Scalable Local-Recoding Anonymization using Locality Sensitive Hashing for Big Data Privacy Preservation

Xuyun Zhang
The University of Auckland
xuyun.zhang@auckland.ac.nz

Christopher Leckie
The University of Melbourne
caleckie@unimelb.edu.au

Wanchun Dou
Nanjing University
douwc@nju.edu.cn

Jinjun Chen
University of Technology, Sydney
jinjun.chen@gmail.com

Ramamohanarao Kotagiri
The University of Melbourne
kotagiri@unimelb.edu.au

Zoran Salcic
The University of Auckland
z.salcic@auckland.ac.nz

ABSTRACT

While cloud computing has become an attractive platform for supporting data intensive applications, a major obstacle to the adoption of cloud computing in sectors such as health and defense is the privacy risk associated with releasing datasets to third-parties in the cloud for analysis. A widely-adopted technique for data privacy preservation is to anonymize data via local recoding. However, most existing local-recoding techniques are either serial or distributed without directly optimizing scalability, thus rendering them unsuitable for big data applications. In this paper, we propose a highly scalable approach to local-recoding anonymization in cloud computing, based on Locality Sensitive Hashing (LSH). Specifically, a novel semantic distance metric is presented for use with LSH to measure the similarity between two data records. Then, LSH with the Min-Hash function family can be employed to divide datasets into multiple partitions for use with MapReduce to parallelize computation while preserving similarity. By using our efficient LSH-based scheme, we can anonymize each partition through the use of a recursive agglomerative k-member clustering algorithm. Extensive experiments on real-life datasets show that our approach significantly improves the scalability and time-efficiency of local-recoding anonymization by orders of magnitude over existing approaches.

Keywords

Big Data, Privacy Preservation, LSH, MapReduce, Cloud

1. INTRODUCTION

The emergence of storing and manipulating big data on cloud computing platforms is creating new privacy and security challenges for the IT industry [5, 23]. While cloud computing has become an attractive platform for supporting data intensive applications, a major obstacle to the adoption of cloud computing in sectors such as health and defense is the privacy risk associated with releasing datasets to third-parties in the cloud for analysis. The datasets in such big data applications, e.g., electronic health records, often contain personal privacy-sensitive information, while they offer significant economic and social benefits if analyzed or mined.

Research Motivation. Privacy concerns are aggravated in the context of cloud computing, due to the redundancy of information in large datasets, and the access risks created by ubiquitous access and multi-tenancy in cloud environments. Third-party organizations can easily tap into diverse data sources such as social networks data and medical records, leading to potential infringement of individuals' privacy rights when linking the datasets together for innovative or value-added insights. Therefore, privacy protection for big data applications and infrastructure is still one of the major concerns for both cloud users and providers [5, 23].

Data anonymization has been extensively studied and widely adopted for data privacy preservation in non-interactive data sharing and dissemination scenarios [11]. It refers to hiding the identity and/or masking privacy-sensitive data so that the privacy of an individual is preserved while special types of aggregate information can be exposed to data users for diverse analysis and mining tasks. A variety of privacy models and data anonymization approaches have been recently proposed [11, 19, 17, 1]. Recently, scalability issues of large-scale data anonymization have drawn researcher' attention [17, 15]. However, applying traditional approaches to big data anonymization poses unique scalability and efficiency challenges due to the data volume, complexity and velocity.

The local-recoding anonymization scheme (a.k.a. cell generalization) groups data records in a dataset into a set of cells and generalizes each cell. Existing approaches [1, 25, 18, 4] model the local-recoding problem as a k-member clustering problem [18], where k is the k-anonymity parameter [22]. Clustering is a natural and effective way to achieve local recoding, while the top-down partitioning method [12, 17] adopted in global schemes like the multidimensional and sub-tree schemes fails to accomplish local recoding.

Limitations of the State-of-the-art. Most existing local-recoding techniques are either serial or distributed without directly optimizing scalability, thus rendering them unsuitable for data intensive applications. Specifically, a class of greedy clustering algorithms with time complexity $O(n^2)$

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA

DOI: http://dx.doi.org/10.1145/2983323.2983841

are proposed for the k-member clustering problem in [25, 18, 4]. The underlying assumption that datasets can fit into memory often fails to hold for very large datasets [10], thus severely limiting their scalability in practice. Given the increasingly abundant availability of big data cloud infrastructure, this raises the question of how to exploit parallelism for the k-member clustering problem.

Although several clustering methods based on the parallel computing paradigm MapReduce [7] have been proposed recently for traditional clustering problems [10, 9], it is still difficult to apply these methods directly to the k-member clustering problem. The k-member clustering problem poses a constraint on the size of each cluster, whereas traditional clustering problems are more concerned with the number of clusters to reflect the inherent clustering structure of the data. Specifically, agglomerative clustering fails to handle very large datasets due to its high time complexity. It is also hard to adopt agglomerative clustering for use with MapReduce due to its inherently serial structure. Point-assignment clustering appears to be a candidate to leverage MapReduce for local recoding due to its ease of parallelization. However, it is a challenge to select and dispatch seeds for this method in MapReduce because of the intrinsic features of k-member clustering. The number of seeds will be approximately $1/k$ of the original data size. Usually, k is far smaller than the original data size. As a result, the number of required seeds will be huge for very large datasets. Another problem is that the size of each cluster cannot be controlled in the point-assignment process, implying that it will fail to satisfy the k-anonymity requirement or requires extra effort to adjust the clusters.

Proposed Solution. In this paper, we address the local-recoding anonymization problem for big data privacy preservation by integrating a recursive partitioning scheme based on Locality Sensitive Hashing (LSH) [14] and MapReduce. To handle the huge volume of data, we leverage the Divide-and-Conquer strategy which is widely adopted in big data solutions using MapReduce [10]. Concretely, the original dataset is recursively split into smaller partitions, and k-member clustering is performed on these partitions in parallel during the partitioning process, thus enhancing both scalability and efficiency. To ensure low data distortion during anonymization, it is required that similar data items are put into the same partition when splitting the dataset. Consequently, it is a challenge to partition very large datasets in a scalable and efficient way while meeting this requirement. Accordingly, we propose to leverage LSH to address this challenge due to its several salient properties: linear time complexity, ability to hash data records beyond (below) a similarity threshold into the same partition with extremely high (low) probability, and parallelization capability.

LSH is only applicable to a limited range of distance measures, e.g., Euclidean distance [2] and Jaccard distance [3]. However, no LSH family has been developed for semantic distances which are often used in generalization based anonymization methods such as local recoding. Consequently, it is a challenge to exploit LSH directly on the conventional semantic distance involved in local recoding. In generalization methods, an attribute value of a data record is usually generalized to a higher level concept to hide specific information while exposing coarser but semantically consistent information. Conventionally, the distance between two attribute values is defined by the path length between them in a user-defined taxonomy tree [25, 18, 4]. To take advantage of the salient features of LSH, we propose a novel semantic distance measure based on the Jaccard distance which defines the distance between two sets. An example is presented below to explain the intuition behind our distance measure.

Table 1: Hospital Patient Data

No.	Name	Sex	Zip.	Edu.	Disease
1	Alice	Female	53715	Master	Flu
2	Bob	Male	53706	Doctorate	Hepatitis
3	Ellen	Female	53710	Doctorate	HIV
4	Beth	Male	53703	12th	Bronchitis

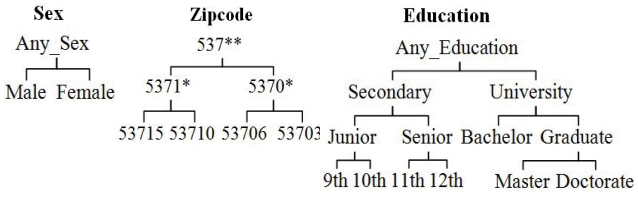


Figure 1: Taxonomy trees for *Sex*, *Zipcode* and *Education*.

