IEICE TRANS. INF. & SYST., VOL.E99–D, NO.4 APRIL 2016

PAPER Special Section on Data Engineering and Information Management

A Scheme for Fast k-Concealment Anonymization

Ryosuke KOYANAGI[†a)], **Ryo FURUKAWA**[††b)], ***Nonmembers***, **Tsubasa TAKAHASHI**[††c)], ***Member***, **Takuya MORI**[††d)], ***Nonmember***, **Toshiyuki AMAGASA**[†††e)], ***and*** **Hiroyuki KITAGAWA**[†††f)], ***Members***

SUMMARY In this paper we propose an improved algorithm for k-concealment, which has been proposed as an alternative to the well-known k-anonymity model. k-concealment achieves similar privacy goals as k-anonymity; it proposes to generalize records in a table in such a way that each record is indistinguishable from at least $k-1$ other records, while achieving higher utility than k-anonymity. However, its computation is quite expensive in particular when dealing with large datasets containing massive records due to its high computational complexity. To cope with this problem, we propose neighbor lists, where for each record similar records are stored. Neighbor lists are constructed in advance, and can also be efficiently constructed by mapping each record to a point in a high-dimensional space and using appropriate multidimensional indexes. Our proposed scheme successfully decreases the execution time from $O(kn^2)$ to $O(k^2n+knlogn)$, and it can be practically applied to databases with millions of records. The experimental evaluation using a real dataset reveals that the proposed scheme can achieve the same level of utility as k-concealment while maintaining the efficiency at the same time.

key words: *k-concealment, privacy, data publishing*

1. Introduction

Many organizations regularly collect massive amount of data, and there are strong demands of sharing these data with others, e.g., researchers, companies, citizens, etc., for academic, commercial, or public reasons. Such data often contain personal information about individuals. Therefore, they need to be processed properly in prior to their secondary use so that no sensitive information is released.

To publish data containing personal information, *data anonymization* [1] is commonly used to transform data from its original form into more generalized forms to prevent leakage of private information. Among others, k-anonymity [2] is one of the most well-known models. Basically, a dataset is said to have k-anonymity property if each record is identical to at least $k-1$ other records. To this end, the *quasi-identifiers* of records are gradually generalized or suppressed until the k-anonymity property is satisfied. Notice that generalization/suppression incurs loss of information to the dataset. Therefore, it is quite important to minimize *information loss* in order to prevent degradation of utility. Thanks to its simplicity and generality, k-anonymity and its variants have been extensively studied [2]–[5] and have been widely used [6], [7].

As an alternative to k-anonymity, Tassa et al. [8] have proposed k-concealment model. The basic motivation for k-concealment model is that it is possible to improve utility of anonymized data without sacrificing the privacy level in a practical sense. More precisely, k-concealment achieves similar privacy goals as k-anonymity; it proposes to generalize records in a table in such a way that each record is indistinguishable from at least $k-1$ other records, while achieving higher utility than k-anonymity.

We chose k-concealment in this work due to the advantage that it provides better utility than k-anonymity in the anonymized data; i.e., data anonymized by k-concealment retain more information than those anonymized by k-anonymity, which would be beneficial in many real applications. However, one of the major drawbacks is that it is computationally expensive than k-anonymization. For example, the time complexity of one of the fastest algorithms for k-anonymity [9] is shown to be $O(n \log n)$, whereas the overall time complexity of the algorithm proposed in [8] is $O(kn^2)$ where n is the number of records. This is too expensive particularly when dealing with large datasets containing large number of records. Therefore, to make k-concealment more practical in many real applications in the Big Data era, more efficient algorithms are required.

In this paper we propose a scheme for faster k-concealment anonymization. The main idea is to precompute neighbor lists (NLs) for each record. More precisely, we assume each quasi-indentifier is of totally ordered numerical domain, like integer or real number*. In the proposed scheme, first, we construct NLs by: 1) mapping each record to a point in a multidimensional space, 2) construct a multidimensional index (kd-tree) [10] for the mapped records, and 3) for each record, find s-nearest point where s is a user-defined parameter such that $s \geq k-1$. The

Manuscript received July 1, 2015.
Manuscript revised November 9, 2015.
Manuscript publicized January 14, 2016.

†The author is with Department of Computer Science, Graduate School of Systems and Information Enginnering, University of Tsukuba, Tsukuba-shi, 305–8571 Japan.
††The authors are with Cloud System Research Laboratories, NEC Corporation, Kawasaki-shi, 211–8666 Japan.
†††The authors are with Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba-shi, 305–8571 Japan.
a) E-mail: rkoya@kde.cs.tsukuba.ac.jp
b) E-mail: r-furukawa@cb.jp.nec.com
c) E-mail: t-takahashi@nk.jp.nec.com
d) E-mail: moritaku@bx.jp.nec.com
e) E-mail: amagasa@cs.tsukuba.ac.jp
f) E-mail: kitagawa@cs.tsukuba.ac.jp
DOI: 10.1587/transinf.2015DAP0030

*We can deal also with categorical domains by mapping them to numerical values.

Copyright © 2016 The Institute of Electronics, Information and Communication Engineers

algorithms for k-concealment have been modified so that they search records only within NLs, thereby making it possible to avoid exhaustive pairwise computations of loss metric. This causes significant reduction of the execution time from $O(kn^2)$ to $O(k^2n+knlogn)$, In general, parameter k is far smaller than the number records (n). Consequently, the proposed algorithm is considered to be useful in many applications where large datasets need to be anonymized. We conduct several experimental evaluations using a real dataset to assess the feasibility and performance of the proposed scheme. The experimental results reveal that the proposed scheme can achieve same level of utility as k-concealment while drastically reducing execution time.

The rest of this paper is organized as follows. Section 2 introduces some basic definitions including k-type anonymization and k-concealment, which is the target of our proposal. The existing approach for k-concealment is reviewed in Sect. 3. The proposed scheme is then introduced in Sect. 4, followed by an experimental evaluation in Sect. 5. We conclude this paper and mention future work in Sect. 6.

2. Basic Definitions

In this section we introduce some basic definitions which are necessary to define k-concealment model, followed by introducing k-concealment model. We also mention information loss function to measure loss of information incurred by data anonymization. All definitions are basically based on Tessa et al. [8] but some notations are modified for brevity.

