q}} = (\mathcal{Q}_1, \mathcal{Q}_2) = (\{q_1, q_2, q_3\}, \{q_4, q_5, q_6\}) = \left(\left\{\frac{3}{5}, 0, 0\right\}, \left\{\frac{2}{5}, 0, 0, \right\}\right)$, and $\overline{\mathcal{Q}'} = (\mathcal{Q}'_1, \mathcal{Q}'_2) = (\{q'_1, q'_2, q'_3\}, \{q'_4, q'_5, q'_6\}) = (\{0, 0, 0\}, \{0, 0, 0\})$. We have $d(\mathcal{DB}, \varphi) = \left(\frac{|\mathcal{B}_1|}{|\mathcal{DB}|}, \frac{|\mathcal{B}_2|}{|\mathcal{DB}|}\right) = \left(\frac{5}{9}, \frac{4}{9}\right)$, and $d(\mathcal{G}, \varphi) = \left(\frac{|x_1|}{|\mathcal{G}|}, \frac{|x_2|}{|\mathcal{G}|}\right) = \left(\frac{3}{5}, \frac{2}{5}\right)$. To transform $d(\mathcal{G}, \varphi)$ to $d(\mathcal{DB}, \varphi)$, we move $\frac{5}{9}$ earth from $\frac{|x_1|}{|\mathcal{G}|}$ to $\frac{|\mathcal{B}_1|}{|\mathcal{DB}|}$ (at cost 0), $\frac{2}{5}$ earth from $\frac{|x_2|}{|\mathcal{G}|}$ to $\frac{|\mathcal{B}_2|}{|\mathcal{DB}|}$ (cost 0), and $\frac{3}{5} - \frac{5}{9} = \frac{2}{45}$ from $\frac{|x_1|}{|\mathcal{G}|}$ to $\frac{|\mathcal{B}_2|}{|\mathcal{DB}|}$ (at cost $\frac{2}{45}$). Thus, $EMD(d(\mathcal{G}, \varphi), d(\mathcal{DB}, \varphi)) = \frac{2}{45}$. Accordingly, we also move $\frac{5}{9}$ earth from $\mathcal{Q}_1$ to $\mathcal{Q}'_1$ (i.e., from q_1 to q'_1, at cost 0), and $\frac{2}{5}$ earth from $\mathcal{Q}_2$ to $\mathcal{Q}'_2$ (i.e., from q_4 to q'_4), hence $\overline{\mathcal{Q}'} = \left(\left\{\frac{5}{9}, 0, 0\right\}, \left\{\frac{2}{5}, 0, 0\right\}\right)$ (at cost 0). When moving $\frac{2}{45}$ earth from $\mathcal{Q}_1$ to $\mathcal{Q}'_2$, there are multiple choices, we can move $\frac{2}{45}$ earth from q_1 to q'_4, q'_5, or q'_6 (at cost $\frac{2}{45}$). Assume that it is moved from q_1 to q'_4, then $\overline{\mathcal{Q}'} = \left(\left\{\frac{5}{9}, 0, 0\right\}, \left\{\frac{4}{9}, 0, 0\right\}\right)$, and $\frac{|\mathcal{B}_j|}{|\mathcal{DB}|} = \sum_{\forall q'_z \in \mathcal{Q}'_j} q'_z$, $j = 1, 2$. The resultant SA distribution $\mathcal{Q}' = \left(\frac{5}{9}, 0, 0, \frac{4}{9}, 0, 0\right)$ follows the proportionality requirement with respect to φ. Alternatively, if $\frac{2}{45}$ earth is moved to q'_5, then $\mathcal{Q}' = \left(\frac{5}{9}, 0, 0, \frac{2}{5}, \frac{2}{45}, 0\right)$. If the earth is moved to q'_6, then $\mathcal{Q}' = \left(\frac{5}{9}, 0, 0, \frac{2}{5}, 0, \frac{2}{45}\right)$. In all three cases, the cost of transforming $\mathcal{Q}$ to $\mathcal{Q}'$ is $\frac{2}{45} \leq EMD(d(\mathcal{G}, \varphi), d(\mathcal{DB}, \varphi))$.

Theorem 4 (Triangle inequality) *Let $\mathcal{G}$ be an EC from table $\mathcal{DB}$ with bucket partition φ. If $\mathcal{D} + \mathcal{U} \leq t$, where $\mathcal{D} = EMD(d(\mathcal{G}, \varphi), d(\mathcal{DB}, \varphi))$ and $\mathcal{U} = \sum_{\forall \mathcal{B} \in \varphi} CET^U_{\mathcal{B}}$, then the SA distribution $\mathcal{Q}$ in $\mathcal{G}$ is t-close to the SA distribution $\mathcal{P}$ in $\mathcal{DB}$.*

Proof We transform $\mathcal{Q}$ to $\mathcal{P}$ in two steps. First, we transform $\mathcal{Q}$ to $\mathcal{Q}'$, a distribution that follows the proportionality requirement with respect to φ; then, we transform $\mathcal{Q}'$ to $\mathcal{P}$. By Lemma 1, the cost of transforming $\mathcal{Q}$ to $\mathcal{Q}'$ is upper-bounded by $\mathcal{D}$. Furthermore, by Theorem 3, the cost of transforming $\mathcal{Q}'$ to $\mathcal{P}$ is upper bounded by $\mathcal{U}$. Therefore, the *EMD* of transforming $\mathcal{Q}$ to $\mathcal{P}$ via $\mathcal{Q}'$ is upper bounded by $\mathcal{D} + \mathcal{U}$. Thus, if $\mathcal{D} + \mathcal{U} \leq t$, the *EMD* between $\mathcal{Q}$ and $\mathcal{P}$ is at most t. □

We now consider the process of dynamically determining the size of an EC, or deciding how many tuples to take out from each bucket to form an EC. First, we consider all tuples of $\mathcal{DB}$ (i.e., all the buckets in φ) as a single EC, r. Then we split r into two ECs by dichotomizing $\mathcal{B}_i$ into $\mathcal{B}^1_i$ and $\mathcal{B}^2_i$, where $i = 1, 2 \ldots, |\varphi|$. $\mathcal{B}^1_i$ and $\mathcal{B}^2_i$ have *approximately* the same size. The left child c_1 of r is composed of $\mathcal{B}^1_i$, and the right child c_2 of r is composed of $\mathcal{B}^2_i$, where $i = 1, 2, \ldots, |\varphi|$. Let $d(c_1, \varphi)$ and $d(c_2, \varphi)$ be the φ distributions in c_1 and c_2 respectively. By Theorem 4, the split is allowed only if both $EMD(d(c_1, \varphi), d(\mathcal{DB}, \varphi)) + \mathcal{U} \leq t$ and $EMD(d(c_2, \varphi), d(\mathcal{DB}, \varphi)) + \mathcal{U} \leq t$ hold. After r is split, we recursively split c_1 and c_2 in the same way. We illustrate this process with an example.

Example 7 Re-consider Example 6, with $t = 0.45$, $\varphi = \{\mathcal{B}_1, \mathcal{B}_2\}$, and $d^u_{12} = 1$. If we strictly follow the proportionality requirement, then there are at most two ECs, each having 5 tuples from $\mathcal{B}_1$ and 4 tuples from $\mathcal{B}_2$. However, we can generate more ECs by dynamically determining their size as follows. By Theorem 3, we have $\mathcal{U} = CET^U_{\mathcal{B}_1} + CET^U_{\mathcal{B}_2} = \frac{2}{9} + \frac{1}{6} = \frac{7}{18}$ (see Example 3). The notation $r = [10, 8]$ in Fig. 6 means that r contains 10 tuples from $\mathcal{B}_1$ and 8 tuples from $\mathcal{B}_2$ (i.e., all tuples in $\mathcal{DB}$). We dichotomize r into $c_1 = [5, 4]$ and $c_2 = [5, 4]$. Since both c_1 and c_2 follow the proportionality requirement with respect to φ, and $\mathcal{U} = \frac{7}{18} < t$, the split is allowed. We proceed to split c_1 into $g_1 = [3, 2]$ and $g_2 = [2, 2]$. Now $d(\mathcal{DB}, \varphi) = \left(\frac{5}{9}, \frac{4}{9}\right)$, $d(g_1, \varphi) = \left(\frac{3}{5}, \frac{2}{5}\right)$, and $d(g_2, \varphi) = \left(\frac{1}{2}, \frac{1}{2}\right)$. Since $EMD(d(g_1, \varphi), d(\mathcal{DB}, \varphi)) + \mathcal{U} = \frac{2}{45} + \frac{7}{18} < 0.45$ and $EMD(d(g_2, \varphi), d(\mathcal{DB}, \varphi)) + \mathcal{U} = \frac{1}{18} + \frac{7}{18} < 0.45$, splitting c_1 is allowed by Theorem 4. If we further dichotomize g_1 into $gg_1 = [2, 1]$ and $gg_2 = [1, 1]$, then $EMD(d(gg_1, \varphi), d(\mathcal{DB}, \varphi)) + \mathcal{U} = \frac{1}{9} + \frac{7}{18} > 0.45$.

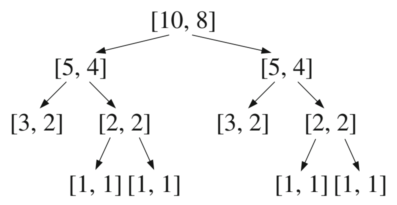

Fig. 6 Example of dynamically determining EC size

Thus, splitting g_1 is not allowed. Still, further splitting g_2 into two ECs each having one tuple from $\mathcal{B}_1$ and one tuple from $\mathcal{B}_2$ is allowed. The process of splitting c_2 is similar to c_1. The recursive splitting process generates the tree shown in Fig. 6. Each leaf node represents the size of a possible EC.

The redistribution phase of SABRE uses the binary tree as illustrated in Fig. 6 to effectively split ECs. The same tree structure has also been employed in Mondrian [15]. However, the trees for the two methods are generated in a very different manner due to the distinct requirements of their underlying privacy models. The Mondrian k-anonymi-zation algorithm, under local recoding, splits a node in the binary tree once it accommodates at least $2k$ tuples. Instead, the redistribution phase of SABRE, tailored for t-closeness, allows such a splitting, only if the resultant ECs strictly satisfy the conditions specified in Theorem 4. Still, the redistribution phase only determines the size of each possible EC (i.e., the number of tuples from each bucket to compose it). How real tuples are retrieved from each bucket to create an EC is discussed in Sect. 4.4.

