Privacy Streamliner: A Two-Stage Approach to Improving Algorithm Efficiency

Wen Ming Liu and Lingyu Wang
Concordia Institute for Information Systems Engineering, Concordia University
Montreal, QC H3G 1M8, Canada
{l_wenmin,wang}@ciise.concordia.ca

ABSTRACT

In releasing data with sensitive information, a data owner usually has seemingly conflicting goals, including privacy preservation, utility optimization, and algorithm efficiency. In this paper, we observe that a high computational complexity is usually incurred when an algorithm conflates the processes of privacy preservation and utility optimization. We then propose a novel *privacy streamliner* approach to decouple those two processes for improving algorithm efficiency. More specifically, we first identify a set of potential privacy-preserving solutions satisfying that an adversary's knowledge about this set itself will not help him/her to violate the privacy property; we can then optimize utility within this set without worrying about privacy breaches since such an optimization is now simulatable by adversaries. To make our approach more concrete, we study it in the context of micro-data release with publicly known generalization algorithms. The analysis and experiments both confirm our algorithms to be more efficient than existing solutions.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*security, integrity, and protection*; K.6.m [**Management of Computing and Information Systems**]: Miscellaneous—*Security*

General Terms

Security, Theory, Algorithms

Keywords

Privacy Preservation, Micro-Data Disclosure, *l*-Diversity

1. INTRODUCTION

In many privacy-preserving applications ranging from micro-data release [9] to social networks [7, 20], a major challenge is to keep private information secret while optimizing the utility of disclosed or shared data. Recent studies further reveal that utility optimization may actually interfere with privacy preservation by leaking additional private information when algorithms are regarded as public knowledge [25, 29]. Specifically, an adversary can determine a guess of the private information to be invalid if it would have caused the disclosed data to take a different form with better utility. By eliminating such invalid guesses, the adversary can then obtain a more accurate estimation of the private information.

A natural solution to this problem is to simulate the aforementioned adversarial reasoning [18, 25, 29]. Specifically, since knowledge about utility optimization can assist an adversary in refining his/her mental images of the private information, we can first simulate such reasoning to obtain the refined mental images, and then enforce the privacy property on such images instead of the disclosed data. However, it has been shown that such approaches are inherently recursive and deemed to incur a high complexity [29].

In this paper, we observe that the interference between privacy preservation and utility optimization actually arises from the fact that those two processes are usually mixed together in an algorithm. On the other hand, we also observe a simple fact that *to meet both goals does not necessarily mean to meet them at exactly the same time*. Based on such observations, we propose a novel *privacy streamliner* approach to decouple the process of privacy preservation from that of utility optimization in order to avoid the expensive recursive task of simulating the adversarial reasoning.

To make our approach more concrete, we study it in the context of micro-data release with publicly known generalization algorithms. Unlike traditional algorithms, which typically evaluate generalization functions in a predetermined order and then release data using the first function satisfying the privacy property, a generalization algorithm under our approach works in a completely different way: The algorithm starts with the set of generalization functions that can satisfy the privacy property for the given micro-data table; it then identifies a subset of such functions satisfying that knowledge about this subset itself will not assist an adversary in violating the privacy property (which is generally not true for the set of all functions, as we will show later); utility optimization within this subset then becomes simulatable by adversaries [12], and is thus guaranteed not to affect the privacy property. We believe that this general principle can be applied to other similar privacy preserving problems, although developing the actual solution may be application-specific and non-trivial.

The contribution of this paper is twofold. First, our privacy streamliner approach is presented through a general framework that is independent of specific algorithmic constructions or utility metrics. This allows our approach to be easily adapted to a broad range of applications to yield efficient solutions. We demonstrate such possibilities by devising three generalization algorithms to suit different needs while following exactly the same approach. Second, our algorithms provide practical solutions for privacy-preserving micro-data release with public algorithms. As confirmed by both

CODASPY'12, February 7–9, 2012, San Antonio, Texas, USA.

A Micro-Data Table t_0

Name	DOB	Condition
Ada	1985	flu
Bob	1980	flu
Coy	1975	cold
Dan	1970	cold
Eve	1965	HIV

The Disclosure Sets

Name	Condition									
	t_{01}	t_{02}	t_{03}	t_{04}	t_{05}	t_{06}	t_{07}	t_{08}	t_{09}	t_{10}
Ada	flu	cold	flu	cold	flu	cold	flu	cold	HIV	HIV
Bob	flu	cold	flu	cold	HIV	HIV	flu	cold	flu	cold
Coy	cold	flu	cold	flu	cold	flu	HIV	HIV	cold	flu
Dan	cold	flu	HIV	HIV	cold	flu	cold	flu	cold	flu
Eve	HIV	HIV	cold	flu	flu	cold	cold	flu	flu	cold

Table 1: The Motivating Example

complexity analysis and experimental results, those algorithms are more efficient than existing algorithms.