Illustrative Example. Table 1 shows a dataset containing four patient records, and Fig. 1 illustrates the corresponding taxonomy trees, where the attribute *Disease* is regarded as privacy-sensitive. In terms of our semantic distance measure (elaborated later), we first create two sets $\{53715, 5371*\}$ and $\{53706, 5370*\}$ to capture the distance of *Zipcode* between Alice and Bob. Note that the root value of the tree is not included. Then, the distance is defined as the Jaccard distance between the two sets, i.e., 1, because they have no common element. Similarly, the distance of *Zipcode* between Alice and Ellen is the set distance between $\{53715, 5371*\}$ and $\{53710, 5371*\}$, i.e., 2/3.

Equipped with this new distance, we employ LSH based on the MinHash function family [3], which has been widely used as a fast method to estimate set similarity. Specifically, a data record is converted into a sparse binary vector while preserving semantic similarity information. The data records are grouped in terms of the hash values calculated from the binary vectors. Once we obtain smaller data partitions, we perform local-recoding anonymization over the partitions in parallel. Based on LSH again, we propose a recursive k-member agglomerative clustering algorithm. The core idea is to utilize LSH recursively to split a partition into a set of small clusters of size not larger than k, and then perform k-member agglomerative clustering over these small clusters. Our LSH based recursive clustering algorithm can be far more time and space efficient than traditional ones that initially treat each single data record as a cluster. Finally, MapReduce is used to implement our algorithms for high scalability and efficiency. We conduct extensive experiments on real-life datasets, and show that our approach significantly improves the scalability and efficiency of local recoding by orders of magnitude over existing approaches.

Contributions. The main contributions of our research are fourfold. First, we propose a novel semantic distance to measure the similarity between two data records for use with LSH. Second, a fast recursive k-member clustering algorithm is proposed based on LSH and agglomerative clustering for local recoding. Third, we show our method can be

implemented effectively by exploiting the MapReduce programming model. Finally, through extensive experiments we show that our approach improves the scalability and time efficiency of local-recoding anonymization by orders of magnitude over existing approaches while preserving privacy.

At present, both the Differential Privacy (DP) model [8] and syntactic anonymity models are widely adopted for data privacy preservation [6]. The DP model has good theoretical basis for protecting privacy. However, it can be very noisy and vulnerably for exponential number of queries. In many practical cases, syntactic anonymity is preferable as it guarantees the correctness of anonymized data. Usually, syntactic anonymity models are applicable in non-interactive data sharing or publishing scenarios, while DP supports interactive data mining or query scenarios [6]. Note that the two paradigms are not necessarily mutually exclusive, and recent efforts have combined them together to boost privacy preservation [6, 21], e.g., k-anonymization can be made differentially private by using random sampling. As such, it is still both practically and theoretically meaningful to investigate syntactic anonymisation problems. Furthermore, we study the anonymisation problem herein from the scalability perspective, thereby being orthogonal to privacy models. Besides the k-anonymity model studied herein, our approach can work with other advanced privacy models with proper adjustment.

The remainder of this paper is organized as follows. The next section reviews related work. In Section 3, our approach is elaborated, covering a new distance, LSH based data partitioning and recursive clustering, and algorithmic details. The approach is empirically evaluated in Section 4. We conclude this paper and discuss future work in Section 5.

2. RELATED WORK

Privacy-preserving data publishing has been surveyed in [11]. We herein briefly review existing research on local recoding and scalability issues in data anonymization.

Recently, clustering techniques have been extensively leveraged to achieve local-recoding anonymization for privacy preservation. Xu et al. [25] studied the anonymization of a local-recoding scheme from the utility perspective and proposed a bottom-up greedy approach and its top-down counterpart. Byun et al. [4] formally modeled local-recoding anonymization as a k-member clustering problem, which requires that the cluster size should not be less than k to achieve k-anonymity, and accordingly proposed a simple greedy clustering algorithm. Li et al. [18] investigated the inconsistency issue of local-recoding anonymization for data with hierarchical attributes, and proposed the KACA (K-Anonymization by Clustering in Attribute hierarchies) algorithm. The above research models local recoding as a k-member clustering problem, and employs greedy clustering techniques. Aggarwal et al. [1] proposed a set of constant factor approximation algorithms for two clustering based anonymization problems, i.e., r-GATHER and r-CELLULAR CLUSTERING, where cluster centers and radiuses are published but the trustworthiness at the record level fails to be guaranteed. However, all methods above suffer from scalability problems when applied to big datasets due to their intrinsic sequential characteristic.

The problem of scalability has been studied for some other anonymization schemes. LeFevre et al. [17] addressed the problem in the multidimensional scheme by introducing scalable decision trees and sampling techniques. Iwuchukwu et al. [15] proposed an R-tree index-based approach by building a spatial index over datasets to achieve high efficiency. Fung et al. [12, 20] proposed a top-down specialization method, which improved the efficiency of the sub-tree scheme by exploiting a data structure named Taxonomy Indexed PartitionS (TIPS). Also, Zhang et al. [27] addressed the sub-tree anonymization problem in the context of big data by exploiting MapReduce for high scalability. However, these approaches use the either multidimensional or sub-tree schemes, thereby failing to work for local recoding. Moreover, the indexing techniques in these methods are unable to address the local-recoding problem because local recoding is conducted in a clustering manner rather than in a top-down partitioning way. Moreover, local recoding has a far larger search space to find the minimally optimal solution than the other two schemes. Close to the work herein, Zhang et al. [26] mainly investigated the local-recoding scheme against proximity privacy breaches (attribute linkage attacks). The work herein is dedicated to the k-member clustering problem and we propose an LSH based anonymization approach, which is orthogonal and complementary to the work in [26].

Locality Sensitive Hashing (LSH) has been extensively explored and adopted due to its ability to preserve spatial neighbours with high probability [14]. Koga et al. [16] use LSH for fast agglomerative hierarchical clustering. But their approach assumes that the maximum coordinate value of any point is below a constant number. Ghinita et al. [13] propose an Approximate Nearest Neighbor (ANN) search based method for anonymizing sparse high-dimensional data, where LSH is used to implement an efficient ANN search.

3. LOCALITY SENSITIVE HASHING BASED LOCAL-RECODING ANONYMIZATION

3.1 Approach Overview

Let D denote a dataset containing n data records of the form $r = \langle v_1, v_2, \cdots, v_m, sv\rangle$, where m is the number of attributes, sv is a privacy-sensitive attribute value, and v_i, $0 \leq i \leq m$, are non-sensitive attribute values which can potentially be linked to individuals uniquely, e.g., *age, zipcode* and *sex*, if combined with other external datasets. These non-sensitive values together are named as a quasi-identifier of the form $qid = \langle q_1, q_2, \cdots, q_m\rangle$, where q_i, $0 \leq i \leq m$ are the original or anonymized values. Under k-anonymity, local recoding is conventionally modeled as a k-member clustering problem [4], aiming to find a set of clusters $\mathbb{C}$, such that each cluster contains at least k ($k > 0$) records (the k-anonymity requirement), and the overall data distortion or information loss is minimized. Formally, it is modeled by:

$$min \sum_{t=1,\cdots,l} IL(C_t), \text{ s.t.:}$$
$$1). \bigcup_{i=1,\cdots,l} C_i = D, \text{and } C_i \cap C_j = \emptyset, 1 \leq i \neq j \leq l,$$
$$2). \forall C \in \mathbb{C}, k \leq |C| \leq 2k-1,$$

where $IL(C_t)$ is the information loss incurred by generalizing records in C_t into their common quasi-identifer. Several data metrics have been proposed to capture data distortion [11], e.g., *ILoss* [24] and the discernibility metric [25]. Without loss of generality, we employ *ILoss* herein because it can capture the information loss of generalization directly. Note that other metrics can be applied to our approach as well and similar conclusions will be drawn.