Function ECSize (φ)

```
1 Let $\varphi = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{|\varphi|}\}$ be the set of buckets generated from
  the bucketization phase;
2 Let $a$ be an array, and $a_i$ be the $i$th element of $a$;
3 Create a tree $ECT$ and its root $r$;
4 Initialize $r._a$ by $r._{a_i} = |\mathcal{B}_i|, i = 1, 2, \ldots, |\varphi|$;
5 dichotomize$(r, \varphi)$;
6 Let $S_a$ be a set of arrays initialized to be empty;
7 Traverse $ECT$, and for each leaf node $n$ add $n._a$ to $S_a$;
8 Return $S_a$;
```

Function ECSize describes our algorithm that determines the number of tuples to be taken out from $\mathcal{B}_i$, $i = 1, 2, \ldots, |\varphi|$. Parameter φ is the set of buckets generated from the bucketization phase (step 1, see Sect. 4.2). Each array a represents an EC, and its ith element a_i is the number of tuples from $\mathcal{B}_i$ when generating the EC (step 2). ECT is a tree with each node associated with an array a. The root of ECT is r, and $r._a$ represents an EC composed of the whole table (steps 3–4). We call function dichotomize to generate ECT (step 5). In the final ECT, each leaf node represents an EC that cannot be further split. Eventually, each leaf node n in ECT is scanned, and its $n._a$ is stored in S_a (steps 7–8).

Procedure dichotomize (n,φ)

```
1 if $n._{a_i} < 2$, for all $i \in \{1, \ldots, |\varphi|\}$ then
2     Return;
3 foreach $i \in \{1, \ldots, |\varphi|\}$ do
4     $a_i^1 = round(0.5 \times n._{a_i})$;
5     $a_i^2 = n._{a_i} - a_i^1$;
6 Set $c_1._a$ by $c_1._{a_i} = a_i^1, i = 1, \ldots, |\varphi|$;
7 $\mathcal{D}_1 = EMD(d(c_1, \varphi), d(\mathcal{DB}, \varphi))$;
8 Set $c_2._a$ by $c_2._{a_i} = a_i^2, i = 1, \ldots, |\varphi|$;
9 $\mathcal{D}_2 = EMD(d(c_2, \varphi), d(\mathcal{DB}, \varphi))$;
10 if $\mathcal{D}_1 + U \leq t$ and $\mathcal{D}_2 + U \leq t$ then
11     Set $c_1$ and $c_2$ to be the left and right child of $n$ respectively;
12     dichotomize$(c_1, \varphi)$;
13     dichotomize$(c_2, \varphi)$;
```

Procedure dichotomize splits ECT recursively. Parameter n is a node in ECT, and φ is the computed bucket partition (see Sect. 4.2) of table $\mathcal{DB}$. Steps 3–5 split the EC denoted by node n into two, whose sizes are determined by a_i^1 and a_i^2 respectively, $i = 1, 2, \ldots, |\varphi|$. If both c_1 and c_2 follow Theorem 4 (steps 6–10), then n will acquire c_1 and c_2 as its children (step 11). Steps 12–13 recursively examines whether the two newly generated children can be split. The process terminates if splitting a node cannot generate two smaller ECs (the evaluation of step 1 is true) or the evaluation of step 10 is false. When none of the nodes in ECT can be split, ECT is fully generated.

Now our presentation of the two phases is complete. In the following, we put the bucketization phase (Sect. 4.2) and the redistribution phase (Sect. 4.3) together and summarize the framework of SABRE.

4.4 SABRE and its two instantiations

Algorithm SABRE provides a high level description of our framework. $\{v_1, v_2, \ldots, v_m\}$ is the domain of the sensitive attribute. The global SA distribution in the whole table $\mathcal{DB}$ is $(p_1, p_2, \ldots, p_m)$, where $p_i = \frac{\mathcal{N}_i}{|\mathcal{DB}|}$ and $\mathcal{N}_i$ is the number of tuples in $\mathcal{DB}$ with SA value of v_i (step 1). Steps 2–8 deal with the *bucketization phase*. Function bucketCat computes the bucketization tree for categorical sensitive attribute (step 5). If the SA is numerical, the tree is generated by function bucketNum (step 8). In the final bucketization tree, each leaf node is associated to a bucket, and all the tuples with the SA values included in the leaf node will be pushed to the corresponding bucket. Thus, a bucket partition of $\mathcal{DB}$, φ, is formed. Step 9 is the *redistribution phase*; function ECSize dynamically determines the size of each possible EC. Steps 10–15 form all possible ECs and output them. S_a, returned by ECSize, is a list of arrays. Each array represents an EC. Given an array a (step 10), SABRE takes a_i tuples from bucket $\mathcal{B}_i \in \varphi$, where a_i is the ith element of a and $i = 1, 2, \ldots, |\varphi|$ (step 13). Then SABRE forms an EC $\mathcal{G}$ out of them (step 14). Each generated EC is output (step 15).

Algorithm: SABRE ($\mathcal{DB}, \mathcal{SA}, t$)

```
1  Let $\{v_1, v_2, \ldots, v_m\}$ be all the $\mathcal{SA}$ values in $\mathcal{DB}$, and
   $\{p_1, p_2, \ldots, p_m\}$ be their distributions;
2  Let $\mathcal{VP}$ be the list of $(v_i, p_i)$, $i = 1, 2, \ldots, m$;
3  if $\mathcal{SA}$ is categorical then
4      Let $\mathcal{H}$ be the domain hierarchy of $\mathcal{SA}$;
5      $\varphi$ = bucketCat($\mathcal{H}, \mathcal{VP}$);
6  else
7      Sort $\mathcal{VP}$ in the ascending order of $\mathcal{SA}$ values;
8      $\varphi$ = bucketNum($\mathcal{VP}$);
9  $S_a$ = ECSize($\varphi$);
10 foreach array $a$ in $S_a$ do
11     Create an empty EC, say $\mathcal{G}$;
12     foreach $a_i$, $i$th element of $a$ do
13         $ec_i$ = takeOut($\mathcal{B}_i, a_i$);
14         add $ec_i$ to $\mathcal{G}$;
15     output $\mathcal{G}$;
```

When taking out tuples from a bucket, SABRE does not distinguish their SA values. The t-closeness between the EC and the whole table is anyway guaranteed by Theorems 3 and 4. Which tuples to pick is to be determined by information loss considerations. As discussed, we adopt the General Loss Metric (GLM) (see Sect. 3.2), as we assume that the anonymized data are for multiple, not known a priori, uses. GLM requires the minimum bound boxes of ECs to be as small as possible. We achieve this by greedily picking tuples of similar QI values, to the extent that is possible.

We provide two instantiations of SABRE: SABRE-KNN and SABRE-AK. They differ only in function takeOut ($\mathcal{B}_i, a_i$) (step 13 of Algorithm SABRE), which determines the tuples that should be picked from each bucket. Both schemes utilize the notion of *nearest neighbors*. SABRE-KNN finds the *exact* neighbors, whereas SABRE-AK uses *approximate* neighbors.

We define a multidimensional space with each of the QI attributes serving as a dimension. The axis for a dimension defined by a numerical QI attribute is straightforward. For a categorical QI, the ordering of all leaves by a pre-order traversal of its domain hierarchy forms the axis. Thus, each tuple is represented as a point in this space. We use the Euclidean distance to measure the distance between two points.

SABRE-KNN selects a_i tuples from bucket $\mathcal{B}_i$ based on a *kNN* search. First, it forms an empty EC $\mathcal{G}$. Then it selects a random tuple x from a randomly selected bucket $\mathcal{B} \in \varphi$. Finally, in each bucket $\mathcal{B}_i, i = 1, 2, \ldots, |\varphi|$, it finds the nearest a_i neighbors of x and adds them into $\mathcal{G}$. x and all its selected nearest neighbors are deleted from their original buckets. Thus, to form an EC, all tuples in the buckets need to be scanned once. The time cost of this operation is $O(|S_\mathcal{G}| \cdot |\mathcal{DB}|)$, where $|S_\mathcal{G}|$ is the number of ECs and $|\mathcal{DB}|$ is the size of the dataset.

As an alternative to the computationally more demanding SABRE-KNN, we also suggest a more efficient scheme, SABRE-AK, that looks for approximate nearest neighbors of x. This is facilitated by the *Hilbert space-filling curve* [22], a continuous fractal that maps regions of the multidimensional QI space to one-dimensional Hilbert values. Each tuple has a Hilbert value corresponding to the region that contains it. If two tuples are close in the multidimensional space, their Hilbert values are also close with high probability. SABRE-AK first sorts all tuples in each bucket in ascending order of their Hilbert values. Then, when looking for x's a_i nearest neighbors in bucket $\mathcal{B}_i$, it selects the a_i tuples that are closest to x in terms of their Hilbert values. In practice, we use binary search to find in $\mathcal{B}_i$ a tuple $\bar{x}$, whose Hilbert value is closest to that of x. Then we check the neighbors of $\bar{x}$ and select the closest a_i ones (including $\bar{x}$) to x. The average time cost of this search is $O\left(|S_\mathcal{G}| \cdot \left(\log \frac{|\mathcal{DB}|}{|\varphi|} + \frac{|\mathcal{DB}|}{|S_\mathcal{G}| \cdot |\varphi|}\right) \cdot |\varphi|\right)$, where $|\varphi|$ is the number of buckets, $\frac{|\mathcal{DB}|}{|\varphi|}$ the average size of a bucket, and $\frac{|\mathcal{DB}|}{|S_\mathcal{G}| \cdot |\varphi|}$ the average number of tuples taken out from a bucket to form an EC. Since $\left(\log \frac{|\mathcal{DB}|}{|\varphi|} + \frac{|\mathcal{DB}|}{|S_\mathcal{G}| \cdot |\varphi|}\right) \cdot |\varphi|) << |\mathcal{DB}|$, we expect SABRE-AK to be more efficient than SABRE-KNN.