2.1 Generalized Database

We start our discussion by defining databases and their generalized forms. We deal with an original relational database R consisting of n records, where each record has m public attributes (a.k.a., *quasi-identifiers*) and additional private attributes. We denote by A_j *attribute set* which is a set of possible values of the j-th attribute, thereby constituting a table of size n by m. We denote by $R_{i,j} (\in A_j)$ the j-th attribute of the i-th record and also denote by $R_i (\in A_1 \times \ldots \times A_m)$ the i-th record. We denote by $\overline{A}_j$ the power set of A_j. We then define a generalized database $\overline{R} = \{\overline{R}_1, \ldots, \overline{R}_n\}$ such that $\overline{R}_i \in \overline{A}_1 \times \ldots \times \overline{A}_m$. Given two records R_i and $\overline{R}_j$, we say R_i (or $\overline{R}_j$) is *consistent* with $\overline{R}_j$ (or R_i) if $R_{i,l} \in \overline{R}_{j,l}$ holds for $l = 1, 2, \ldots, m$, and is denoted by $R_i \prec \overline{R}_j$.

2.2 k-Type Anonymity Model

Having defined generalized databases, we review well-known k-anonymity, and then mention k-type anonymity models proposed by Tessa et al. [8].

A k-anonymization of a database R is a generalization of R, where, for all i, there exist at least k generalized records $\overline{R}_i$:

$$\forall i,\ \left|\{\overline{R}_j \mid \overline{R}_j = \overline{R}_i\}\right| \geq k.$$

We denote by $\mathcal{A}^k_R$ k-anonymization of database R.

As alternatives of k-anonymity, Tessa et al. proposed k-type anonymity models [8], which are relaxations of k-anonymity for improved utility.

$(1,k)$-anonymity A generalization of database R, $\overline{R}$, is called a $(1,k)$-anonymization of R if each record in R is consistent with at least k records in $\overline{R}$:

$$\forall i,\ \left|\{\overline{R}_j \mid R_i \prec \overline{R}_j\}\right| \geq k.$$

$(k,1)$-anonymity $\overline{R}$ is called a $(k,1)$-anonymization of R if each record in $\overline{R}$ is consistent with at least k records in R:

$$\forall j,\ \left|\{R_i \mid R_i \prec \overline{R}_j\}\right| \geq k.$$

(k,k)-anonymity $\overline{R}$ is called a (k,k)-anonymization of R if it is both a $(1,k)$- and a $(k,1)$-anonymization of R:

$$\forall i,\ \left|\{\overline{R}_j \mid R_i \prec \overline{R}_j\}\right| \geq k \text{ and } \forall j,\ \left|\{R_i \mid R_i \prec \overline{R}_j\}\right| \geq k$$

We denote by $\mathcal{A}^{1,k}_R$, $\mathcal{A}^{k,1}_R$, and $\mathcal{A}^{k,k}_R$ $(1,k)$-, $(k,1)$-, and (k,k)-anonymization of database R, respectively. Note here that the following interrelationship has been proven in [8] stating that k-anonymity is the most strict, followed by (k,k)-anonymity; $(1,k)$- and $(k,1)$-anonymity models are the most relaxed ones:

$$\mathcal{A}^k_R \subset \mathcal{A}^{k,k}_R \subset \mathcal{A}^{k,1}_R, \mathcal{A}^{1,k}_R,$$

$$\mathcal{A}^{1,k}_R \setminus \mathcal{A}^{k,1}_R \neq \emptyset,\ \mathcal{A}^{k,1}_R \setminus \mathcal{A}^{1,k}_R \neq \emptyset.$$

2.3 Bipartite Graph Representation

For better understanding, we introduce bipartite graph representation of k-anonymization and k-type anonymization models. Let us assume a bipartite graph $G(R,\overline{R}) = (U(R), V(\overline{R}), E(R,\overline{R}))$, where $U(R)$ and $V(\overline{R})$ are sets of vertexes corresponding to R and $\overline{R}$, respectively; and $E(R,\overline{R})$ is a set of edges that connect $u_i \in U(R)$ and $v_j \in V(\overline{R})$ such that an edge (u_i, v_j) exists if and only if R_i is consistent with $\overline{R}_j$. Additionally, we denote by $deg(u_i)$ the degree of $u_i \in R_i$ defined as $|\{j \mid R_i \prec \overline{R}_j\}|$. Likewise, we denote by $deg(v_j)$ the degree of $v_j \in R_j$ defined as $|\{i \mid R_i \prec \overline{R}_j\}|$.

Using the bipartite graph representation, we can understand k-anonymity and k-type anonymity models as follows. For better understating, we show illustrative examples. The original database (Fig. 1) has three attributes, namely, Name, Age, and Zipcode, and a private attribute Disease, and it is 2-anonymized.

k-anonymity $\mathcal{A}^k_R$: G consists of some complete bipartite graphs that contain at least $k + k$ vertexes (Fig. 2).

$(1,k)$-anonymity $\mathcal{A}^{1,k}_R$: $deg(u_i) \geq k$ for all $u_i \in U(R)$ (Fig. 3).

$(k,1)$-anonymity $\mathcal{A}^{k,1}_R$: $deg(v_j) \geq k$ for all $v_j \in V(R)$ (Fig. 4).

(k,k)-anonymity $\mathcal{A}^{k,k}_R$: $deg(u_i) \geq k$ and $deg(v_j) \geq k$ for all $u_i \in U(R)$ and $v_j \in V(R)$ (Fig. 5).

Name	Age	Zipcode	* Disease
Alice	30	1055	Measles
Bob	21	1055	Flu
Carol	21	1123	Angina
David	20	1155	Flu
Eve	18	1123	Diabetes

Fig. 1 Original database.

Name	Age	Zipcode	Age	Zipcode	* Disease
Alice	30	1055	21-30	1055-1123	Measles
Bob	21	1055			Flu
Carol	21	1123			Angina
David	20	1155	18-20	1123-1155	Flu
Eve	18	1123			Diabetes

Fig. 2 k-Anonymity.