We propose a Locality Sensitive Hashing (LSH) based

local-recoding approach for big data anonymization using MapReduce. Our approach consists of two main phases. The original dataset is split into smaller partitions by using LSH in the first phase, and in the second phase an LSH based recursive k-member clustering algorithm is run on each partition in parallel. LSH is leveraged in data partitioning to gain scalability and efficiency, and ensure similar records are placed into the same partitions with high probability. Note that the false positive and negative errors may occur in this step, i.e., similar records are put into different partitions or dissimilar records are put into the same partitions. But these errors are tolerable without substantially affecting the data utility of the final anonymous data, because partition sizes are usually much larger than resultant clusters whose sizes are slightly larger than k. As the partitioning process in the first phase can be regarded as a coarse clustering, the partitions produced are named as β-cluster for convenience. In contrast, a final cluster for k-anonymization is named as a k-member cluster. In the second phase, a recursive clustering method is proposed based on LSH to achieve k-member clustering on each β-cluster. Although this method is serial in essence, we run it on multiple Reducers simultaneously as 'plug-ins' in MapReduce like in [10], to gain high scalability.

Algorithm 1 sketches the main steps of our approach. The MinHash family is used to fulfill LSH in our approach. To facilitate MinHashing, the original data records are converted to binary vectors in Step 1. The transformation is required to preserve the semantic similarity so that the MinHashing result can reflect the similarity of the original records. The LSH based partitioning is accomplished in Step 2. Step 3 performs the LSH based recursive clustering algorithm on β-clusters in parallel. The final step generalizes the records in a cluster to produce the final anonymous dataset. Details of each step are elaborated in the following subsections.

Algorithm 1 LSH based Local Recoding with MapReduce.

Input: Dataset D, privacy parameter k, LSH parameter α.
Output: Anonymous dataset $D^\star$.

1: Convert data record r to binary vector r^B;
2: Run MinHash based LSH on $D^B = \{r^B\}$, to obtain a set of β-clusters: $\mathbb{C}^\beta = \{C_1^\beta, \cdots, C_t^\beta\}$, $t = |\mathbb{C}^\beta|$;
3: For each β-cluster $C_i^\beta \in \mathbb{C}^\beta$, $1 \le i \le t$: run the LSH based recursive clustering algorithm on C_i^β, to obtain k-member clusters $\mathbb{C}_i = \{C_{i1}, \cdots, C_{im_i}\}$, $m_i = |\mathbb{C}_i|$;
4: For each cluster $C_j \in \mathbb{C}$, $\mathbb{C} = \bigcup_{i=1}^{t} \mathbb{C}_i$, generalize C_j to $C_j^\star$, replacing each attribute value with a general one;
5: Return $D^\star = \bigcup_{j=1}^{m_j} C_j^\star$, where $m_j = \sum_{i=1}^{t} m_i$.

3.2 Provenance Set based Semantic Distance

Many quasi-identifier attributes are categorical with a user-defined taxonomy tree, e.g., *Zipcode* and *Education* in Fig. 1, whose similarity is usually captured by the semantic closeness among their domain values. Although LSH is a powerful and appealing technique to find similar items, no LSH function family is available for the semantic distance conventionally defined by taxonomy trees [18, 4]. It is desired to convert the form of original data to another where we can define an LSH family. To this end, we propose a novel semantic distance measure on quasi-identifiers. Note that numerical attributes can be hashed directly using the LSH family with L_p ($p \in (0, 2]$) distance [2]. But to focus on the main idea of our approach, we assume that the numerical attributes are discretized into taxonomy trees in our approach, a common way to handle numerical attributes, e.g., [12, 20].

Semantic Distance for Categorical Values. Given a taxonomy tree T for an attribute, we define the notion of the provenance set of a categorical value in the tree as follows.

Definition 1. (*Provenance Set of a categorical value*) The provenance set of a categorical value v is the set that consists of itself and all its ancestors except the topmost one. Let $P^{i+1}(v)$ denote the parent of the value $P^i(v)$ in T, $0 \le i \le H$, where H is the height of T and $P^0(v)$ is defined as the value v itself. The provenance set of a value v, denoted as PS_v, is constructed by $PS_v = \{v, P^1(v), \cdots, P^{H-1}(v)\}$. Then, we can have $|PS_v| = H$.

For example, the provenance set of the value $Master$ in Fig. 1 is $PS_{Master} = \{Master, Graduate, University\}$.

Note that if the path length from a value to the topmost node in the taxonomy tree is less than H, we extend the path by adding nodes with height information to make its length H. For instance, the provenance set of the value $Bachelor$ in Fig. 1 is $PS_{Bachelor} = \{Bachelor, Bachelor^1, University\}$.

Definition 2. (*Provenance Set based Semantic Distance between two values*) For any two categorical values v and v' from the same taxonomy tree, the provenance set based semantic distance between them is defined as:

$$d_{PS}(v, v') \triangleq d_J(PS_v, PS'_v), \tag{1}$$

where $d_J(PS_v, PS'_v)$ is the Jaccard distance measure.

Note that $d_{PS}(v, v')$ is a distance metric as it is defined based on the Jaccard distance. To compute the distance, let the Least Common Ancestor (LCA) of two values v and v' be denoted as $LCA(v, v')$, and the path length from v or v' to $LCA(v, v')$ be i, $0 \le i \le H$. Then, it can be derived that $P^i(v) = P^i(v') = LCA(v, v')$, and for any l, $i \le l \le H$, $P^l(v) = P^l(v')$. Note that when $l = H$, $P^H(v) = P^H(v')$ still holds because they are the topmost node of the taxonomy tree. Accordingly, $d_{PS}(v, v')$ can be computed by

$$d_{PS}(v, v') = 1 - \frac{|PS_v \cap PS'_v|}{|PS_v \cup PS'_v|} = \frac{2i}{H + i}. \tag{2}$$

Semantic Distance for Quasi-identifiers. We extend the notion of provenance set to quasi-identifiers. Let $U_{qid} = \{v_i | v_i = Proj_i(qid)\}$ denote the set of the single categorical values contained in qid, where $Proj_i(qid)$ is a projection function that gets the i^{th} coordinate. Then we define the provenance set based on U_{qid}.

Definition 3. (*Provenance Set of a quasi-identifier*) The provenance set of a quasi-identifier $qid = \langle v_1, v_2, \cdots, v_m \rangle$ is the union of the provenance sets of the values contained in qid. Formally, let PS_{pid} denote the provenance set for a quasi-identifier, i.e., $PS_{pid} = \bigcup_{i=1}^{m} PS_{v_i}$, $v_i \in U_{pid}$.

For example, the provenance set of Alice's quasi-identifier is $PS_{\langle Female, 53715, Mastr \rangle} = \{Female, 53715, 5371*, Master, Graduate, University\}$.