5 SABRE for streaming data

In this section, we extend SABRE to the context of streaming data. Typically, data streams are unbounded, so t-closeness should be restricted to a window.

Definition 6 ((ω,t)-*closeness*) Let $\mathcal{S}_{in}$ be an input stream, SA its sensitive attribute, and $\mathcal{S}_{out}$ the output stream generated from $\mathcal{S}_{in}$. We monitor $\mathcal{S}_{in}$ on a window of size ω. We say that $\mathcal{S}_{out}$ follows the (ω, t)-closeness, if and only if the following conditions hold:

- For each tuple $x \in \mathcal{S}_{in}$, there exists in $\mathcal{S}_{out}$ an equivalence class, which contains the corresponding anonymized tuple of x.
- For each equivalence class $\mathcal{G} \in \mathcal{S}_{out}$, there exists a window $\mathcal{W}$ in $\mathcal{S}_{in}$, such that $\mathcal{G}$ is generated from the tuples in $\mathcal{W}$, and the SA distribution in $\mathcal{G}$ does not differ from that of all tuples in $\mathcal{W}$ by more than a given threshold t.

Definition 7 (*Expiring constraint*) Assume an (ω,t)-closeness scheme T, which takes as input a data stream $\mathcal{S}_{in}$ and generates an output data stream $\mathcal{S}_{out}$. Let $\mathcal{I}$ be a positive integer, and $\omega = j \cdot \mathcal{I}$, where $j > 0$ is an integer. Assume that each window in $\mathcal{S}_{in}$ advances by $\mathcal{I}$ tuples. T satisfies the

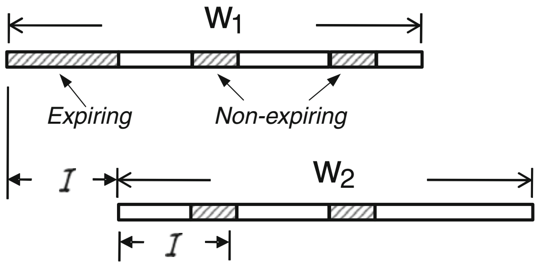

Fig. 7 Windows and their advance

expiring constraint if and only if, each time a window in $\mathcal{S}_{in}$ advances its first $\mathcal{I}$ tuples expire and are output.

As the window slides, a sequence of windows $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_i, \mathcal{W}_{i+1}, \ldots$ are generated. When window $\mathcal{W}_i$ advances by $\mathcal{I}$ tuples, a new window $\mathcal{W}_{i+1}$ is generated. The first (oldest) $\mathcal{I}$ tuples in $\mathcal{W}_i$, which are left outside $\mathcal{W}_{i+1}$, should be output. We call these *expiring* tuples. Thus, in Fig. 7, as the window slides from $\mathcal{W}_1$ to $\mathcal{W}_2$, the tuples in $\mathcal{W}_1 \setminus \mathcal{W}_2$ (the first $\mathcal{I}$ tuples in $\mathcal{W}_1$), are expiring and should be output.

Assume that $\mathcal{S}_{in}$ is an input (original) stream, and an (ω, t)-closeness scheme is applied on it. As tuples arrive from $\mathcal{S}_{in}$, they are first buffered in memory. When the window advances from $\mathcal{W}_i$ to $\mathcal{W}_{i+1}$, we create ECs for the expiring tuples in $\mathcal{W}_i$. Still, the expiring tuples on their own may not satisfy t-closeness to the SA distribution of all tuples in $\mathcal{W}_i$; thus, the created ECs may contain both expiring and non-expiring tuples. As an example, let $\mathcal{W}_i$ be the advancing window and take $\mathcal{I} = 1$. With high probability, the single expiring tuple (the first tuple in $\mathcal{W}_i$) is not t-close to the distribution of all the tuples in $\mathcal{W}_i$. Therefore, we need to accompany the single expiring tuple with some non-expiring tuple(s) in $\mathcal{W}_i$ to form an EC. After the ECs for the expiring tuples are generated, we output all these ECs.

As the window advances continuously, an output stream $\mathcal{S}_{out}$ is generated. Given that the output ECs may contain non-expired tuples, when the window slides to $\mathcal{W}_{i+1}$, some tuples in $\mathcal{W}_{i+1}$ may have already been output. For instance, in Fig. 7, as the window slides from $\mathcal{W}_1$ to $\mathcal{W}_2$, the first $\mathcal{I}$ tuples in $\mathcal{W}_1$ are expiring. ECs are created for the expiring tuples, and contain both the expiring tuples and non-expiring ones; thus, ECs are composed of the tuples in the shaded segments in $\mathcal{W}_1$. These ECs are output. Thus, by the time the window assumes the position $\mathcal{W}_2$, some of its tuples have already been output.

Based on the above distinctions, we define the following classes of tuples, illustrated in Fig. 8:

1. $\mathcal{R}(\mathcal{W})$, all tuples in window $\mathcal{W}$;
2. $\mathcal{R}(\mathcal{W}, o)$, the already output tuples from $\mathcal{W}$;
3. $\mathcal{R}(\mathcal{W}, \bar{o})$, the not-yet-output tuples in $\mathcal{W}$, $\mathcal{R}(\mathcal{W}, o) \cup \mathcal{R}(\mathcal{W}, \bar{o}) = \mathcal{R}(\mathcal{W})$;
4. $\mathcal{R}(\mathcal{W}, e)$, the expiring tuples of $\mathcal{W}$ when it advances (i.e., the first $\mathcal{I}$ tuples of $\mathcal{W}$), and
5. $\mathcal{R}(\mathcal{W}, e\bar{o})$, the expiring and not-yet-output tuples of $\mathcal{W}$.

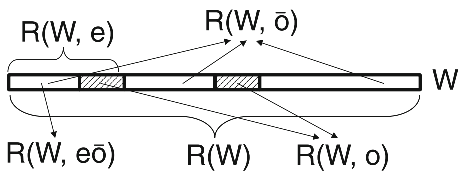

Fig. 8 Tuples classification

Table 4 Streaming notations

Notation	Denotation
$\mathcal{S}_{in}$	The input data stream
$\mathcal{S}_{out}$	The output data stream generated from $\mathcal{S}_{in}$
$\mathcal{W}$	A window in $\mathcal{S}_{in}$
$\mathcal{I}$	Time interval of window advance
ω	Window size; $\omega = j \cdot \mathcal{I}$, where $j > 0$ is an integer
$\mathcal{R}(\mathcal{W})$	All the tuples in $\mathcal{W}$
$\mathcal{R}(\mathcal{W}, o)$	All the already output tuples of $\mathcal{W}$
$\mathcal{R}(\mathcal{W}, \bar{o})$	All the not-yet-output tuples in $\mathcal{W}$, $\mathcal{R}(\mathcal{W}) = \mathcal{R}(\mathcal{W}, o) \cup \mathcal{R}(\mathcal{W}, \bar{o})$
$\mathcal{R}(\mathcal{W}, e\bar{o})$	Expiring and not-yet-output tuples in $\mathcal{W}$
$\mathcal{SA}(\mathcal{W})$	$\mathcal{SA}$ distribution in $\mathcal{R}(\mathcal{W})$
$\mathcal{SA}(\mathcal{W}, \bar{o})$	$\mathcal{SA}$ distribution in $\mathcal{R}(\mathcal{W}, \bar{o})$
$\mathcal{D}_{\mathcal{W}\bar{o}}$	$EMD(\mathcal{SA}(\mathcal{W}), \mathcal{SA}(\mathcal{W}, \bar{o}))$

Table 4 lists some notations that we use in this section.

In the following we briefly present Algorithm SABRE$_\mathsf{W}$, which extends SABRE to the context of streaming data. We first give some intuition of the algorithm by an example.

Example 8 Assume that $\mathcal{S}_{in}$ is the original input stream of SABRE$_\mathsf{W}$. Figure 9 illustrates the sequence of windows, as the window advances in $\mathcal{S}_{in}$. When $\mathcal{W}_1$ is sliding at instant ι_1, $\mathcal{R}(\mathcal{W}_1) = \mathcal{R}(\mathcal{W}_1, \bar{o})$; i.e, none of its tuples is yet output. We partition tuples in $\mathcal{W}_1$ into S^1_G, a list of ECs, so that the SA distribution of each EC in S^1_G is t-close to that of all the tuples in $\mathcal{W}_1$. ECs that contain expiring tuples are output and deleted from S^1_G. After that $\mathcal{W}_1$ slides to $\mathcal{W}_2$. When $\mathcal{W}_2$ begins to advance at instant ι_2, assume that $\mathcal{R}(\mathcal{W}_2, \bar{o})$ and $\mathcal{R}(\mathcal{W}_2)$ have similar SA distributions (i.e., the SA distribution of not-yet-output tuples in $\mathcal{W}_2$ is similar to that of all tuples in $\mathcal{W}_2$). Like what we did for $\mathcal{W}_1$, we partition $\mathcal{R}(\mathcal{W}_2, \bar{o})$ into a set of ECs, S^2_G (more in *Case 2* of SABRE$_\mathsf{W}$ below). All the ECs with expiring tuples are output and removed from S^2_G. Then, $\mathcal{W}_2$ slides to $\mathcal{W}_3$. When $\mathcal{W}_3$ slides at instant ι_3, assume that $\mathcal{R}(\mathcal{W}_3, e\bar{o})$ is empty (i.e., all expiring tuples have already