Name	Age	Zipcode	Age	Zipcode	* Disease
Alice	30	1055	30	1055	Measles
Bob	21	1055	21-30	1055-1123	Flu
Carol	21	1123	18-21	1055-1155	Angina
David	20	1155	20	1155	Flu
Eve	18	1123	18	1123	Diabetes

Fig. 3 $(1,k)$-Anonymity.

Name	Age	Zipcode	Age	Zipcode	* Disease
Alice	30	1055	21-30	1055	Measles
Bob	21	1055	21	1055-1123	Flu
Carol	21	1123	21	1055-1123	Angina
David	20	1155	20-21	1123-1155	Flu
Eve	18	1123	18-20	1123-1155	Diabetes

Fig. 4 $(k,1)$-Anonymity.

Name	Age	Zipcode	Age	Zipcode	* Disease
Alice	30	1055	21-30	1055	Measles
Bob	21	1055	21-30	1055	Flu
Carol	21	1123	21	1055-1123	Angina
David	20	1155	18-21	1123-1155	Flu
Eve	18	1123	18-20	1123-1155	Diabetes

Fig. 5 (k,k)-Anonymity.

2.4 k-Concealment

Having introduced k-anonymity and k-type anonymity models, we are ready to define k-concealment model. In fact, k-concealment is a stronger version of $(1,k)$-anonymity[†]; while $(1,k)$-anonymity requires each record (or node) $R \in D$ to have at least k adjacent edges, k-concealment additionally requires that each record $R \in D$ has at least k adjacent (bipartite) matches.

More precisely, we introduce *possible* and *impossible* links. We consider that an edge $e = (u_i, v_j)$ is a possible link if G has at least one complete (bipartite) matching that contains e, whereas it is an impossible link if it does not contribute to any complete matching.

In the bipartite graph representation, k-concealment (denoted by C_R^k) can be defined as follows:

[†]In [8], the authors mention that similar stronger versions of $(k,1)$- and (k,k)-anonymity can also be possible. However, they chose $(1,k)$-anonymity as the basis by taking into account adversary's possible preference.

Name	Age	Zipcode	Age	Zipcode	* Disease
Alice	30	1055	21-30	1055	Measles
Bob	21	1055	21-30	1055	Flu
Carol	21	1123	20-21	1123-1155	Angina
David	20	1155	18-21	1123-1155	Flu
Eve	18	1123	18-20	1123-1155	Diabetes

Fig. 6 k-Concealment.

Table 1 LM cost for Fig. 1

Age (cost)	Zipcode (cost)
21-30 (0.75)	1055 (0.00)
21-30 (0.75)	1055 (0.00)
20-21 (0.08)	1123-1155 (0.32)
18-21 (0.25)	1123-1155 (0.32)
18-20 (0.17)	1123-1155 (0.32)

k-concealment C_R^k: every vertex is of least degree k even though all impossible links are removed (Fig. 6).

Note here that the following interrelationship including k-concealment has been shown in [8] stating that k-concealment is more relaxed than k-anonymity, but is more strict than $(1,k)$-anonymity; i.e.:

$$\mathcal{A}_R^k \subset C_R^k \subset \mathcal{A}_R^{1,k},$$

Consequently, given a database, its anonymization by k-concealment can deliver better utility than its k-anonymization.

2.5 Measuring Information Loss

In data anonymization it is important to minimize loss of information incurred by generalization of data. To quantify information loss, in [8] Loss Metric (LM) [11], [12] (denoted by $\prod(R,\overline{R})$) has been used:

$$\prod(R,\overline{R}) = \frac{1}{n}\sum_{i}^{n} c(\overline{R}_i)$$

where $c(\overline{R}_i)$ is a commonly used LM function defined as:

$$c(\overline{R}_i) = \frac{1}{m}\sum_{j}^{m} \frac{|\overline{R}_{i,j}| - 1}{|A_j| - 1}$$

The intuition behind LM is to compute the average coverage ratio of all attributes.

Table 1 shows an example of LM calculated by the original database R (Fig. 1) and its anonymization by k-concealment (Fig. 6). In this database, attribute Age is of an integer domain whose range is $[18, 30]$, which comprises 13 distinct values. Taking the first record whose age is $[21, 30]$ for example, it includes 10 distinct values. Therefore, we can calculate the cost for this record by $\frac{|\overline{R}_{1,1}|-1}{|A_1|-1} = \frac{10-1}{13-1} = 0.75$. Consequently, we get $\prod(R,\overline{R}) = (0.75+0.00+0.75+0.00+0.08+0.32+0.25+0.32+0.17+0.32)/(2*5) = 0.496$.

3. Existing Algorithm

In this section we review algorithms for k-concealment

Algorithm 1 $(k, 1)$-Anonymization

Require: Database R, integer k
Ensure: Database $\overline{R}$ that satisfies $(k, 1)$-anonymity
1: **for** $i = 1 \ \ldots \ n$ **do**
2: $\quad \overline{R}_i \Leftarrow \{R_i\}$
3: $\quad$ **while** $|\{R_j \mid R_j \prec \overline{R}_i\}| < k$ **do**
4: $\quad\quad$ Find record R_j that minimizes $dist(\overline{R}_i, R_j)$.
5: $\quad\quad \overline{R}_i \Leftarrow \overline{R}_i \cup R_j$
6: $\quad$ **end while**
7: **end for**

anonymization proposed in [8]. The whole process proceeds in the following steps: 1) transform from the original database R to $\mathcal{A}_R^{k,1}$; 2) transform from $\mathcal{A}_R^{k,1}$ to $\mathcal{A}_R^{k,k}$; and 3) transform from $\mathcal{A}_R^{k,k}$ to C_R^k.

3.1 Algorithm for $(k, 1)$-Anonymization

The first step is to apply $(k, 1)$-anonymization to the original database, and the algorithm is shown in Algorithm 1, which is a greedy version of the corresponding algorithm in [8]. It tries to find for each $R_i \in R$ other $k-1$ records in R such that the generalization cost is minimized. Notice that the cost for generalization is calculated by:

$$dist(\overline{R}_i, R_j) = c(\overline{R}_i \cup R_j) - c(\overline{R}_i)$$

Notice that generalization is denoted by $\cup$. A generalization $\overline{R}_i$ with R_j is represented by $\overline{R}_i \cup R_j$. After finding k records for R_i, they are generalized to form $\overline{R}_i$. In fact, this algorithm does not always find the optimal combination in terms of cost, because it greedily finds records to generalize one by one. However, it was proven in [8] that it offers an approximation ratio guarantee of $k - 1$ with respect to LM $\prod$.