Definition 4. (*Provenance Set based Semantic Distance between two quasi-identifiers*) For any two quasi-identifiers

$qid = \langle v_1, v_2, \cdots, v_m\rangle$ and $qid' = \langle v'_1, v'_2, \cdots, v'_m\rangle$, their provenance set based semantic distance is defined as:

$$d_{PS}(qid, qid') \triangleq d_J(PS_{qid}, PS_{qid'}), \tag{3}$$

Analogous to the case of a single value, $d_J(PS_{qid}, PS_{qid'})$ is a distance metric. As the intersection of any pair of provenance sets of two values from different attributes is an empty set, the Jarccard similarity coefficient between PS_{qid} and $PS_{qid'}$ can be derived by

$$J(PS_{qid}, PS_{qid'}) = \frac{|PS_{qid} \cap PS_{qid'}|}{|PS_{qid} \cup PS_{qid'}|} = \frac{\sum_{l=1}^{m} |PS_{v_l} \cap PS_{v'_l}|}{\sum_{l=1}^{m} |PS_{v_l} \cup PS_{v'_l}|}.$$

Let the path length from v_l or v'_l to $LCA(v_l, v'_l)$ be i_l, $0 \le l \le m$. Then, $d_{PS}(PS_{qid}, PS_{qid'})$ is calculated by

$$d_{PS}(PS_{qid}, PS_{qid'}) = \frac{2\sum_{l=1}^{m} i_l}{\sum_{l=1}^{m} H_l + \sum_{l=1}^{m} i_l}. \tag{4}$$

For example, the distance between Alice's and Ellen's quasi-identifiers shown in Table 1 can be calculated by $d_{PS}(\langle Female, 53715, Master\rangle, \langle Female, 53710, Doctorate\rangle) = \frac{2*(0+1+1)}{1+2+3} = \frac{2}{3}$.

To determine whether the provenance set based distance for categorical values is sound and reasonable, we explore its relationship with the commonly used distance in existing work [25, 18, 4] where the semantic distance between two categorical values is conventionally defined based on their path in the tree. For convenience, we call this a path based distance. The distance between v and v' is usually given as

$$d(v, v') \triangleq L(v, v')/(2H), \tag{5}$$

where $L(v, v')$ is the shortest path length between v and v'. Note that $2H$ is the maximum path length between any two nodes in the tree. $L(v, v')$ can be computed with ease by finding the LCA of v and v'. Note that $d_{PS}(v, v') = \delta, (0 < \delta < 1)$ does not imply $d(v, v') = \delta$, and vice versa.

Conventionally, the path based distance between two quasi-identifiers qid and qid' is often defined by

$$d(qid, qid') \triangleq \sum_{i=1}^{m} \omega_i d(v_i, v'_i), \tag{6}$$

where ω_i, $1 \le i \le m$, is the weight for the i^{th} quasi-identifier attribute. As the provenance set based distance treats each attribute equally, let ω_i be $1/m$ to facilitate the comparison.

Since the two types of quasi-identifier distance fail to produce the same value of similarity, we examine if the provenance set based distance has the same distinguishability as the path based one, i.e., if the latter can discriminate that a quasi-identifier qid is more similar to qid' than another qid'', the former should draw the same conclusion rather than qid is more similar to qid'' than qid'. This desirable and appealing property is formally defined as follows.

Definition 5. (*Semantic Similarity Order Preserving Property*) Two distance metrics $d_x(\cdot,\cdot)$ and $d_y(\cdot,\cdot)$ are said to be semantic similarity order preserving if the following condition holds: For any three values v, v' and v'':

$$d_x(v, v') \le d_x(v, v'') \Leftrightarrow d_y(v, v') \le d_y(v, v'').$$

Actually, the similarity order information is necessary and sufficient to make decisions in many real-world applications. For instance, when choosing points in clustering algorithms, the information that two points are closer than other pairs is more important than the exact distance value of the two points when determining which points should be clustered.

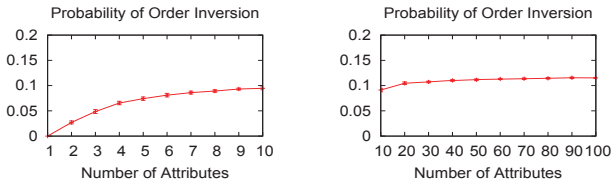


Figure 2: Order inversion frequency w.r.t. #(attributes).

However, the provenance set based distance of two quasi-identifiers fails to possess the semantic similarity order preserving property (but the distance on single categorical values has such a property), i.e., for quasi-identifiers qid, qid' and qid'', $d(qid, qid') \le d(qid, qid'') \not\Leftrightarrow d_{PS}(qid, qid') \le d_{PS}(qid, qid'')$. For example, let $H_1 = 3$, $H_2 = 5$, $\langle i_1, i_2\rangle = \langle 3, 0\rangle$ and $\langle i'_1, i'_2\rangle = \langle 3, 0\rangle$ for two quasi-identifier pairs $\langle qid, qid'\rangle$ and $\langle qid, qid''\rangle$, respectively. $d(qid, qid') = \frac{3}{10} < d(qid, qid'') = \frac{1}{3}$, but $d_{PS}(qid, qid') = \frac{6}{11} > d(qid, qid'') = \frac{2}{5}$.

Intuitively, implicit constraints exist between (4) and (6). We empirically reveal the order relationship between the two inequalities $d(qid, qid') \le d(qid, qid'')$ and $d_{PS}(qid, qid') \le d_{PS}(qid, qid'')$. We use the notion of order inversion to describe the phenomenon that the order-preserving property is violated. Fig. 2 shows the change of the probability of order inversion with respect to the number of quasi-identifier attributes, ranging from 1 to 10 (the left subfigure), and 10 to 100 (the right subfigure). We assume the data is uniformly distributed. The sample size of quasi-identifier pairs is 100,000. Each group of experiments is repeated 20 times.

Observation 1. The probability that the order-preserving property is violated is relatively low, around 10%

In fact, Observation 1 derived from Fig. 2 explicitly captures the implicit constraints. Therefore, we maintain that the provenance set based similarity measure can guide local recoding without substantially affecting the data utility.

3.3 LSH based Partitioning with MapReduce

Converting Quasi-identifier into Binary Vector. To make use of the MinHash LSH family, it is necessary to convert the provenance set of a quasi-identifier to a binary vector representation which is named as a characteristic vector. The characteristic vector of a quasi-identifier qid is denoted as $V^B(qid)$, where B suggests "Binary". Let U denote the universe of all quasi-identifier attribute values in provenance sets, and Rt_i be the root of the taxonomy tree T_i, $1 \le i \le m$. Then, $U = \bigcup_{i=1}^{m} DOM_i \backslash \{Rt_i\}$. We set the size of characteristic vectors to be the size of the universe, i.e., $|V^B(qid)| \leftarrow |U|$ for any qid. Let each bit in $V^B(qid)$ correspond to an element in U. A bit is set as 1 if and only if its corresponding element is in PS_{qid}. Otherwise, the bit is set as 0. In total, the number of 1s is $\sum_{i=1}^{m} H_i$ in a characteristic vector. The vector is sparse if the number of attribute values is large, which is common in real-world applications.

Algorithm 2 details how to convert a quasi-identifier to its characteristic vector. The subroutine `contact` in Line 7 is to concatenate two binary vectors together. For example, Alice's quasi-identifier $\langle Female, 53715, Master\rangle$ can be con-

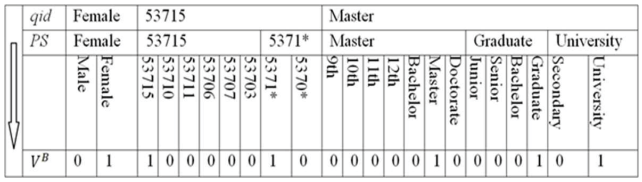


Figure 3: Convert Alice's Quasi-identifier into a 0-1 Vector.

verted into $\langle 0,1,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,0,1\rangle$ in terms of Algorithm 2, as illustrated in Fig. 3. With characteristic vectors, the MinHash family can be utilized to estimate $d_{PS}(qid, qid')$.