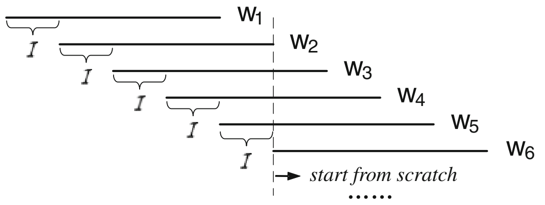

Fig. 9 An example for Algorithm $\mathsf{SABRE_W}$

been output). Under this circumstance, we do not have to anonymize any tuple, and $\mathcal{W}_3$ simply slides to $\mathcal{W}_4$ (see *Case 1* of $\mathsf{SABRE_W}$ below). When $\mathcal{W}_4$ advances at instant ι_4, assume that the SA distribution of $\mathcal{R}(\mathcal{W}_4, \bar{o})$ is very 'different' from that of $\mathcal{R}(\mathcal{W}_4)$. If we partition $\mathcal{R}(\mathcal{W}_4, \bar{o})$ into a set of ECs, some of them (containing expiring tuples) may also be very 'different' from $\mathcal{R}(\mathcal{W}_4)$ with respect to SA distribution, and (ω, t)-closeness cannot be guaranteed any more. However, $\mathcal{R}(\mathcal{W}_4, e\bar{o})$ (i.e., the expiring and not-yet-output tuples in $\mathcal{W}_4$) falls in $\mathcal{W}_2$, i.e. $\mathcal{R}(\mathcal{W}_4, e\bar{o}) \subset \mathcal{R}(\mathcal{W}_2, \bar{o})$.[3] Since $\mathcal{R}(\mathcal{W}_2, \bar{o})$ is already partitioned into $S^2_{\mathcal{G}}$, each tuple in $\mathcal{R}(\mathcal{W}_4, e\bar{o})$ must belong to an EC of $S^2_{\mathcal{G}}$ (see *Case 3* of $\mathsf{SABRE_W}$ below). Thus, we output all the ECs in $S^2_{\mathcal{G}}$ and delete them. Hence, we ensure that all tuples in $\mathcal{R}(\mathcal{W}_4, e\bar{o})$ are output, while now all the tuples in $\mathcal{W}_2$ have also been output. Then, as $\mathcal{W}_5$ slides at instant ι_5, all its expiring tuples have already been output because they fall in $\mathcal{W}_2$. So $\mathcal{W}_5$ simply slides to $\mathcal{W}_6$. When $\mathcal{W}_6$ enters, none of its tuples is output, and the anonymization of $\mathsf{SABRE_W}$ starts from scratch.

Algorithm $\mathsf{SABRE_W}$ anonymizes an input stream $\mathcal{S}_{in}$ into an output stream $\mathcal{S}_{out}$ to attain the *(ω,t)-closeness* and *expiring constraint*. $\mathsf{SABRE_W}$ is based on Algorithm SABRE. Window $\mathcal{W}$ buffers the first ω tuples from input stream $\mathcal{S}_{in}$ (steps 2–3). When $\mathcal{I}$ new tuples have arrived from $\mathcal{S}_{in}$ (step 5), the first $\mathcal{I}$ tuples in $\mathcal{W}$ expire and need to be output. $\mathcal{R}(\mathcal{W}, e\bar{o})$ is the set of expiring and not-yet-output tuples in $\mathcal{W}$ at the moment of its advance. We distinguish the following cases.

- *Case 1* (*steps 7–9*): $|\mathcal{R}(\mathcal{W}, e\bar{o})| = 0$. All expiring tuples in $\mathcal{W}$ have already been output (when a window advanced in a previous iteration), so we do not need to anonymize any tuple. $\mathcal{W}$ simply slides by $\mathcal{I}$ tuples (step 8) and $\mathsf{SABRE_W}$ goes back to step 5 (step 9).
- *Case 2* (*steps 14–19*): $|\mathcal{R}(\mathcal{W}, e\bar{o})| > 0$ and $\mathcal{D}_{\mathcal{W}\bar{o}} < t$. In this case, the not-yet-output tuples in $\mathcal{W}$ and all the tuples in $\mathcal{W}$ have similar $\mathcal{SA}$ distributions. We partition $\mathcal{R}(\mathcal{W}, \bar{o})$, the not-yet-output tuples in $\mathcal{W}$, into a set of ECs and output those containing expiring tuples. Algorithm SABRE is called and $\mathcal{R}(\mathcal{W}, \bar{o})$ is anonymized into $S_{\mathcal{G}}$, a set of ECs (step 15). For each equivalence class $\mathcal{G} \in S_{\mathcal{G}}$, $\mathrm{EMD}(\mathcal{SA}(\mathcal{G}), \mathcal{SA}(\mathcal{W}, \bar{o})) \leq t - \mathcal{D}_{\mathcal{W}\bar{o}}$, where $\mathcal{SA}(\mathcal{G})$ is the SA distribution in $\mathcal{G}$. Since $EMD(\mathcal{SA}(\mathcal{G}), \mathcal{SA}(\mathcal{W}, \bar{o})) + EMD(\mathcal{SA}(\mathcal{W}, \bar{o}), \mathcal{SA}(\mathcal{W})) \leq t - \mathcal{D}_{\mathcal{W}\bar{o}} + \mathcal{D}_{\mathcal{W}\bar{o}} = t$, i.e. the *EMD* of transforming $\mathcal{SA}(\mathcal{G})$ to $\mathcal{SA}(\mathcal{W})$ via $\mathcal{SA}(\mathcal{W}, \bar{o})$ is at most t, we conclude that $EMD(\mathcal{SA}(\mathcal{G}), \mathcal{SA}(\mathcal{W})) \leq t$ (the proof is similar to that of Theorem 4). Therefore, $\mathcal{G}$ is t-close to $\mathcal{R}(\mathcal{W})$. Then, ECs that contain expiring tuples are output and deleted from $S_{\mathcal{G}}$ (steps 16–19). We denote the procedure of Case 2 by DIRECT, which means that ECs for $\mathcal{R}(\mathcal{W}, e\bar{o})$ are obtained *directly* from a partition of not-yet-output tuples in $\mathcal{W}$.
- *Case 3* (*steps 21–24*): $|\mathcal{R}(\mathcal{W}, e\bar{o})| > 0$ and $\mathcal{D}_{\mathcal{W}\bar{o}} \geq t$. In this case, the SA distribution of $\mathcal{R}(\mathcal{W}, \bar{o})$ and that of $\mathcal{R}(\mathcal{W})$ are very 'different'. If we partition $\mathcal{R}(\mathcal{W}, \bar{o})$ into a set of ECs (as *Case 2*), some ECs containing expiring tuples may also be very 'different' from $\mathcal{R}(\mathcal{W})$ in terms of SA distribution, and *(ω,t)*-closeness may not be guaranteed. However, each tuple of $\mathcal{R}(\mathcal{W}, e\bar{o})$ is contained in one EC of $S_{\mathcal{G}}$ (see Theorem 5 below for a formal proof). To anonymize and output $\mathcal{R}(\mathcal{W}, e\bar{o})$ we output all the ECs in $S_{\mathcal{G}}$ and clear it (steps 22–24). We denote the procedure of *Case* 3 by $\mathsf{INDIRECT}$, which means that ECs for $\mathcal{R}(\mathcal{W}, e\bar{o})$ are *not* obtained directly from a partition of not-yet-output tuples in $\mathcal{W}$.

Algorithm: $\mathsf{SABRE_W}(\mathcal{S}_{in}, t, \omega, \mathcal{I})$

```
1  Let $\mathcal{S}_{in}$ and $\mathcal{S}_{out}$ be the input and output streams respectively;
2  Let $\mathcal{W}$ be a window in $\mathcal{S}_{in}$;
3  $\mathcal{W}$ buffers the first $\omega$ tuples from $\mathcal{S}_{in}$;
4  Let $S_{\mathcal{G}}$ be a set of ECs, initialized to be empty;
5  while $\mathcal{I}$ new tuples have arrived from $\mathcal{S}_{in}$ do
6  Case 1:
7      if $|\mathcal{R}(\mathcal{W}, e\bar{o})| = 0$ then
8          Update $\mathcal{W}$ by advancing $\mathcal{I}$ tuples;
9          Continue;
10     Let $\mathcal{SA}(\mathcal{W}, \bar{o})$ be the $\mathcal{SA}$ distribution of $\mathcal{R}(\mathcal{W}, \bar{o})$;
11     Let $\mathcal{SA}(\mathcal{W})$ be the $\mathcal{SA}$ distribution of $\mathcal{R}(\mathcal{W})$;
12     Let $\mathcal{D}_{\mathcal{W}\bar{o}} = EMD(\mathcal{SA}(\mathcal{W}, \bar{o}), \mathcal{SA}(\mathcal{W}))$;
13 Case 2:
14     if $\mathcal{D}_{\mathcal{W}\bar{o}} < t$ then                  /* DIRECT begins */
15         $S_{\mathcal{G}}$ = SABRE ($\mathcal{R}(\mathcal{W}, \bar{o})$, $\mathcal{SA}$, $t - \mathcal{D}_{\mathcal{W}\bar{o}}$);
16         foreach $x \in \mathcal{R}(\mathcal{W}, e\bar{o})$ do
17             Find $\mathcal{G} \in S_{\mathcal{G}}$, which contains $x$;
18             Put $\mathcal{G}$ to $\mathcal{S}_{out}$;
19             Delete $\mathcal{G}$ from $S_{\mathcal{G}}$                /* DIRECT end */
20 Case 3:
21     else                                                  /* INDIRECT begins */
22         foreach $\mathcal{G} \in S_{\mathcal{G}}$ do
23             Add $\mathcal{G}$ to $\mathcal{S}_{out}$;
24             Delete $\mathcal{G}$ from $S_{\mathcal{G}}$               /* INDIRECT ends */
25     Update $\mathcal{W}$ by advancing $\mathcal{I}$ tuples;
```

[3] $\mathcal{R}(\mathcal{W}_2, \bar{o})$ is *not* the set of not-yet-output tuples of $\mathcal{W}_2$ at instant ι_4; it is the set of not-yet-output tuples of $\mathcal{W}_2$ at instant ι_2 when $\mathcal{W}_2$ is advancing.

After the anonymization of $\mathcal{R}(\mathcal{W}, e\bar{o})$, window $\mathcal{W}$ advances (step 25).

In Algorithm $\mathsf{SABRE_W}$, each time $\mathcal{I}$ new tuples arrive (step 5), window $\mathcal{W}$ will advance by $\mathcal{I}$ tuples (steps 8, 25). Therefore, as $\mathcal{W}$ continuously advances, a sequence of windows is generated. We use $\mathcal{W}$ to represent each generated window, but, for the sake of clarity, in the following formal proof we will use different notations to represent different windows. For example, we use $\mathcal{W}$ to represent the window at instant $\imath$ and $\mathcal{W}'$ to represent the window at instant $\imath'$. We also note that DIRECT or $\mathsf{INDIRECT}$ is applied only when a window is advancing. The correctness of $\mathsf{SABRE_W}$ depends on the three cases discussed above. Its formal proof will be presented in Theorem 6. We first prove that *Case 3* follows (ω, t)-closeness by Theorem 5, which is based on the following three lemmas.