3.2 Algorithm for $(1, k)$-Anonymization

The next step is to apply $(1, k)$-anonymization to $(k, 1)$-anonymized data for transforming it to (k, k)-anonymized data. The algorithm is shown in Algorithm 2. For each record $R_i \in R$ that are consistent with less than k records in $\overline{R}$, the algorithm scans records $\overline{R}_j \in \overline{R}$ that are not consistent with R_i and finds $k - l$ records with less generalization costs. Then it replace those $k - l$ records $\overline{R}_j$ with $R_i \cup \overline{R}_j$. This algorithm also does not always find optimal combination and there is no guarantee.

3.3 Algorithm for k-Concealment

As is discussed in Sect. 2.4, k-concealment can be understood as complete matching in bipartite graph model. First, we enumerate possible links. Let us assume a bipartite graph corresponding to a database R and its generalization $\overline{R}$ $G(R, \overline{R}) = (U, V, E)$, where $U = \{u_1, \ldots, u_n\}$, $V = \{v_1, \ldots, v_n\}$, and $E \subseteq U \times V$ are sets of vertexes corresponding to R and $\overline{R}$ and set of edges respectively. Then we generate from G a new directed graph $H(U, F)$, where

Algorithm 2 $(1, k)$-Anonymization

Require: Databases $R, \overline{R}$, integer k
Ensure: Database $\overline{R'}$ that satisfies $(1, k)$-anonymity
1: **for** $i = 1 \ \ldots \ n$ **do**
2: $\quad l \Leftarrow |\{\overline{R}_j \mid R_i \prec \overline{R}_j\}|$
3: $\quad$ **if** $l < k$ **then**
4: $\quad\quad$ Find $k - l$ records in $\overline{R}$ such that $\{\overline{R}_j \mid R_i \not\prec \overline{R}_j\}$ that have minimum generalization cost $dist(\overline{R}_j, R_i)$.
5: $\quad\quad \overline{R}_j \Leftarrow \overline{R}_j \cup R_i$
6: $\quad$ **end if**
7: **end for**

Algorithm 3 Check possibilities of links.

Require: Bipartite graph $G(U, V, E)$
Ensure: Possibilities of links E
1: Direct graph $H \Leftarrow (U, F)$ with $F = \{(u_i, u_j) \mid (u_i, v_j) \in E\}$.
2: Find all strongly connected components of H.
3: **for** $(u_i, v_j) \in E$ **do**
4: $\quad (u_i, v_j)$ is possible link if u_i and v_i belong to the same strongly connected component in H.
5: **end for**

Algorithm 4 k-Concealment

Require: Database $R, \overline{R} \in \mathcal{A}_R^{k,k}$, integer k
Ensure: Database $\overline{R}$ that satisfies k-concealment
1: check possible links
2: **for** $i = 1 \ \ldots \ n$ **do**
3: $\quad Q \Leftarrow \{j \mid R_i \prec \overline{R}_j\}$
4: $\quad P \Leftarrow$ subset of Q that connected to R_i by a possible link.
5: $\quad$ **if** $k \leq |P|$ **then**
6: $\quad\quad$ Goto next i
7: $\quad$ **end if**
8: $\quad j \Leftarrow \arg\min_{j \in Q \setminus P}(dist(\overline{R}_i, R_j))$
9: $\quad \overline{R}_i \Leftarrow \overline{R}_i \cup R_j$
10: $\quad$ Recheck possible links
11: $\quad$ Return to step 3
12: **end for**

$U = \{u_1, \ldots, u_n\}$ and $F = \{(u_i, u_j) \mid (u_i, v_j) \in E\}$ are sets of nodes and edges, respectively. In fact, an edge (u_i, v_i) in G is a possible link if and only if it is a part of any cycle in H.

Algorithm 3 is to check the possibility of each edge. We generate H and enumerate strongly connected components for subsequent possibility checking; i.e., the possibility of an edge (u_i, v_j) can be identified by checking whether u_i and v_j belong to one or more strongly connected components (SCCs) or not. We can use any algorithm for SCC decomposition, e.g., Tarjan [13] or Cheriyan and Mehlhorn [14]. Both algorithms are linear in time.

Algorithm 4 shows an algorithm for k-concealment. If a degree of vertex u_i is less than k by ignoring an impossible link (u_i, u_j), we make (u_i, v_j) a possible link by adding a new edge (u_j, v_i). If there are some choices of impossible links, we choose the best one greedily according to loss metric. Again, this algorithm does not offer any optimality.

3.4 Time Complexity and Problem

The overall complexity of the above algorithms for k-concealment is shown to be $O(kn^2)$. It is not very bad, but when dealing with large datasets, e.g., more than 10,000 records, it becomes impractical to apply k-concealment. For this reason, we need more efficient algorithms to make k-concealment practical to large datasets.

4. Proposed Scheme

4.1 Basic Idea and Assumptions

In this section we present the proposed scheme for efficient k-concealment anonymization. The basic idea of our scheme is to pre-calculate *neighbor lists (NLs)* for each record, and utilize them when searching such records that can be generalized at low costs.

First, we describe the basic assumptions. We assume that each attribute A_i $(i = 1, \ldots, m)$ is of a totally-ordered numerical type, such as integer, real number, etc. In addition, we assume for each attribute there are the upper and the lower bounds. Thus we can map each record to a point in normalized m-dimensional space $[0, 1]^m$. Notice that other data types, such as strings and categorical attributes, can also be handled by mapping such values to those in numerical domain. For example, let us consider a nominal or ordinal categorical attribute with values $\{A, B, C\}$. In this case we take an integer domain $[0, 2]$ and map each value in the same order, i.e., $A \to 0$, $B \to 1$, and $C \to 2$. Notice that such mappings impose total order in the mapped domain even though there is no explicit ordering in the original domain. However, in practice, this is not a big problem, there is no ordering even in the original domain.

We show an example of normalized database and its 2-D plot in Fig. 7.

To quantify the cost for generalization, we use Loss Metric (LM) as mentioned above.