Algorithm 2 Characteristic Vector Converting.

Input: Quasi-identifier $qid = \langle v_1, v_2, \cdots, v_m\rangle$.
Output: Characteristic vector $V^B(qid)$.

```
Initialize V^B(qid) as an empty vector;
for taxonomy tree TT_i, 1 ≤ i ≤ m do
    for level L_j, 1 ≤ j ≤ H_i do
        Initialize vector V_ij^B with |L_j| zeros;
        if P^{j-1}(v_i) equals to the k^th value in L_j then
            Set the k^th bit of V_ij^B to be 1;
        V^B(qid) ← concat(V^B(qid), V_ij^B);
Return V^B(qid).
```

LSH Preliminary. The core idea of LSH is to hash similar data points into the same buckets with high probability [14]. Given a distance metric $d(\cdot,\cdot)$, let $d_1 < d_2$ be two distance values, and $p_1,\ p_2$ $(0 \le p_1,\ p_2 \le 1)$ be two probability values. A family of functions $\mathcal{F}$ is said to be $(d_1,\ d_2,\ p_1,\ p_2)$-sensitive if for every $h \in \mathcal{F}$, if we have:

1). $d(x,y) \le d_1 \Rightarrow Pr[h(x) = h(y)] \ge p_1$;
2). $d(x,y) \ge d_2 \Rightarrow Pr[h(x) = h(y)] \le p_2$;

The event $h(x) = h(y)$ means x and y are hashed into the same bucket. Ideally, it is expected that p_1 and p_2 can be driven apart as far as we wish while keeping d_1 and d_2 fixed. Indeed, this is achieved by amplifying a LSH family with the banding technique. Given a family of $\alpha \cdot \gamma$ hash functions, this technique splits the hash values into γ bands of α rows, where $\alpha, \gamma > 0$ are two integers selected by the user. With the banding technique, two points will be hashed into the same bucket if all their α hash values in at least one out of the γ bands are equal. Let s be the similarity between two points. The probability they are hashed into the same bucket is $1 - (1 - s^\alpha)^\gamma$.

MinHash is a technique for quickly estimating set similarity [3]. Let h be a hash function that maps the elements of a set S to an integer. The MinHash on S is defined to be the minimum hash value, i.e., $h_{min}(S) = min_{x \in S}\{h(x)\}$. For any two sets A and B, the following equality holds [3]: $Pr[h_{min}(A) = h_{min}(B)] = J(A,B)$, where $J(A,B)$ represents the Jaccard similarity coefficient of A and B.

LSH based Data Partitioning with MapReduce. The data partitioning process using LSH consists of two main steps, namely, hashing records into buckets and constructing partitions via merging buckets. If the banding parameter $\gamma > 1$, then multiple copies of a record will exist in several buckets. Such redundancy builds the similarity connection among buckets so that we can construct larger groups of similar records. However, as our purpose is to partition the dataset, it is necessary to remove $\gamma - 1$ redundant copies when merging buckets, which will be very time-consuming. Still, in the MapReduce paradigm, if γ and the number of Reducers are larger than one, it is hard to remove all redundant copies as a Reducer will fail to determine whether more copies of an item exist in other Reducers. If only one Reducer is used, the scalability issue will emerge. Hence, the parameter γ is set as 1 in our approach.

Consequently, we use the other banding parameter α to control the granularity of the data partitions. The probability that two records are hashed into the same bucket is s^α with $\gamma = 1$. If s is close to 0, i.e., the two records are considerably dissimilar, s^α will rapidly approach to 0 when α grows linearly. Hence, we can limit the chance of putting dissimilar records into a common bucket to an extremely low level by increasing α, without substantially affecting the probability of putting similar records into the same buckets. Due to this salient property, the data partitions can still retain relatively the high level clustering structures.

Algorithm 3 describes the *Map* function that accomplishes the MinHash LSH based data partitioning. The core of the algorithm is to generate the bucket ID for each data record. Simply, a bucket ID is produced by concatenating all the MinHash values together. After emitting key-value pairs, the *Shuffle* phase of MapReduce sorts and aggregates the pairs in terms of their bucket ID. Each Reducer receives all records within a bucket. A bucket here is a β-cluster as well. This process is fully parallel in nature.

Algorithm 3 Data Partitioning with MinHash LSH (Map).

Input: Data record (ID_r, r), $r \in D$; banding parameter α, hash function family of row permutation $\mathcal{F}$.
Output: Key-value pair $\langle bucketID, r\rangle$.

```
Convert r to characteristic vector V^B(r) by Algorithm 2;
Initialize bucketID with an empty string;
for i, 1 ≤ i ≤ α do
    Compute MinHash value via hashing h_{a,b}(x) ∈ F, i.e.,
    minHash_i ← h_min(V^B(r)), described in Algorithm 4;
    bucketID ← concat(bucketID, minHash_i);
Emit (bucketID, r).
```

As a subroutine for Algorithm 3, Algorithm 4 shows how to compute the MinHash value from a characteristic vector. The function $Proj_i(V^B(r))$ returns the i^{th} coordinate of characteristic vector $V^B(r)$. Hash functions in $\mathcal{F}$ are in the form of $h_{a,b}(x) = (ax+b)\ mod\ N_{Prime}$, where a and b are random integers, and N_{Prime} is the smallest prime number larger than $|U|$. Functions $h_{a,b}(x)$ can be regarded as row permutations of $V^B(r)$, required by MinHash computation.

Algorithm 4 MinHash Calculating.

Input: Characteristic vector $V^B(r)$; row permutation hash function $h_{a,b}(x) \in \mathcal{F}$.
Output: MinHash value $minHash$.

```
Initialize minHash, i.e., minHash ← +∞;
for i, 1 ≤ i ≤ |U| do
    if Proj_i(V^B(r)) == 1 then
        index ← h_{a,b}(i) mod |U|;
        If index < minHash, update minHash ← index;
Return minHash.
```

It can be seen from Algorithm 3 and Algorithm 4 that the

time and space complexity of the *Map* function are $O(1)$. Therefore, it is highly time-efficient and scalable.

3.4 LSH based Recursive Clustering

LSH based Recursive Clustering. After partitioning the data, we can perform k-member clustering for local recoding on Reducers as 'plug-ins'. Due to data skewness, the size of β-clusters after LSH based partitioning can vary significantly. Data skewness is quite common in most real-world datasets. Large β-clusters still give rise to scalability issues for a Reducer running traditional k-member clustering algorithms. To address such issues, we propose a recursive clustering method by using LSH again.

The basic idea is that a β-cluster of size larger than k is recursively partitioned into a set of smaller β-clusters by MinHash based LSH until the sizes of all the resultant β-clusters are less than or equal to k. Agglomerative clustering is conducted on the set of resultant β-clusters to produce k-member clusters. Intuitively, the recursive partitioning process is fast as it can be completed in linear time. As to the agglomerative clustering process, it will be time and space efficient because the number of resultant β-clusters is relatively small and the size of some β-clusters are close to k. Algorithm 5 elaborates the details, where the symbols and notations are self-explanatory. The LSH based Recursive Clustering algorithm is abbreviated as LSH-RC.

Algorithm 5 LSH based Recursiving Clustering (LSH-RC).

Input: β-cluster C, privacy and binding parameters: k, α.
Output: k-member clusters $\mathbb{C}$, remaining β-cluster C^r.