Lemma 2 *Assume that ω is the window size of input stream, each window advances by $\mathcal{I}$ tuples, and $\omega = j \cdot \mathcal{I}$, where $j > 0$ is an integer. Then, for two overlapping windows $\mathcal{W}'$ and $\mathcal{W}$, with $\mathcal{W}'$ generated before $\mathcal{W}$, the first $\mathcal{I}$ tuples in $\mathcal{W}$ also fall in $\mathcal{W}'$.*

Proof Since window size ω is a multiple of advance size $\mathcal{I}$, once two windows overlap, their overlapping area contains at least $\mathcal{I}$ tuples. Furthermore, since $\mathcal{W}'$ is generated before $\mathcal{W}$, the first $\mathcal{I}$ tuples in $\mathcal{W}$ are in the overlapping area, i.e., the first $\mathcal{I}$ tuples in $\mathcal{W}$ also fall in $\mathcal{W}'$. $\square$

If $\mathsf{INDIRECT}$ is applied in Algorithm $\mathsf{SABRE_W}$, then $j > 1$. The proof is by contradiction. For $j = 1$ ($\omega = \mathcal{I}$), whenever a window advances, all its tuples are expiring and will be output. This makes the anonymizations in all the windows independent. Therefore, when a window $\mathcal{W}$ advances, $\mathcal{R}(\mathcal{W}, \bar{o}) = \mathcal{R}(\mathcal{W})$ and $\mathcal{D}_{\mathcal{W}\bar{o}} = 0$. Hence the precondition for $\mathsf{INDIRECT}$ to be applied does not hold.

Lemma 3 *Given a window $\mathcal{W}$, assume that* $\mathsf{INDIRECT}$ *is applied when it advances at instant $\imath$, and that at instant $\imath'$, $\imath' < \imath$, there is an application of* DIRECT *and no further application of* DIRECT *between $\imath'$ and $\imath$. Then there is no application of* $\mathsf{INDIRECT}$ *between $\imath'$ and $\imath$ either.*

Proof The advance of $\mathcal{W}$ incurs the call of $\mathsf{INDIRECT}$, so $|\mathcal{R}(\mathcal{W}, e\bar{o})| > 0$ and $\mathcal{D}_{\mathcal{W}\bar{o}} \geq t$ at instant $\imath$ (the preconditions of *Case 3*). Assume that there is AI, an application of $\mathsf{INDIRECT}$, between $\imath'$ and $\imath$. Let $\mathcal{W}'$ be the window, whose advance at instant $\imath'$ incurs the application of DIRECT. Suppose that DIRECT partitions $\mathcal{R}(\mathcal{W}', \bar{o})$ (not-yet-output tuples of $\mathcal{W}'$ at instant $\imath'$) into $\mathcal{S_G}$, a list of ECs. AI releases those and only those ECs that are in $\mathcal{S_G}$ to the output stream (steps 22–24). We make two points: First, all tuples in $\mathcal{W}'$ have been output after AI; Second, AI outputs tuples only in $\mathcal{W}'$. There are two possibilities for an overlapping relationship between $\mathcal{W}'$ and $\mathcal{W}$. *Possibility 1: $\mathcal{W}'$ overlaps with $\mathcal{W}$*. By Lemma 2, $\mathcal{R}(\mathcal{W}, e\bar{o})$, a subset of the first $\mathcal{I}$ tuples of $\mathcal{W}$, falls in $\mathcal{W}'$. All the tuples in $\mathcal{W}'$ are already output after AI (point 1), so $|\mathcal{R}(\mathcal{W}, e\bar{o})| = 0$ at instant $\imath$. This conclusion contradicts the fact that $|\mathcal{R}(\mathcal{W}, e\bar{o})| > 0$. *Possibility 2: $\mathcal{W}'$ does not overlap with $\mathcal{W}$*. In $\mathsf{SABRE_W}$ only DIRECT and $\mathsf{INDIRECT}$ output tuples. During the period between $\imath'$ and $\imath$ there is no further application of DIRECT, hence only AI outputs tuples in this period. Since AI outputs tuples only in $\mathcal{W}'$ (point 2), none of the tuples in $\mathcal{W}$ has been output when $\mathcal{W}$ begins to slide. Therefore, at instant $\imath$, $\mathcal{R}(\mathcal{W}, \bar{o}) = \mathcal{R}(\mathcal{W})$ and $\mathcal{D}_{\mathcal{W}\bar{o}} = 0$; this conclusion contradicts the fact that $\mathcal{D}_{\mathcal{W}\bar{o}} \geq t$. In conclusion, under the assumption that there is an application of $\mathsf{INDIRECT}$ between $\imath'$ and $\imath$, none of the two possibilities is borne. By reductio ad absurdum, we conclude that there is no application of $\mathsf{INDIRECT}$ between $\imath'$ and $\imath$. $\square$

Intuitively, Lemma 3 indicates that one application of $\mathsf{INDIRECT}$ cannot be followed immediately by another application of $\mathsf{INDIRECT}$. There must be at least one application of DIRECT between them.

Lemma 4 *Given a window $\mathcal{W}$, assume that* $\mathsf{INDIRECT}$ *is applied when it advances at instant $\imath$. Then there is at least one application of* DIRECT *before instant $\imath$.*

Proof We prove this also by contradiction. Assume that there is no application of DIRECT before instant $\imath$. Let the list of generated windows be $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_i, \mathcal{W}_{i+1}, \ldots$. When $\mathcal{W}_1$ begins to advance, all its tuples are not-yet-output (i.e., $\mathcal{R}(\mathcal{W}_1, \bar{o}) = \mathcal{R}(\mathcal{W}_1)$). Thus, DIRECT is applied. Therefore, for any window $\mathcal{W}_i (i > 1)$, if $\mathcal{W}_i$ advances at instant $\imath_i$, there is at least one application of DIRECT before instant $\imath_i$. Since there is no application of DIRECT before instant $\imath$ at the advancing moment of $\mathcal{W}$, then $\mathcal{W}$ must be $\mathcal{W}_1$. However, this conclusion contradicts the precondition that $\mathsf{INDIRECT}$ is applied when $\mathcal{W}$ slides. By contradiction, the assumption is incorrect, and there is at least one application of DIRECT before instant $\imath$. $\square$

Theorem 5 *In Algorithm* $\mathsf{SABRE_W}$*, given a window $\mathcal{W}$, assume that* $\mathsf{INDIRECT}$ *is applied when it advances at instant $\imath$. Let $\mathcal{R}(\mathcal{W}, e\bar{o})$ be the expiring and not-yet-output tuples in $\mathcal{W}$ at instant $\imath$. Then each tuple in $\mathcal{R}(\mathcal{W}, e\bar{o})$ is contained in one EC of $\mathcal{S_G}$.*

Proof The advance of $\mathcal{W}$ incurs the call of $\mathsf{INDIRECT}$, so $|\mathcal{R}(\mathcal{W}, e\bar{o})| > 0$ and $\mathcal{D}_{\mathcal{W}\bar{o}} \geq t$ at instant $\imath$ (the preconditions of *Case 3*). Assume that LD, the last application of DIRECT before instant $\imath$, occurs at instant $\imath'$, $\imath' < \imath$ (there is no further application of DIRECT between $\imath'$ and $\imath$). By Lemma 4, it follows that LD exists. Let $\mathcal{W}'$ be the window whose advance at instant $\imath'$ incurs LD. Since $\mathcal{S_G}$ records the list of ECs generated by the last application of DIRECT, we know that LD partitions $\mathcal{R}(\mathcal{W}', \bar{o})$ (not-yet-output tuples of $\mathcal{W}'$ at instant ι') into $S_{\mathcal{G}}$, a list of ECs (step 15).

Assume that $\mathcal{W}'$ does not overlap with $\mathcal{W}$. During the period between ι' and ι, there is neither an application of INDIRECT (Lemma 3) nor an application of DIRECT. Still, only DIRECT and INDIRECT output tuples. In effect, there is no tuple output between ι' and ι. In addition, at instant ι' only ECs containing tuples in $\mathcal{W}'$ have been released to the output stream (steps 16–19). We conclude that none of tuples in $\mathcal{W}$ has been output when $\mathcal{W}$ slides. So at instant ι, $\mathcal{R}(\mathcal{W}, \bar{o}) = \mathcal{R}(\mathcal{W})$ and $\mathcal{D}_{\mathcal{W}\bar{o}} = 0$, which contradicts the fact that $\mathcal{D}_{\mathcal{W}\bar{o}} \geq t$. By contradiction, it follows that $\mathcal{W}'$ overlaps with $\mathcal{W}$. By Lemma 2 we know that $\mathcal{R}(\mathcal{W}, e\bar{o})$, a subset of the first $\mathcal{I}$ tuples in $\mathcal{W}$, falls in $\mathcal{W}'$, i.e., $\mathcal{R}(\mathcal{W}, e\bar{o}) \subset \mathcal{R}(\mathcal{W}', \bar{o})$. Therefore, each tuple in $\mathcal{R}(\mathcal{W}, e\bar{o})$ is contained in one EC of $S_{\mathcal{G}}$. In addition, each EC of $S_{\mathcal{G}}$ is t-close to $\mathcal{W}'$ with respect to its SA distribution. □

Theorem 6 *Algorithm* SABRE$_W$ *follows the* (ω, t)*-closeness and expiring constraint.*

Proof Given an input stream $\mathcal{S}_{in}$, SABRE$_W$ anonymizes it into an output stream $\mathcal{S}_{out}$ by the three cases we have presented. We discuss these three cases one by one.

Case 1 (steps 7–9). Since all the expiring tuples are already output, the expiring constraint is met. This case does not anonymize and output any tuple, so the satisfaction of (ω, t)-closeness depends on the remaining two cases.