4.2 Neighbor List Construction

First, we construct for each record a NL to facilitate subsequent greedy-based generalization of records. More precisely, neighbor list NL_i for record $R_i \in R$ contains top s records that have low generalization cost in LM w.r.t. R_i, where s is a user-specified parameter such that $s \geq k - 1$. To construct NLs, we need to compute pair-wise generalization costs in LM between two different records, which require $O(n^2)$ in time.

To alleviate this problem we exploit an approximated cost function by taking advantage of the feature that each record can be represnted as a point in $[0, 1]^m$. More precisely, we quantify generalization cost in terms of L_1-distance in $[0, 1]^m$:

$$dist'(R_i, R_j) = \sum_{k}^{m} \frac{\left|R_{i,k} - R_{j,k}\right|}{|A_k|}$$

In addition, we exploit a multidimensional index, kd-tree [10], to retrieve records, resulting in $O(kn \log n)$ time complexity for constructing NLs.

Name	Age	Zipcode	* Disease
Alice	1.00	0.00	Measles
Bob	0.25	0.00	Flu
Carol	0.25	0.68	Angina
David	0.17	1.00	Flu
Eve	0.00	0.68	Diabetes

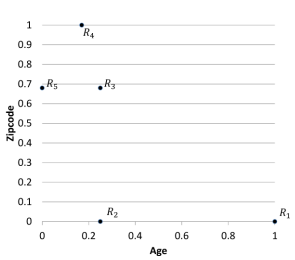


Fig. 7 Normalized database and 2-D plot.

4.3 Algorithms for $(k, 1)$-Anonymization

To speedup $(k, 1)$-anonymization, we exploit NLs to avoid exhaustive search over records. Before explaining the idea, we introduce hyperrectangle-based representation of generalized records; i.e., we can regard a generalized record $\overline{R}_i$ as a minimum bounding m-dimensional hyperrectangle that contains all records consistent with $\overline{R}_i$. Consequently, the distance (generalization cost) $dist(\overline{R}_i, R_j)$ is equivalent to distance between $\overline{R}_i$ and R_j. Here we introduce an approximation of $dist(\overline{R}_i, R_j)$:

$$dist(\overline{R}_i, R_j) \approx min_{R_k \in \overline{R}_i} dist'(R_k, R_j)$$

In this approximation, instead of calculating the real distance between R_j and $\overline{R}_i$, we identify the closest record in $\overline{R}_i$ from R_j and use its distance as dist$(\overline{R}_i, R_j)$.

Algorithm 5 shows an algorithm for $(k, 1)$-anonymization using NLs. Differences from Algorithm 1 lie in Line 3, 5, and 7. For each record R_i, we maintain candidate set of records C and initialize it by the records in NL_i. For each record $R_j \in C$, we compute the cost by LM to find the one with least generalization cost. When we find it, we generalize R_i with the record, and merge NL_j with C to populate new candidates. Notice that we maintain the number of candidates in C at most $2s$ according to previously computed LM values.

Algorithm 5 Fast $(k, 1)$-Anonymization using NL

Require: Database R, integer k, Neighbor Lists $NL_1, NL_2, \ldots, NL_n$
Ensure: Database $\overline{R}$ that satisfies $(k, 1)$-anonymity
1: **for** $i = 1 \ \ldots \ n$ **do**
2: $\quad \overline{R}_i \Leftarrow \{R_i\}$
3: $\quad C \Leftarrow NL_i$
4: $\quad$ **while** $|\{R_j \mid R_j \prec \overline{R}_i\}| < k$ **do**
5: $\quad\quad R_j \Leftarrow \arg\min_{R_j \in C}(dist(\overline{R}_i, R_j))$
6: $\quad\quad \overline{R}_i \Leftarrow \overline{R}_i \cup R_j$
7: $\quad\quad C \Leftarrow (C \cup NL_j) \setminus \{R_j \mid R_j \prec \overline{R}_i\}$
8: $\quad\quad$ erase elements from C except s-elements with a little cost.
9: $\quad$ **end while**
10: **end for**

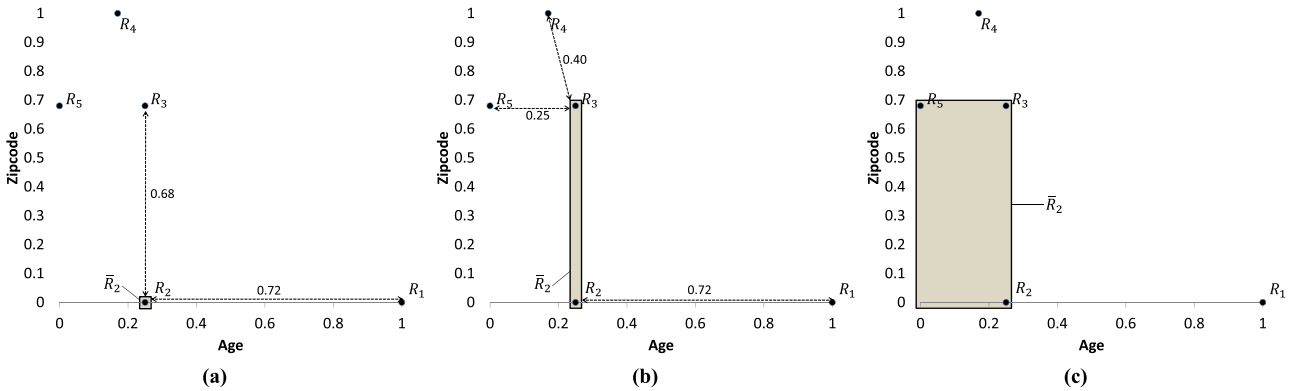


Fig. 8 Example of $(k, 1)$-anonymization using NL.

Figure 8 illustrates a running example of $i = 2$ for a database Fig. 1 where $k = 3$. Each point and rectangle represent a record and a generalized record, respectively. (a) Initially, we have $NL_1 = \{2, 3\}, NL_2 = \{1, 3\}$ $NL_3 = \{4, 5\}, NL_4 = \{3, 5\}, NL_5 = \{3, 4\}$, and $C = \{1, 3\}$. Then we choose R_3, because it has the minimum cost to generalize $\overline{R}_2$. (b) Consequently, we generalize $\overline{R}_2$ by R_3 so that it subsumes R_2 and R_3, and populate C by NL_3, accordingly. Now, we need to reduce $C = \{1, 4, 5\}$ by choosing records with less cost, resulting $C = \{4, 5\}$. (c) Likewise, we choose R_5 and generalize $\overline{R}_2$ with it.