1: Initialize $\mathbb{C} \leftarrow \emptyset$; $C^r \leftarrow \emptyset$; small cluster set $\mathbb{C}^S \leftarrow \emptyset$;
2: **if** $|C| < k$ **then**
3: $C^r \leftarrow C$;
4: **else if** $|C| == k$ **then**
5: $\mathbb{C} \leftarrow \mathbb{C} \cup \{C\}$;
6: **else**
7: Partition C: β-clusters $\mathbb{C}^\star \leftarrow$ `LSH_Partitioning`(C);
8: **for** $C^\star \in \mathbb{C}^\star$ **do**
9: **if** $|C^\star| < k$ **then**
10: $\mathbb{C}^S \leftarrow \mathbb{C}^S \cup \{C^\star\}$;
11: **else if** $|C^\star| == k$ **then**
12: $\mathbb{C} \leftarrow \mathbb{C} \cup \{C^\star\}$;
13: **else**
14: $(\mathbb{C}', C') \leftarrow$ `LSH_RC`$(C^\star, k, \alpha)$;
15: $\mathbb{C} \leftarrow \mathbb{C} \cup \mathbb{C}'$; $\mathbb{C}^S \leftarrow \mathbb{C}^S \cup \{C'\}$;
16: Perform *beta*-cluster based aggolomerative clustering on small clusters $\mathbb{C}^S$: $((\mathbb{C}'', C'') \leftarrow$ `Beta_AC`$(\mathbb{C}^S)$;
17: $\mathbb{C} \leftarrow \mathbb{C} \cup \mathbb{C}''$; $C^r \leftarrow \mathbb{C}''$.

LSH-RC invokes two subroutines. One is the LSH based data partitioning algorithm, denoted as LSH-Partitioning. Note that this algorithm is a serial version rather than the parallel one described in Algorithm 3, but they share the same spirit. The serial one can be implemented by adding a mapping table between *bucketID* and its bucket based on Algorithm 3. Thus, we omit the algorithmic details of the serial version. The other subroutine is the β-cluster based Agglomerative Clustering algorithm abbreviated as Beta-AC, which performs the k-member clustering concretely. Beta-AC's algorithmic details will be described in the next subsection.

In LSH-RC, a remaining cluster is a cluster of size less than k, which fails to be merged with other clusters. LSH-RC returns it to a higher level to merge with the higher level clusters. The small clusters of size less than k from the partitioning process or remaining clusters from the lower levels are retained in $\mathbb{C}^s$. At the end of the recursion, these small clusters are k-member clustered by the Beta-AC algorithm.

Beta-cluster based Agglomerative Clustering. According to Algorithm 5, the number of small clusters will not exceed the number of resultant β-clusters produced in Line 7 of LSH-RC. Usually, the number of data partitions is relatively small compared with the number of records in the dataset producing the partitions. Hence, it is promising to leverage the agglomerative clustering method, rather than the greedy method, for k-member clustering of the small clusters. In this way, the data distortion can be reduced compared with the greedy methods. The rationale of adopting agglomerative clustering here is that performing the clustering scheme on small clusters can be regarded as a later stage of performing it on the same number of records from scratch. Usually, the agglomerative method initializes each record as a cluster, and performs pair-wise merging. The number of clusters decreases gradually. Much more time and space are required in the earlier stages due to the large number of clusters. However, the LSH based partitioning can circumvent the computation in the these stages. Hence, our approach can be time and space efficient even after adopting the agglomerative clustering technique.

To determine which two clusters should be merged in clustering, a distance measure on clusters should be defined. As the objective of k-member clustering is to minimize data distortion, it is desired that the data distortion of the merged cluster is as low as possible. For generalization, the data distortion of a cluster is proportional to the distance between two records that are farthest away from each other, and the number of records in the cluster. Traditionally, the distance between clusters C and C' is often defined as

$$d(C, C') = (|C| + |C'|) \cdot max_{qid \in C, qid' \in C'}\{d(qid, qid')\}. \quad (7)$$

Note that the above distance is defined in the sense of the path based distance. In contrast to traditional agglomerative clustering algorithms, a cluster will not be considered for further merging if its size is equal to or larger than k. If no two clusters are of size less than k, the merging stops. Thus, the maximum size of a cluster after merging is $2k-2$.

To minimize data distortion, it is desirable that the sizes of k-member clusters are k or slightly larger than k. But the agglomerative method with (7) will not try to produce such clusters. Instead, the distance in (7) tends to guide the clustering process to merge smaller clusters first. This makes clusters relatively large in later clustering stages. As a result, the clustering process has to merge two large clusters, and the sizes of k-member clusters are much larger than k (but still less than $2k-2$). Thus, the distance defined in (7) is not suitable for the β-cluster based agglomerative clustering algorithm, as the sizes of β-clusters are already relatively large. But in traditional approaches, (7) is suitable for the greedy methods because the size of a cluster is controlled. To mitigate this issue, we propose a flexible distance:

$$d(C, C') = (\theta \cdot |\Delta| + 1) \cdot max_{qid \in C, qid' \in C'}\{d(qid, qid')\}, \quad (8)$$

where $\Delta = |C| + |C'| - k$, and θ, $0 \leq \theta \leq 1$, is a weight for adjusting how much the cluster size or the diameter of a potential merged cluster affects the distance. If two clusters tend to produce a cluster of size k, their distance is smaller.

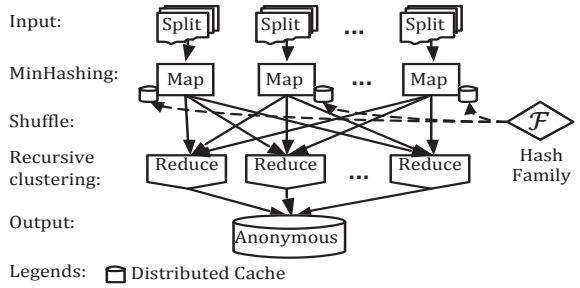


Figure 4: Execution process of LSH based local recoding.

Algorithm 6 describes the details of Beta-AC. A priority queue *PQueue* is used to improve the performance by keeping the distances between any two clusters. In the while-loop, two clusters with the shortest distance are merged in Line 3 and then the priority queue and other information are updated. A possible remaining cluster is handled in Line 11.

Algorithm 6 Beta-AC: β-cluster Agglomerative Clustering.

Input: Small clusters $\mathbb{C}^0$; privacy parameter k.
Output: k-member clusters $\mathbb{C}$; remaining cluster C^r.
1: $\forall C_x^0, C_y^0$, $x \neq y$: $PQueue \leftarrow \langle C_x^0, C_y^0, d(C_x^0, C_y^0)\rangle$;
2: **while** $PQueue$ is not empty **do**
3: $\langle C'_x, C'_y, d(C'_x, C'_y)\rangle \leftarrow PQueue$; $C'_z \leftarrow C'_x \cup C'_y$;
4: $\mathbb{C}^{(i+1)} \leftarrow \mathbb{C}^i \backslash \{C'_x, C'_y\}$;
5: Delete entries involving C'_x or C'_y in $PQueue$;
6: **if** $|C'_z| \geq k$ **then**
7: $\mathbb{C} \leftarrow \mathbb{C} \cup \{C'_z\}$;
8: **else**
9: $\mathbb{C}^{(i+1)} \leftarrow \mathbb{C}^{(i+1)} \cup \{C'_z\}$;
10: $\forall C' \in \mathbb{C}^{(i+1)}$, $PQueue \leftarrow \langle C', C'_z, d(C', C'_z)\rangle$;
11: If $|\mathbb{C}^{(i+1)}| == 1$ and $C'' \in \mathbb{C}^{(i+1)}$, $C^r \leftarrow C''$.