Case 2 (steps 14–19). Given an advancing window $\mathcal{W}$ at instant ι, for each expiring and not-yet-output tuple, an EC is formed and released to $\mathcal{S}_{out}$. As each expiring tuple is output, the expiring constraint is met. Besides, since each output EC is t-close to $\mathcal{W}$, (ω, t)-closeness is satisfied.

Case 3 (steps 21–24). Given an advancing window $\mathcal{W}$ at instant ι, by Theorem 5, each expiring and not-yet-output tuple is contained in an EC formed by tuples from another window $\mathcal{W}'$, generated before $\mathcal{W}$. All ECs containing expiring tuples are output, hence the expiring constraint is met. Each output EC is t-close to $\mathcal{W}'$, so this case attains (ω, t)-closeness as well.

Table 5 The CENSUS dataset

Attribute	Cardinality	Type
Age	79	Numerical (4)
Sex	2	Categorical (1)
Education	17	Numerical (4)
Marital status	6	Categorical (2)
Race	9	Categorical (1)
Work class	10	Categorical (3)
Birth place	83	Categorical (2)
Salary	50	Numerical

In conclusion, SABRE$_W$ abides to both (ω, t)-closeness and the expiring constraint. □

6 Experimental study

In this section, we evaluate the performance of our SABRE-based schemes: SABRE-KNN, SABRE-AK, and SABRE$_W$. We compare SABRE against tIncognito [18] and tMondrian [19], i.e., the t-closeness schemes extended from Incognito [14] and Mondrian [15], respectively. The prototype was implemented in Java and the experiments were run on a core-2 duo 2.33 GHz CPU machine, with 4 GB RAM, running windows XP. We use the CENSUS dataset [1], which contains 5,00,000 tuples, and has 8 attributes as shown in Table 5; the value following the type is the height of the corresponding attribute hierarchy. For instance, attribute *marital status* is categorical and has a hierarchy of height 2. The first 7 attributes are used by default as the QI, and the last one (i.e., *salary*) as the sensitive attribute. We generate 5 microdata tables by randomly taking 1,00,000–5,00,000 tuples from the dataset; the one with 1,00,000 tuples is the default dataset. t-closeness provides protection against the disclosure of SA values. Yet it does not handle identity disclosure. k-anonymity copes with identity disclosure by ensuring that each released tuple is indistinguishable from at least $k-1$ other tuples with respect to their QI values. Still, to create a level playing field, as tIncognito and tMondrian do, SABRE combines t-closeness and k-anonymity together. We set the default value of k to be 6 (i.e., the size of EC is at least 6). The closeness threshold t is a variable, in default it is set to 0.35.

We use several metrics to evaluate the quality of the anonymized dataset under the schemes we compare. First, we measure the average information loss (see Sect. 3.2). Then we study the utility of the anonymized dataset using *median relative errors* [31] and *KL-divergence* [10]. We also compare the efficiency of the various schemes based on the elapsed time.

6.1 Basic results

We first study the effect of varying the closeness threshold t. Figure 10 shows the results. As expected, as t grows and the requirement for similarity between the *salary* distribution in each EC and that in the whole table is relaxed, the information quality for all schemes is improved (Fig. 10a). The two SABRE-based schemes are about equally effective (with SABRE-KNN slightly better than SABRE-AK) and provide superior information quality (i.e., lower average information loss) compared to both tIncognito and tMondrian. The benefit of a scheme tailored for t-closeness emerges. After all, SABRE selects tuples from buckets in a sophisticated

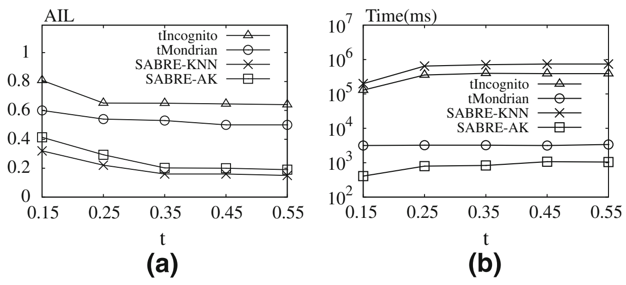

Fig. 10 Effect of varying closeness threshold

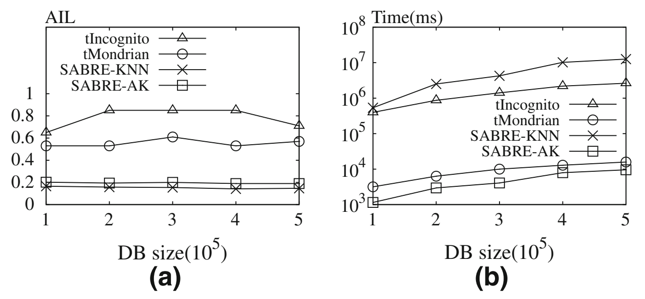

Fig. 12 Effect of varying $\mathcal{DB}$ dimensionality (size)

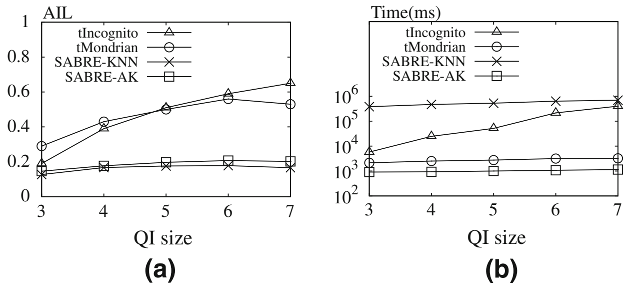

Fig. 11 Effect of varying QI size

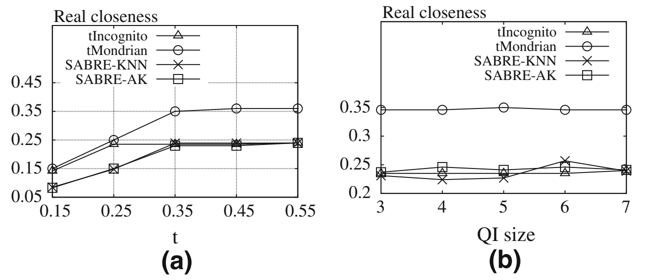

Fig. 13 Real closeness

manner, and forms ECs with tuples of as close as possible QI values. Thus, it competes successfully against the schemes of both tIncognito and tMondrian; these schemes were principally designed for the less complex problem of k-anonymization. Moreover, as Fig. 10b shows, SABRE-AK is the most efficient. In all cases, it takes no more than 4 s to complete the processing. The time efficiency of tMondrian is comparable to SABRE-AK. They are two orders of magnitude faster than the other two methods.

Next, we investigate the effect of the QI dimensionality (size). We vary the QI size from 3 to 7. When the QI dimensionality increases, data becomes more sparse in the QI space, due to the higher-dimensional degrees of freedom offered; thus, the formed ECs are more likely to have bigger minimum bounding boxes. Thus, we expect information quality to worsen as dimensionality grows for all methods. Still, Fig. 11a shows that the average information loss of both tIncognito and tMondrian grows in a substantially steep manner as QI size grows, while the SABRE-based schemes degrade only marginally. Thus, the information quality gap between tIncognito/tMondrian and the SABRE-based schemes *widens* as the QI size increases. This result clearly indicates that the SABRE-based methods are more scalable with respect to QI size. Moreover, Fig. 11b shows that SABRE-AK is the fastest, followed by tMondrian, tIncognito, and SABRE-KNN.

Our next experiment studies the effect of database size. We vary the size of microdata table from 1,00,000 to 5,00,000 tuples. The results are illustrated in Fig. 12. As Fig. 12a shows, the data size has no much effect on the information quality of the schemes, except tIncognito. Still, as expected, the elapsed time increases as the table size grows for all schemes; SABRE-AK and tMondrian remain superior in this case as well.

SABRE guarantees that the *EMD* difference between the distribution of a sensitive attribute in an EC and that in the original input table is at most t. However, the actual closeness of the anonymized data may be smaller than t. We check the real closeness of the anonymized data as follows. For each EC, we calculate its closeness value, and we use the maximum one as the real closeness of the whole anonymized dataset. In Fig. 13a, we vary t from 0.15 to 0.55. In Fig. 13b, we set t to 0.35, and vary the QI size from 3 to 7. We observe that the real *EMD* difference of all schemes is smaller than the given threshold t. This result indicates that the anonymized data achieves better privacy than the requirement. Furthermore, the SABRE-based schemes and tIncognito achieve consistently smaller real difference than tMondrian; this result indicates that SABRE and tIncognito offer better privacy than tMondrian.

Figure 14 presents the results of the four approaches as we vary the k parameter of the k-anonymity guarantee that all methods also provide. As k increases, identity protection is improved. However, a larger k implies that more tuples will be in an EC, hence the minimum bounding box to cover them becomes larger, which results in higher information loss. The behavior of tIncognito is not uniform, due to its highly heuristic nature.