As can be seen, the proposed algorithm improves the processing performance by avoiding exhaustive pair-wise cost calculations. Instead, we only calculate costs over records populated from NLs in candidate set C. As a result the complexity of this step is decreased to $O(kn \log k)$ in time.

4.4 Algorithm for $(1, k)$-Anonymization

In this algorithm we need to find, for each record R_i such that $deg(u_i) < k$, additional $k - deg(u_i)$ generalized records $\overline{R}_j$ that minimize $dist(\overline{R}_j, R_i)$ for ensuring $(1, k)$-anonymity. To speedup this process, we again utilize NLs; i.e., we only search for records within NL_i under the assumption that NL_i likely contains records that are close to R_i, thereby avoiding costly exhaustive search of records.

Algorithm 6 shows an algorithm using neighbor lists for faster $(1, k)$-anonymization. Differences from Algorithm 2 lie in Lines 3 and 4. In many cases we can expect that $deg(u_j)$ is close to k. Thus we do not need to pay much efforts for finding additional generalized records, resulting in $O(k^2 n)$ of time complexity.

However, there may be some undesirable cases, e.g., a record is consistent with all generalized records and the corresponding neighbor list is filled up with them, etc. In such cases the time complexity will become $O(n^2)$. Such situations can be characterized by biased degree distribution in the bipartite graph model. Worse than that, biased degree distribution leads to more information to be lost; i.e., we need to add more edges to ensure (k, k)-anonymization, and this means that the data needs to be more and more generalized. For this reason it is important to avoid extremely biased degree distributions. One possible workaround is to take into account degree distribution in the cost function, which is a part of our future work.

Algorithm 6 Fast $(1, k)$-Anonymization using NL

Require: Database R, integer k, Neighbor list $NL_1, NL_2, \ldots, NL_i$.
Ensure: Database $\overline{R}$ that satisfies $(1, k)$-anonymity
1: **for** $i = 1 \ldots n$ **do**
2: $l \Leftarrow |\{\overline{R}_j \mid R_i \prec \overline{R}_j\}|$
3: **if** $l < k$ **then**
4: $C \Leftarrow \{\overline{R}_j \mid R_k \in NL_i \wedge R_k \prec \overline{R}_j\} \setminus \{\overline{R}_j \mid R_i \prec \overline{R}_j\}$
5: Find $k - l$ records $\overline{R}_j \in C$ which minimize $dist(\overline{R}_j, R_i)$.
6: $\overline{R}_j \Leftarrow \overline{R}_j \cup R_i$
7: **end if**
8: **end for**

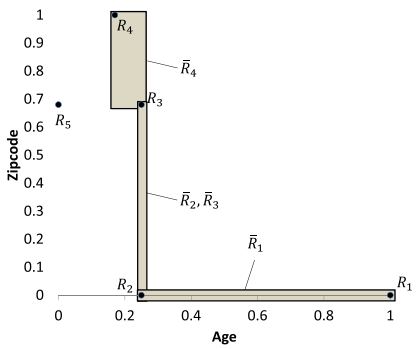


Fig. 9 Example of $(1, k)$-anonymization using NL.

Figure 9 is a running example of $(1, k)$-anonymization for $i = 5$ where $k = 2$. Note that $\overline{R}_5$ is not visualized because it already included R_5. We have to choose additional one generalized record includes R_5. Suppose $NL_5 = \{3, 4\}$; then candidates are $\{\overline{R}_2, \overline{R}_3, \overline{R}_4\}$, because they subsume R_3, R_4. As a result, $\overline{R}_4$ will be chosen and generalized with R_5.

4.5 Algorithm for k-Concealment

Algorithm 7 shows the proposed algorithm for k-concealment. Differences from the original algorithm can be seen in Lines 1, 4, and 10. In the original algorithm, SCC de-

Algorithm 7 Fast k-Concealment

Require: Database $R, \overline{R} \in \mathcal{A}_R^{k,k}$, integer k
Ensure: Database $\overline{R}$ that satisfies k-concealment
1: Enumerate SCC in H
2: **for** $i = 1 \ldots n$ **do**
3: $Q \Leftarrow \{\overline{R}_j \mid R_i \prec \overline{R}_j\}$
4: $P \Leftarrow \{j \mid R_j \prec \overline{R}_i \wedge u_i$ and u_j belongs to the same SCC $\}$.
5: **if** $k \leq |P|$ **then**
6: Goto next i
7: **end if**
8: $j \Leftarrow \arg\min_{j \in Q \setminus P}(dist(\overline{R}_i, R_j))$
9: $\overline{R}_i \Leftarrow \overline{R}_i \cup R_j$
10: Merge SCCs that u_i and u_j belong to.
11: Return to step 3.
12: **end for**

Table 2 Complexities

	Original	**Proposed**
Neighbor list construction	—	$O(kn \log n)$
$(k, 1)$-Anonymization	$O(kn^2)$	$O(k^2 n)$
$(1, k)$-Anonymization	$O(n^2)$	$O(k^2 n)$
k-Concealment	$O(kn^2)$	$O(kn)$
Overall	$O(kn^2)$	$O(kn \log n + k^2 n)$

composition needs to be performed whenever an additional edge is created. Instead, we initially enumerate SCCs, and merge two SCCs when an edge is created. The complexity of this step is reduced to $O(kn)$ in such cases.

In fact, the cost for this step is almost negligible. We can say that the number of generalization in this step is at most $n-1$, because merging SCCs can happen at most $n-1$ times. Consequently, the number of edges added in this step is at most $\frac{1}{k}$ of all edges. Furthermore, an edge (u_j, v_i) always exists when we add edge (u_i, v_j); therefore, we can expect that $dist(\overline{R}_j, R_i)$ is small. For these reasons, we can ignore the increased cost by updating link possibilities lazily.