LSH based Local Recoding. This LSH based recursive clustering algorithm is wrapped in a *Reduce* function so that we can achieve MinHash LSH based data partitioning and k-member clustering within a one-pass MapReduce job. To make the *Reduce* function more scalable, a k-member cluster is emitted immediately once generated in Line 7 of Algorithm 6. Finally, the clusters in the resultant set $\mathbb{C}$ are recoded into higher level representations by generalization.

To visually show the proposed approach implemented with MapReduce, Fig. 4 illustrates the execution process overview of LSH-RC. To ensure mappers produce identical bucket IDs, the family of permutation hash functions used in MinHash computation is passed to each *Map* function as a global variable by the distributed cache mechanism . Only the parameters of hash functions, i.e., a, b in $ax + b$, are delivered.

4. EXPERIMENTAL EVALUATION

4.1 Experiment Settings

To evaluate the effectiveness and time-efficiency of the LSH based Recursive Clustering approach (LSH-RC), we compare it with the Greedy k-member Clustering (GC) method proposed in [18], which also represents the approaches in [25, 4]. The GC approach is a state-of-the-art approach for local recoding with clustering techniques. As mentioned in Section 3.1, the metric *ILoss* [24] is employed to measure data distortion. The value of *ILoss* is normalized to facilitate comparisons. So, we gauge *ILoss* and execution time for each group of experiments.

Table 2: Experiment Parameter Settings

Group	θ	α	#(records) k	#(nodes)
1	0.05∼0.5	N/A	1	N/A
2	0.1	2∼4	10∼100	1
3	0.1	2	100∼10,000	10
4	0.1	2	10,000	10∼20

The experiments are conducted on the research cloud platform in our institute. The Hadoop cluster consists of 20 virtual machines of type *m1.medium*, having 2 virtual CPUs and 4 GB Memory. Serial algorithms are executed on one virtual machine of the same type. The maximum heap size of the Java VM is set as 4 GB. All algorithms are implemented in Java and Hadoop MapReduce APIs. The source code can be found at https://dl.dropboxusercontent.com/u/55237646/LSH-RC-master.zip. Each round of experiments is repeated 10 times, with means and standard errors being reported for evaluation.

We use the *Adult* dataset from UCI Machine Learning Repository[1], a publicly available dataset commonly used as a de facto benchmark for testing anonymization algorithms [25, 18, 4, 17, 12, 20]. After pre-processing, the dataset consists of 18,909 records. We utilize nine attributes out of 14 attributes in our experiments. The attribute *Work Class* is utilized as the sensitive attribute. The eight quasi-identifier attributes include both categorical and numerical ones like *Age*. Numerical attributes are discretized according to the technique utilized in [12]. Since we attempt to evaluate the scalability with respect to data volume, the size of the original *Adult* dataset is blown up to generate a series of larger datasets, which is a technique adopted in [20].

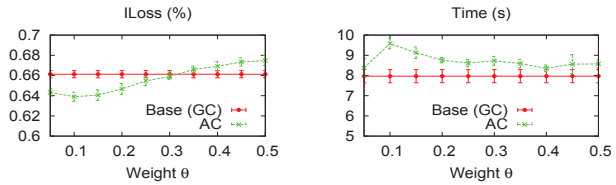
We regard the datasets to be "big" in terms of the number of records, rather than the size, because the former dominates the computational complexity of k-member algorithms. Although the number of attributes can affect the computational complexity, we herein mainly focus on the challenges incurred by the number of data recodes as the dimensionality of microdata is relatively low. To circumvent the problem how big is "big", big data usually means traditional tools fail to handle it within a tolerable elapsed time [23]. The number of records in an enlarged dataset can reach tens of millions, rendering most existing approaches incapable of handling such a dataset. Hence, the enlarged datasets are big enough to evaluate our approach effectively.

4.2 Experiment Process and Results

We conduct four groups of experiments to comprehensively evaluate the techniques proposed in our approach. The k-anonymity parameter is set as 10 throughout all experiments. Other parameters are listed in Table 2.

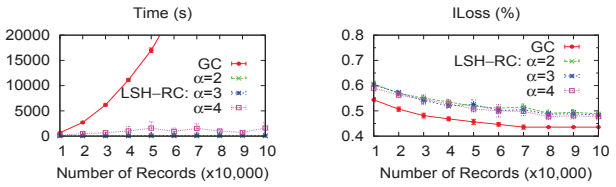
Effectiveness of Flexible Cluster Distance. To evaluate the effectiveness of the flexible cluster distance defined in (8), we implement the serial Agglomerative k-member Clustering (AC) algorithm with the new distance. The greedy method GC with conventional distance in (7) is used as a base for comparison. The number of records (1,000) is large

[1]http://archive.ics.uci.edu/ml/datasets/Adult



(a) (b)

Figure 5: *ILoss* (a) & execution time (b) w.r.t. θ.



(a) (b)

Figure 6: Execution time (a) & *ILoss* (b) w.r.t. #(records).

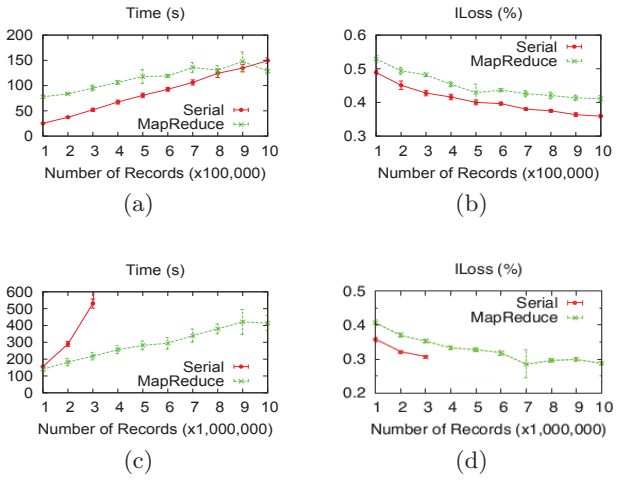
enough as we only aim to evaluate the distances. The results of *ILoss* and time with respect to θ are depicted in Fig. 5.

We can see from Fig. 5(a) that lower information loss can be achieved at some values of the weight than the conventional distance. The lowest *ILoss* is achieved around $\theta = 0.1$. Note that this is equal to $1/k$, where $k = 10$. If $\theta = 1/k$, then $0 \leq \theta \cdot |\Delta| \leq 1$, meaning the size of a potential merged cluster affects less on the distance than the diameter does. The degree increases when $|\Delta|$ grows, with the maximum level equal to that of the diameter. With the analysis above and the experimental results, the weight of θ is suggested to be set as $1/k$ to produce less data distortion. AC's execution time is just slightly higher than GC's from Fig. 5(b).

Comparison with Traditional Approaches. To evaluate the effectiveness and efficiency of the recursive mechanism, we compare it with the greedy method. To conduct a fair comparison, we implement LSH-RC in a serial manner. For the banding parameter $\alpha > 4$, LSH-RC will run out of memory like GC. This is because larger α will make LSH based partitioning produce too many clusters of relatively small size, which approaches to the case of GC. The results with respect to the number of records are described in Fig. 6.

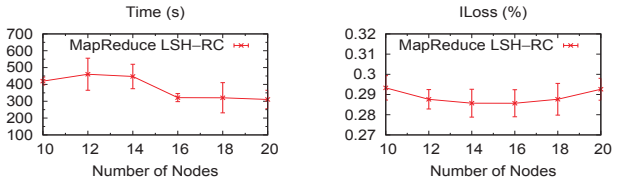
From Fig. 6(a), the execution time of the greedy method soars with increasing data volume, while the recursive clustering method grows only slightly. The difference in execution time between the greedy method and LSH-RC enlarges by orders of magnitude, illustrating that the recursive method outperforms the greedy method significantly. On the other hand, the greedy method produces less data distortion by observing Fig. 6(b). However, this benefit is gained at the cost of prohibitively long execution time. As to α, the execution time increases when α becomes large, while the data distortion decreases correspondingly.