In next experiment, we study the performance of SABRE in the context of data stream. The instantiation of SABRE

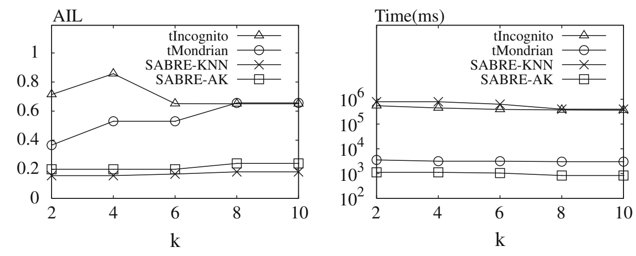

Fig. 14 Effect of varying k

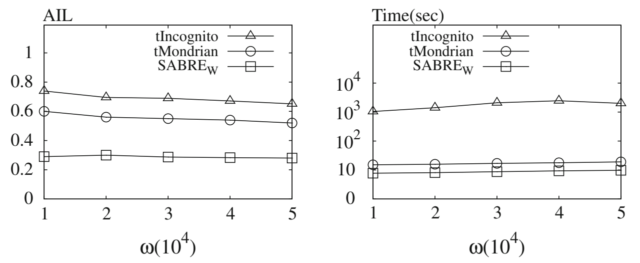

Fig. 16 Effect of varying WS (Streaming Data)

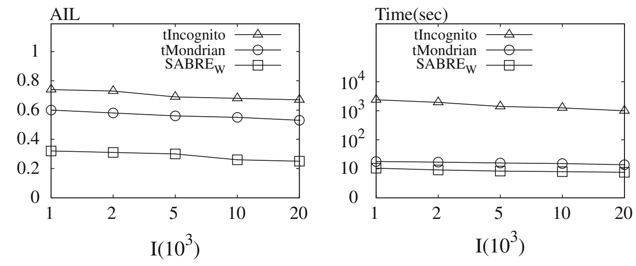

Fig. 15 Effect of varying $\mathcal{I}$ (Streaming Data)

employed by Algorithm SABRE$_W$ is SABRE-AK. We extend tIncognito and tMondrian to the context of streaming data in a similar fashion. We use the whole CENSUS dataset to simulate the stream. Figure 15 shows our results with the window size ω set at 20,000, and varying the value of $\mathcal{I}$, the time interval of window advance. Let $|S|(|S| >> \omega)$ be the size of stream, then the number of times to advance the window is $\frac{|S|-\omega}{\mathcal{I}} \approx \frac{|S|}{\mathcal{I}}$. Every time the window advances there is one anonymization. So when $\mathcal{I}$ approaches ω, the anonymization process is invoked fewer times, hence the elapsed time decreases. Figure 16 shows our results with $\mathcal{I}$ set at 5,000, and varying the window size. As ω grows, the cost of single anonymization also increases, hence the elapsed time rises as well. As shown in the figures, SABRE$_W$ retains its superiority in the data streaming context, outperforming tIncognito and tMondrian by a wide margin in information quality. In addition, SABRE$_W$ and, to a certain extent, tMondrian perform much better than tIncognito in terms of time efficiency.

6.2 Accuracy of aggregation queries

Apart from information loss, we also study the utility of the anonymized data. In this section, we focus on aggregation queries as they are the basis of statistical analysis and many data mining applications (e.g., association rule mining and decision trees). We first consider the following type of aggregation queries with the *median relative error* as the metric [31]:

```
SELECT COUNT(*) FROM Anonymized-data
WHERE pred($A_1$) AND ... AND pred($A_\lambda$)
AND pred($SA$)
```

Each A_i is a QI attribute. $\mathcal{SA}$ is a sensitive attribute. The query has predicates on the λ randomly selected QI attributes and SA. Let A be one of these $\lambda + 1$ attributes (λ QI attributes + SA). $pred(A)$ has the form of $A \in R$. R is a random interval in the domain of A. R has the length of $|A| \cdot \theta^{\frac{1}{\lambda+1}}$, where $|A|$ is the domain length of A and θ is the *expected selectivity*. Given a query, the precise result $prec$ is computed from the original table, and the estimated result est is obtained from the anonymized table. To calculate est, we assume that tuples in each EC are uniformly distributed, and consider the intersection between the query and the EC. We define $\frac{|est-prec|}{prec} \times 100\%$ as the *relative error*. Our workload consists of 10,000 queries, and we measure the workload error as the *median relative error*. Relative error is undefined when $prec$ is 0. If $prec$ in a given query is 0, we drop that query.

We first set θ to 0.1, and vary λ. Each predicate in the WHERE clause except the one on SA has some error. As λ increases, the number of predicates increases, hence the overall error of all the predicates is expected to increase as well. Not surprisingly, the error increases as λ grows, as shown in Fig. 17a. In the following experiments we fix λ to 3. Next we set θ to 0.1, and vary the QI size. As the QI size increases, data tend to be more sparse in the QI space, and it is more likely that ECs with bigger minimum bound boxes are created. Consequently, the information loss of the anonymized data grows. As information quality worsens, the error we measure will also increase. Expectedly, in Fig. 17b the workload error increases with QI size. In Fig. 17c we fix θ to 0.1 and vary t. As t grows, the requirement on the distribution of each EC is relaxed, hence the information quality of anonymized data rises, and the error we measure drops. Finally, in Fig. 17d we vary the selectivity θ. When θ increases, the range R for each attribute in a predicate increases. This makes the minimum bound box of an EC more likely to be entirely contained in the query region, therefore, the estimate becomes slightly more accurate and the error smaller. In all the above experiments, we find that SABRE-based schemes offer better utility. Remarkably, SABRE-based techniques outperform tIncognito by one order of magnitude in terms of median relative error.

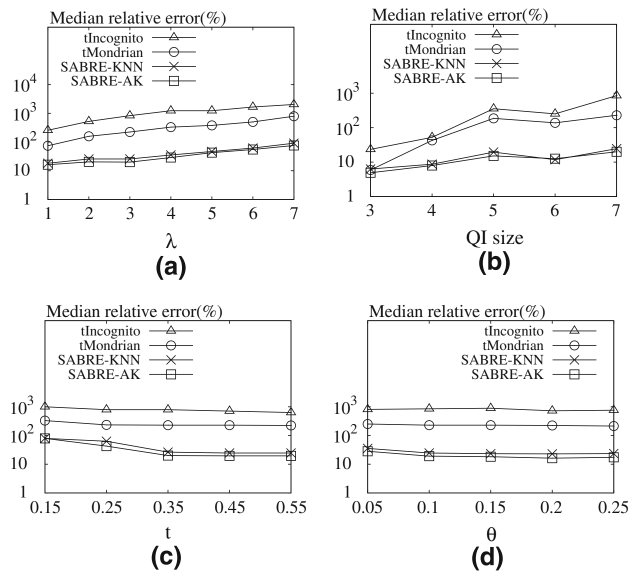

Fig. 17 Median relative error

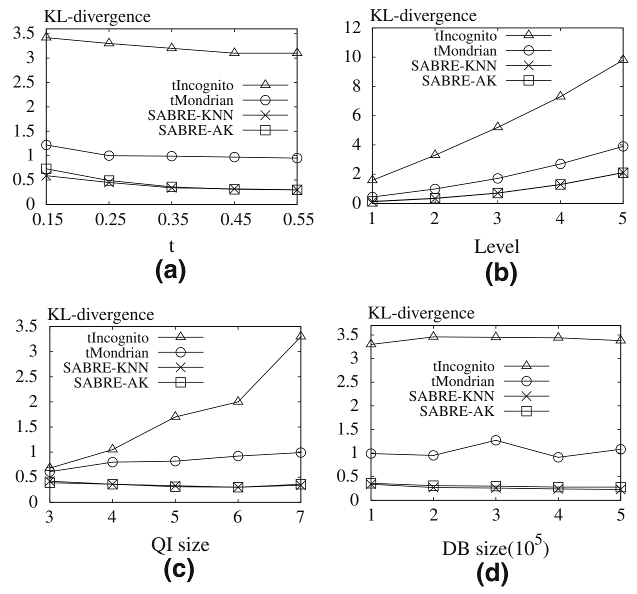

Fig. 18 *KL-divergence* with OLAP queries

In the following, we evaluate the utility based on typical OLAP queries using the *KL-divergence* metric (as in [10]):
SELECT $A_1, A_2, \ldots, A_\mu$, COUNT(*)
FROM Anonymized-data
WHERE $\mathcal{SA}$ = val
GROUP BY $A_1, A_2, \ldots, A_\mu$
All GROUP-BYs for all possible combinations of the QI attributes compose the OLAP datacube lattice. Level μ of the lattice corresponds to all GROUP-BYs over exactly μ attributes. We build two datacube lattices on the CENSUS dataset: α (on the original dataset), and β (on the anonymized dataset). For each cell of β, we consider the intersection between the cell and each EC, assuming a uniform distribution of tuples within the EC. Let α_c and β_c be values of a cell in α and β respectively. We use *KL-divergence* to measure the difference between the cells in α and those in β:

$$KL-divergence(\alpha, \beta) = \sum_{\forall cell\ c} \alpha_c \times \log \frac{\alpha_c}{\beta_c}$$

In Fig. 18a, the level of the lattice is set to 2, and we vary t. As t increases, *KL-divergence* decreases. This is so because, for larger t, the closeness requirement is relaxed, hence the information quality of the anonymized data is improved. Ideally, the lower the *KL-divergence* is, the better the quality of the anonymized data. When all the cells in α and β are the same, *KL-divergence* is 0. In Fig. 18b we vary the lattice level. When the level is higher, the granularity of the GROUP-BYs in the aggregation query becomes finer. On the other hand, when the level is lower, an aggregation query is more likely to include the whole range of an anonymized EC. This effect makes the four schemes perform better at lower levels. In Fig. 18c we set the level of lattice to 2, and vary the QI size. SABRE-KNN and SABRE-AK are significantly more scalable than the other two methods with growing QI size. Eventually, in Fig. 18d we set the level to 2, and vary the size of the dataset, to get a slightly improving trend of information quality, due to increasing data density. In all the above comparisons, SABRE-based schemes clearly outperform tIncognito and tMondrian.

7 Discussion

In this section, we make a discussion on SABRE. First, we examine the question about the effect of the SA hierarchy on information quality. For a categorical SA, the set of children of a split node in the bucketization tree depends on the pre-defined domain hierarchy of SA. With Fig. 1 as the hierarchy, the root of the bucketization tree is *respiratory and digestive diseases*. If it is split, its children will be *respiratory diseases* and *digestive diseases*, which are pre-defined in Fig. 1. Thus, the bucketization process is affected by the SA hierarchy. Besides, the structure of a hierarchy is shaped by the number of its leaves and the fanouts of its root and internal nodes (the height is automatically decided thereby). We assume that the hierarchy is well defined by its domain expert.