4.6 Computational Complexities

Table 2 summarizes the computational complexities of the existing scheme and the proposed scheme. Overall, we can reduce from $O(kn^2)$ to $O(kn \log n + k^2 n)$.

5. Experimental Evaluation

To evaluate the performance of the proposed scheme, we conducted some experiments. This section presents the experimental results. All programs were implemented with C++ and compiled by gcc 4.8.2. All experimentation was run on CentOS 7.0 with a 2.4 GHz 12 core processor, 192GB main memory.

5.1 Performance Comparison with the Original Algorithm

First, we compared the proposed method with the original k-concealment algorithm in term of information loss (utility) and execution time.

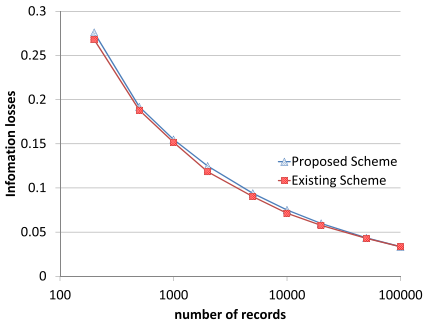


Fig. 10 Information loss w/varying # records.

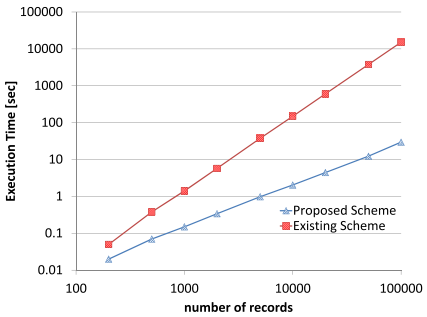


Fig. 11 Execution time w/varying # records.

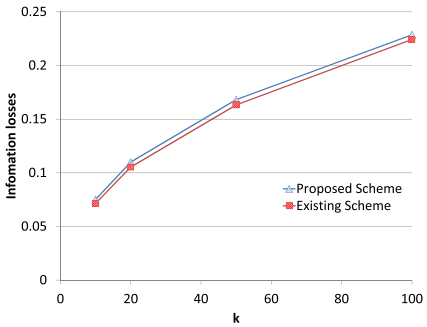


Fig. 12 Information losses w/varying k.

As for the experimental data, we used *Adult* dataset in the UCI Machine Learning Repository [15], and extracted four attributes of numerical domains. To prepare datasets with different numbers of records, we additionally generated synthetic records according to the data distribution in the original dataset.

Figures 10 and 11 show the results when $k = 10$. We can see that the execution time significantly differs between the two methods; the proposed scheme is about 500x faster than the original algorithm with 100,000 records. In the meantime, the information loss remains almost the same level.

We also measured information loss and execution time when varying parameter k. Figures 12 and 13 show the results when the number of records $n = 10{,}000$. Again, the proposed scheme is significantly faster than the original algorithm, while the information loss remains the same

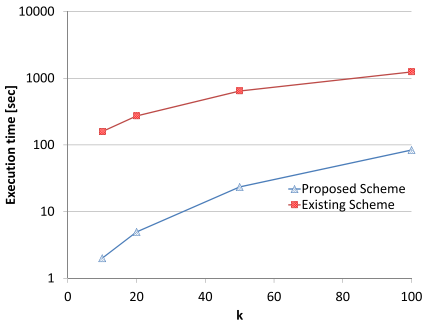


Fig. 13 Execution time w/varying k.

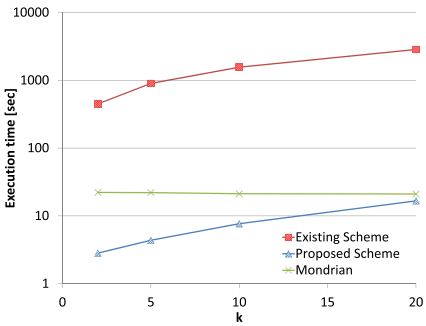


Fig. 14 Compare with mondrian (4 attrs.)

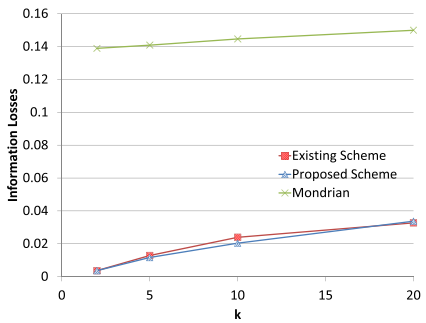


Fig. 15 Compare with mondrian (4 attrs.)

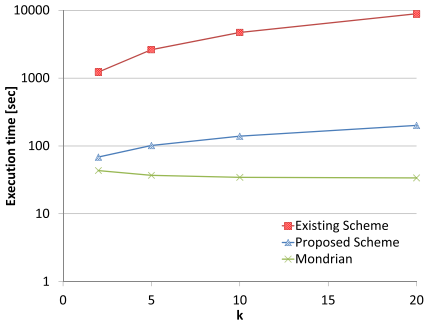


Fig. 16 Compare with mondrian (14 attrs.)

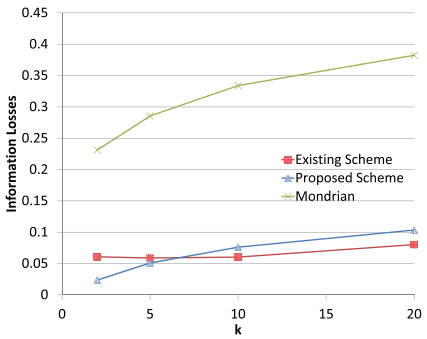


Fig. 17 Compare with mondrian (14 attrs.)

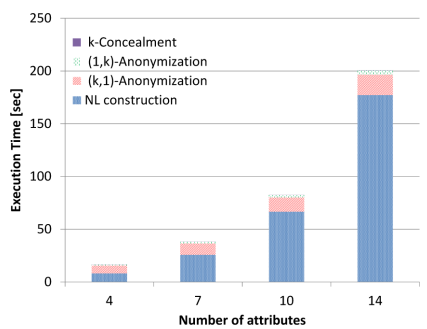


Fig. 18 Execution time breakdown.

level. More precisely, the difference of execution time by two methods will follow $\frac{n}{k}$.