LSH-RC with MapReduce. To evaluate the effectiveness of adopting MapReduce for data partitioning, we compare the performance of serial LSH-RC and MapReduce based LSH-RC. The number of records ranges up to 10,000,000, which is big enough for the evaluation as analyzed before. Fig. 7 shows the results. When the data volume is smaller than 1,000,000, it can be seen from Fig. 7(a) and (b) that serial LSH-RC outperforms MapReduce based LSH-RC for both execution time and data distortion. From Fig. 7(c) and (d), when the data volume exceeds 1,000,000, serial LSH runs slower than MapReduce based LSH-RC, and runs out of memory when the number of records is larger than 3,000,000. However, MapReduce based LSH-RC still goes smoothly with a slight linear increase in execution time as the data volume grows. The data distortion incurred by LSH-RC is higher than serial LSH-RC. However, we can see from the *ILoss* results that the data distortion can degrade satisfactorily as the data volume increases.



(a) (b) (c) (d)

Figure 7: Execution time (a) & (c), and *ILoss* (b) & (d) w.r.t. #(records).



(a) (b)

Figure 8: Execution time (a) & *ILoss* (b) w.r.t. #(records).

Scalability with Respect to Computation Nodes. To further evaluate the scalability of MapReduce based LSH-RC, we measure its performance with respect to the number of Reducers. The number of Reducers ranges from 10 to 20, with 2 as the step length. The results are shown in Fig. 8. We can see from Fig. 8(a) that there is a tendency that the execution time decreases with the growth of the number of Reducers. This is reasonable because if more Reducers are employed, the number of β-clusters processed in each Reducer will diminish correspondingly. The data distortion remains nearly at the same level in terms of Fig. 8(b). Thus, it is an effective way for our approach to address the scalability problem by adding more computing nodes, without substantially affecting data utility.

Above all, the results from the four groups of experiments

show that our approach significantly improves the scalability and time-efficiency of local recoding by orders of magnitude over existing approaches while ensuring data utility.

5. CONCLUSTIONS AND FUTURE WORK

In this paper, we have investigated the scalability problem of local-recoding anonymization for big data privacy preservation, and proposed a highly scalable approach to local-recoding anonymization based on Locality Sensitive Hashing (LSH). Specifically, we have proposed a novel distance measure called provenance set based semantic distance for the semantic distance between two data records, to make LSH applicable to the semantic distance. Then, the MinHash LSH family has been implemented with MapReduce to split the original data into small partitions that contain similar data records. Finally, a highly efficient LSH based recursive algorithm of agglomerative clustering is proposed to conduct k-member clustering. This algorithm is implemented with MapReduce and executed on the data partitions in parallel for high scalability. Experiments have demonstrated that our approach can significantly improve the scalability and time-efficiency of local recoding by orders of magnitude over existing approaches. Privacy preservation for analyzing, sharing and mining big data is a challenging research area. Based on the contributions herein, we plan to integrate our approach with present scalable machine learning and data mining platforms, aiming to achieve scalable privacy preserving big data mining or analytics.

6. ACKNOWLEDGMENTS

This paper is partially supported by the National Natural Science Foundation of China under Grant No. 91318301, the Key Research and Development Project of Jiangsu Province under Grant No. BE2015154.

7. REFERENCES

[1] G. Aggarwal, R. Panigrahy, T. Feder, D. Thomas, K. Kenthapadi, S. Khuller, and A. Zhu. Achieving anonymity via clustering. *ACM Transactions on Algorithms*, 6(3):49, 2010.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proc. FOCS'06*, pages 459–468, 2006.

[3] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *Proc. STOC'98*, pages 327–336, 1998.

[4] J.-W. Byun, A. Kamra, E. Bertino, and N. Li. Efficient k-anonymization using clustering techniques. In *Proc. DASFAA'07*, pages 188–200. 2007.

[5] S. Chaudhuri. What next?: A half-dozen data management research goals for big data and the cloud. In *Proc. PODS'12*, pages 1–4, 2012.

[6] C. Clifton and T. Tassa. On syntactic anonymity and differential privacy. *Trans. Data Privacy*, 6(2):161–183, 2013.

[7] J. Dean and S. Ghemawat. Mapreduce: a flexible data processing tool. *Comm. ACM*, 53(1):72–77, 2010.

[8] C. Dwork. Differential privacy. In *Proc. ICALP'06*, pages 1–12, 2006.

[9] A. Ene, S. Im, and B. Moseley. Fast clustering using mapreduce. In *SIGKDD'11*, pages 681–689, 2011.

[10] R. L. Ferreira Cordeiro, C. Traina Junior, A. J. Machado Traina, J. López, U. Kang, and C. Faloutsos. Clustering very large multi-dimensional datasets with mapreduce. In *SIGKDD'11*, pages 690–698, 2011.

[11] B. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14, 2010.

[12] B. C. Fung, K. Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *IEEE TKDE*, 19(5):711–725, 2007.

[13] G. Ghinita, P. Kalnis, and Y. Tao. Anonymous publication of sensitive transactional data. *IEEE TKDE*, 23(2):161–174, 2011.

[14] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC'98*, pages 604–613, 1998.

[15] T. Iwuchukwu and J. F. Naughton. K-anonymization as spatial indexing: Toward scalable and incremental anonymization. In *VLDB'07*, pages 746–757, 2007.

[16] H. Koga, T. Ishibashi, and T. Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowl. Inf. Systm.*, 12(1):25–53, 2007.

[17] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization techniques for large-scale datasets. *ACM TODS*, 33(3):17, 2008.

[18] J. Li, R.-W. Wong, A.-C. Fu, and J. Pei. Anonymization by local recoding in data with attribute hierarchical taxonomies. *IEEE TKDE*, 20(9):1181–1194, 2008.

[19] N. Li, T. Li, and S. Venkatasubramanian. Closeness: A new privacy measure for data publishing. *IEEE TKDE*, 22(7):943–956, 2010.

[20] N. Mohammed, B. Fung, P. C. Hung, and C.-K. Lee. Centralized and distributed anonymization for high-dimensional healthcare data. *ACM TKDD*, 4(4):18, 2010.

[21] J. Soria-Comas, J. Domingo-Ferrer, D. Sánchez, and S. Martínez. Enhancing data utility in differential privacy via microaggregation-based k-anonymity. *The VLDB Journal*, 23(5):771–794, 2014.

[22] L. Sweeney. k-anonymity: A model for protecting privacy. *Int'l J. Uncertain. Fuzz.*, 10(05):557–570, 2002.

[23] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding. Data mining with big data. *IEEE TKDE*, 26(1):97–107, 2014.

[24] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proc. SIGMOD'06*, pages 229–240, 2006.

[25] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *SIGKDD'06*, pages 785–790, 2006.

[26] X. Zhang, W. Dou, J. Pei, S. Nepal, C. Yang, C. Liu, and J. Chen. Proximity-aware local-recoding anonymization with mapreduce for scalable big data privacy preservation in cloud. *IEEE TC*, 64(8):2293–2307, 2015.

[27] X. Zhang, L. T. Yang, C. Liu, and J. Chen. A scalable two-phase top-down specialization approach for data anonymization using mapreduce on cloud. *IEEE TPDS*, 25(2):363–373, 2014.