In Table 5, `salary` is a numerical sensitive attribute with 50 distinct values. Next, we consider them as 50 leaves, and build hierarchies over them to see the effects on anonymization process. To simplify the problem, we assume that the fanouts in a simulated hierarchy are uniform, while its leaves are sorted from left to right by the ascending order of their values. Figure 19 presents our experimental results. We observe

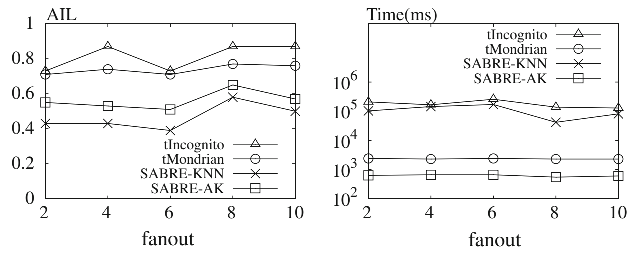

Fig. 19 Effect of varying fanout

that the information quality of the anonymized data does not change uniformly as a function of fanout. Still, overall the curves suggest that a smaller value of fanout tends to preserve more information. After all, when the fanout is smaller, the hierarchy is deeper. Thus, the intermediate levels between the root and leaves are increased, hence the bucketization gains more flexibility in partitioning $\mathcal{SA}$ values to buckets at several levels. In effect, the redistribution phase can later redistribute tuples to ECs more effectively.

Apart from the preceding discussion, we also examine the potential for an extension of SABRE to settings with more than one sensitive attributes. Without loss of generality, assume that $\mathcal{DB}$ has two sensitive attributes $SA_1 = \{u_1, u_2, \ldots, u_n\}$ and $SA_2 = \{v_1, v_2, \ldots, v_m\}$. For the sake of simplicity we consider the case that SA_1 is independent of SA_2. SABRE is extended to attain t-closeness with respect to both SA_1 and SA_2 as follows. It first transforms $\mathcal{DB}$ to $\mathcal{DB}_1$, which satisfies t-closeness with respect to SA_1. Then it checks each EC in $\mathcal{DB}_1$ to determine whether it also observes t-closeness with respect to SA_2. If it does not, then it is merged with its nearest neighbor ECs, until t-closeness is attained. This effect is always achievable because an EC formed from the union of two ECs will not have an increased distance from $\mathcal{DB}$ with respect to SA_2 distribution. In particular, let $\mathcal{P}$ be the overall SA_2 distribution in $\mathcal{DB}$, and $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be the SA_2 distributions of two ECs in $\mathcal{DB}_1$, respectively. Let $\mathcal{Q}$ be the SA_2 distribution of the EC formed from the union of the two ECs; then, $EMD(\mathcal{P}, \mathcal{Q}) \leq \max\{EMD(\mathcal{P}, \mathcal{Q}_1), EMD(\mathcal{P}, \mathcal{Q}_2)\}$ [18]. In effect, after all the required merges of ECs in $\mathcal{DB}_1$, we can transform $\mathcal{DB}_1$ to $\mathcal{DB}_2$, which attains t-closeness with respect to both SA_1 and SA_2.

Lastly, we discuss the applicability of other distance metrics in SABRE. Let d be a distance measure, and $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ be any three sensitive attribute distributions. To be applicable in SABRE, d needs to satisfy the following properties: I. *non-negativity*: $d(\mathcal{P}_1, \mathcal{P}_2) \geq 0$; II. *identity of indiscernibles*: $d(\mathcal{P}_1, \mathcal{P}_2) = 0$ if and only if $\mathcal{P}_1 = \mathcal{P}_2$; and III. *triangle inequality*: $d(\mathcal{P}_1, \mathcal{P}_3) \leq d(\mathcal{P}_1, \mathcal{P}_2) + d(\mathcal{P}_2, \mathcal{P}_3)$. If d also has the property of *symmetry*, then d is a metric. Therefore, besides *EMD*, other measures such as Euclidean metric and Hamming distance can also be applied to SABRE. However, neither the *KL-divergence* nor the Jensen-Shannon divergence has the triangle inequality property, hence they cannot be applied to SABRE. Still, the square root of the Jensen-Shannon divergence is a metric and therefore a possible candidate distance measure for SABRE. Nevertheless, when a new distance measure is applied to SABRE, the upper bound of the cost related with a bucket (Theorem 2) needs to be customized for that measure.

As a final note, t-closeness based on *EMD* has the drawback that it defines no clear intelligible relationship between t and the privacy it affords. However, *EMD* is still a meaningful distance measure for t-closeness, due to its following attractive properties [19]: 1. Awareness of semantic closeness; 2. Simplicity for understanding; 3. Subset property (i.e., if table $\mathcal{DB}$ satisfies t-closeness in QI, then it also satisfies t-closeness in any subset of QI.)

8 Conclusion and future work

This paper proposed SABRE, a novel framework for distribu-tion-aware microdata anonymization based on the t-closeness principle. SABRE guarantees t-closeness in an elegant and efficient manner, without depending on techniques developed for other privacy models. We have shown the applicability of our scheme on both categorical and numerical attributes. Our extensive experimental study demonstrated that our two SABRE instantiations, SABRE-AK and SABRE-KNN, clearly outperform previous schemes in terms of information quality, while SABRE-AK also outperforms them in terms of efficiency. In conclusion, SABRE provides the best known resolution of the tradeoff between privacy, information quality, and computational efficiency with a t-closeness guarantee in mind.

An interesting direction for future work is to develop a scheme that auto-detects and proposes a t value that achieves good balance between privacy and data utility. In SABRE, the actual closeness between the distribution of sensitive values in each EC and that in the whole dataset can be smaller than the t threshold. Exploiting this fact in order to achieve an even better tradeoff between privacy and information quality is another item in our agenda for future research. We have applied SABRE to data stream, a dynamic setting with append-only ordered tuples. Still, another dynamic setting is the one of table updates with both tuple insertion and deletion. Therefore, a third possible future work is the republication of an updated table with respect to t-closeness.

Acknowledgments We thank Tiancheng Li and Ninghui Li from Purdue University for kindly providing us the implementations of tIncognito and tMondrian. This work is supported by two AcRF grants from Singapore's MOE. Jianneng Cao and Kian-Lee Tan are partially supported by grant T12-0702-P02. Panagiotis Karras is supported by grant T1 251RES0807.

References

1. http://www.ipums.org
2. Aggarwal, G., Feder, T., Kenthapadi, K., Khuller, S., Panigrahy, R., Thomas, D., Zhu A.: Achieving anonymity via clustering. In: Proceedings of PODS (2006)
3. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Anonymizing tables. In: Proceedings of ICDT (2005)
4. Bayardo, R.J., Agrawal, R.: Data privacy through optimal k-anonymization. In: Proceedings of ICDE (2005)
5. Byun, J.-W., Sohn, Y., Bertino, E., Li, N.: Secure anonymization for incremental datasets. In: Secure Data Management, pp. 48–63 (2006)
6. Cao, J., Carminati, B., Ferrari, E., Tan, K.-L.: Castle: A delay-constrained scheme for ks-anonymizing data streams. In: Proceedings of ICDE (2008)
7. Cao, J., Carminati, B., Ferrari, E., Tan, K.-L.: Castle: Continuously anonymizing data streams. Accepted by IEEE Transactions on Dependable and Secure Computing (2009)
8. Fung, B.C.M., Wang, K., Fu, A.W.-C., Pei, J.: Anonymity for continuous data publishing. In: Proceedings of EDBT (2008)
9. Fung, B.C.M., Wang, K., Yu, P.S.: Top-down specialization for information and privacy preservation. In: Proceedings of ICDE (2005)
10. Ghinita, G., Karras, P., Kalnis, P., Mamoulis, N.: Fast data anonymization with low information loss. In: Proceedings of VLDB (2007)
11. Ghinita, G., Karras, P., Kalnis, P., Mamoulis, N.: A framework for efficient data anonymization under privacy and accuracy constraints. ACM Trans. Database Syst. **34**(2), 1–47 (2009)
12. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: Proceedings of KDD (2002)
13. Kooiman, P., Willenborg, L., Gouweleeuw, J.: Pram: A method for disclosure limitation for microdata. Research paper/Statistics Netherlands (9705) (1997)
14. LeFevre, K., DeWitt, D.J., Ramakrishnan R.: Incognito: Efficient full-domain k-anonymity. In: Proceedings of SIGMOD (2005)
15. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: ICDE (2006)
16. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Workload-aware anonymization. In: Proceedings of KDD (2006)
17. Li, J., Tao, Y., Xiao, X.: Preservation of proximity privacy in publishing numerical sensitive data. In: Proceedings of SIGMOD (2008)
18. Li, N., Li, T., Venkatasubramanian S.: t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In: Proceedings of ICDE (2007)
19. Li, N., Li, T., Venkatasubramanian, S.: Closeness: a new privacy measure for data publishing. In: To appear in IEEE Transactions on Knowledge and Data Engineering (2009)
20. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: ℓ-diversity: privacy beyond k-anonymity. In: Proceedings of ICDE (2006)
21. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: Proceedings of PODS (2004)
22. Moon, B., Jagadish, H.V., Faloutsos, C., Saltz, J.H.: Analysis of the clustering properties of the hilbert space-filling curve. IEEE Trans. Knowl. Data Eng. **13**(1), 124–141 (2001)
23. Pei, J., Xu, J., Wang, Z., Wang, W., Wang, K.: Maintaining k-anonymity against incremental updates. In: Proceedings of SSDBM (2007)
24. Rebollo-Monedero, D., Forné, J., Domingo-Ferrer, J.: From t-closeness-like privacy to postrandomization via information theory. In: To appear in IEEE Transactions on Knowledge and Data Engineering (2009)
25. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vis. **40**, 99–121 (2000)
26. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. **13**(6), 1010–1027 (2001)
27. Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information (abstract). In: Proceedings of PODS (1998)
28. Wang, K., Fung, B.C.M.: Anonymizing sequential releases. In: Proceedings of KDD (2006)
29. Xiao, X., Tao, Y.: Anatomy: simple and effective privacy preservation. In: Proceedings of VLDB (2006)
30. Xiao, X., Tao, Y.: M-invariance: towards privacy preserving re-publication of dynamic datasets. In: Proceedings of SIGMOD (2007)
31. Xiao, X., Tao, Y.: Dynamic anonymization: accurate statistical analysis with privacy preservation. In: Proceedings of SIGMOD (2008)
32. Xu, J., Wang, W., Pei, J., Wang, X., Shi, B., Fu, A.W.-C. Utility-based anonymization using local recoding. In: Proceedings of KDD (2006)
33. Zhang, Q., Koudas, N., Srivastava, D., Yu, T.: Aggregate query answering on anonymized tables. In: Proceedings of ICDE (2007)