5.2 Compare with k-Anonymity

In addition, we compared the performance of the proposed scheme with one of the fastest k-anonymization algorithm, *Mondrian* [9], and k-concealment algorithm as well.

Figures 14, 15, 16, and 17 show the results of execution time and information loss for four (4) and 14 attributes. In case of four attributes (upper figures), our scheme was the fastest, followed by Mondrian and k-concealment. As for information loss, we observe significant improvement for both our scheme and k-concealment to Mondrian as k-concealment offers better utility in anonymized data.

In the case of 14 attributes, we observe a similar tendency about information loss as four attributes, while Mondrian is faster than the proposed scheme. This is due to the overhead of multidimensional index (kd-tree); i.e., it is known that the efficiency of kd-tree is affected by the number of dimensions. To make a more detailed analysis, Fig. 18 shows a breakdown of execution time in each step in the case of $k = 20$ when varying the number of attributes. As can be seen, NL construction is the most time-consuming part as the number of attributes grows. To this problem, we may choose another multidimensional index that are designed for high dimensional datasets.

6. Conclusion

In this paper we have proposed improved algorithms for efficient k-concealment anonymization. To speed up the process, we have introduced neighbor lists where, for each record, similar records are stored, and have presented al-

gorithms that exploit them. We have conducted some experiments, and the experimental results have shown that the proposed scheme is significantly efficient than the original k-concealment algorithm, while offering improved utility at the same level. Furthermore, we have shown that for low-dimensional data the proposed scheme is as fast as one of the fastest k-anonymization algorithm, Mondrian, while improving utility in anonymized data.

In the future work we plan to improve the performance for high-dimensional data. For this purpose, we attempt to exploit another multidimensional index and improve the related algorithms. Another future work is to cope with biased degree distribution by improved cost function.

References

[1] B.C.M. Fung, K. Wang, R. Chen, and P.S. Yu, "Privacy-preserving data publishing: A survey of recent developments," ACM Comput. Surv., vol.42, no.4, pp.14:1–14:53, June 2010.
[2] L. Sweeney, "K-anonymity: A model for protecting privacy," Int. J. Uncertain. Fuzziness Knowl.-Based Syst., vol.10, no.5, pp.557–570, Oct. 2002.
[3] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," ACM Trans. Knowl. Discov. Data, vol.1, no.1, March 2007.
[4] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," ICDE, pp.106–115, 2007.
[5] K. Stokes and V. Torra, "N-confusion: A generalization of k-anonymity," Proceedings of the 2012 Joint EDBT/ICDT Workshops, EDBT-ICDT '12, New York, NY, USA, pp.211–215, ACM, 2012.
[6] A. Gkoulalas-Divanis, P. Kalnis, and V.S. Verykios, "Providing k-anonymity in location based services," SIGKDD Explor. Newsl., vol.12, no.1, pp.3–10, Nov. 2010.
[7] V.K. Sihag, "A clustering approach for structural k-anonymity in social networks using genetic algorithm," Proceedings of the CUBE International Information Technology Conference, CUBE '12, New York, NY, USA, pp.701–706, ACM, 2012.
[8] T. Tassa, A. Mazza, and A. Gionis, "k-concealment: An alternative model of k-type anonymity," Transactions on Data Privacy, vol.5, no.1, pp.189–222, 2012.
[9] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA, p.25, 2006.
[10] J.L. Bentley, "Multidimensional binary search trees in database applications," IEEE Trans. Softw. Eng., vol.5, no.4, pp.333–340, July 1979.
[11] V.S. Iyengar, "Transforming data to satisfy privacy constraints," Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, New York, NY, USA, pp.279–288, ACM, 2002.
[12] M. Nergiz and C. Clifton, "Thoughts on k-anonymization," Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on, p.96, 2006.
[13] R. Tarjan, "Depth first search and linear graph algorithms," SIAM Journal on Computing, vol.1, no.2, pp.146–160, 1972.
[14] J. Cheriyan and K. Mehlhorn, "Algorithms for dense graphs and networks on the random access computer," Algorithmica, vol.15, no.6, pp.521–549, 1996.
[15] D.N.A. Asuncion, "UCI machine learning repository," 2007.



Ryosuke Koyanagi received the B. Inf. Eng. from University of Tsukuba in 2014. He is currently a graduate student in Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba. His research interests include algorithms and data structures for data privacy and semi-structured data. He is a student member of IPSJ and DBSJ.



Ryo Furukawa received a B.E. in Precision Engineering from Tokyo Metropolitan University in 2006, and M.E. in Engineering from Tokyo Institute of Technology in 2008. He is currently a researcher of NEC Corporation since 2008. His research interests include privacy protection and optimization for data analytics.



Tsubasa Takahashi received a B.E. in Information Engineering, M.E. and Ph.D. in Engineering, all from University of Tsukuba, in 2008, 2010 and 2014, respectively. He is currently a researcher of NEC Corporation since 2010. His research interests include data mining and data privacy for massive data analytics. He is a member of IPSJ, DBSJ and ACM.



Takuya Mori received a B.E. in Naval Architecture and Ocean Engineering in 1995 and a M.E. in Information Engineering from the University of Tokyo in 1997. He is currently a researcher of NEC Corporation since 1997. His research interests include data privacy for massive data analytics and cyber security. He is a member of IPSJ.



Toshiyuki Amagasa received B.E., M.E., and Ph.D degrees from the Department of Computer Science, Gunma University in 1994, 1996, and 1999, respectively. Currently, he is an associate professor at the Division of Information Engineering, Faculty of Engineering, Information and Systems, University of Tsukuba. His research interests cover database systems, XML databases, and database application in scientific domains. He is a senior member of IEICE and IEEE, and a member of IPSJ, DBSJ, and ACM.



Hiroyuki Kitagawa received the B.Sc. degree in physics and the M.Sc. and Dr.Sc. degrees in computer science, all from the University of Tokyo. He is currently a full professor at Faculty of Engineering, Information and Systems and at Center for Computational Sciences, University of Tsukuba. He serves as President of the Database Society of Japan. His research interests include data integration, databases, data mining, and information retrieval. He is an IEICE Fellow, an IPSJ Fellow, a member of ACM and IEEE, and an Associate Member of the Science Council of Japan.