The rest of this paper is organized as follows. We first build intuitions through an example in the remainder of this section. We then present our main approach and supporting theoretical results in Section 2. Section 3 devises three generalization algorithms by following the approach. Section 4 experimentally evaluates the efficiency and utility of our algorithms. We discuss the possibilities for extending our approach and the practicality of the approach in Section 5. We then review related work in Section 6, and finally conclude the paper in Section 7.

Motivating Example

The left table in Table 1 shows a micro-data table t_0 to be released. To protect individuals' privacy, the *identifier* Name will not be released. Also, the identifiers are partitioned into *anonymized groups*, with the *quasi-identifier* DOB inside each such group modified to be the same value [22] (in this paper, we will only consider generalization and leave suppression [15] and bucketization [26] for the future work). For simplicity, we will focus on the partitioning of identifiers while omitting the modification of quasi-identifiers. For this particular example, we assume the desired privacy property to be that the highest ratio of a *sensitive value* Condition in any anonymized group must be no greater than $\frac{2}{3}$ [19].

By our *privacy streamliner* approach, we need to start with all partitions of the identifiers that can satisfy the privacy property. In this example, any partition that includes $\{Ada, Bob\}$ or $\{Coy, Dan\}$ will violate the privacy property, since the two persons inside each of those groups share the same condition. It can be shown that there are totally 9 partitions satisfying the privacy property, as shown below. We will refer to the set of such identifier partitions as the *locally safe set (LSS)*.

$$P_1 = \{\{Ada, Coy\}, \{Bob, Dan, Eve\}\},$$
$$P_2 = \{\{Ada, Dan\}, \{Bob, Coy, Eve\}\},$$
$$P_3 = \{\{Ada, Eve\}, \{Bob, Coy, Dan\}\},$$
$$P_4 = \{\{Bob, Coy\}, \{Ada, Dan, Eve\}\},$$
$$P_5 = \{\{Bob, Dan\}, \{Ada, Coy, Eve\}\},$$
$$P_6 = \{\{Bob, Eve\}, \{Ada, Coy, Dan\}\},$$
$$P_7 = \{\{Coy, Eve\}, \{Ada, Bob, Dan\}\},$$
$$P_8 = \{\{Dan, Eve\}, \{Ada, Bob, Coy\}\},$$
$$P_9 = \{\{Ada, Bob, Coy, Dan, Eve\}\}$$

It may seem to be a viable solution to start optimizing data utility inside the LSS, since every partition here can satisfy the privacy property. However, such an optimization may still violate the privacy property, because it is not simulatable by adversaries [12] unless if we assume the LSS to be public knowledge (that is, adversaries may know that each identifier partition in the LSS can satisfy the privacy property for the unknown table t_0). Unfortunately, this knowledge about LSS could help adversaries to violate the privacy property. In this case, it can be shown that adversaries' mental image about the micro-data table would only include t_{01} and t_{02} shown in the right table in Table 1. In other words, adversaries can determine that t_0 must be either t_{01} or t_{02}. Clearly, the privacy property is violated since *Eve* is associated with *HIV* in both cases.

Since the LSS may contain too much information to be assumed as public knowledge, we turn to its subsets. In this example, it can be shown that by removing P_7 from the *LSS*, the disclosure set becomes $\{t_{01}, t_{02}, t_{03}, t_{04}\}$. The privacy property is now satisfied since the highest ratio of a sensitive value for any identifier is $\frac{1}{2}$. We call such a subset of the LSS the *globally safe set (GSS)*. Optimizing data utility within the GSS will not violate privacy property, because the GSS can be safely assumed as public knowledge and the optimization is thus simulatable by adversaries.

However, there is another complication. At the end of utility optimization, one of the generalization functions in the GSS will be used to release data. The information disclosed by the GSS and that by the released data is different, and by intersecting the two, adversaries may further refine their mental image of the micro-data table. In this example, since the adversaries' mental image about the micro-data table in terms of the GSS is $\{t_{01}, t_{02}, t_{03}, t_{04}\}$, adversaries know both *Ada* and *Bob* must be associated with either *flu* or *cold*. Now suppose the utility optimization selects P_3, then from the released table, adversaries will further know that either *Ada* or *Eve* must have *flu* while the other has *HIV*. Therefore, adversaries can now infer that *Ada* must have *flu*, and *Eve* must then have *HIV*.

To address this issue, we will further confine the utility optimization to a subset of the GSS. In this example, if we further remove P_3, P_6, P_8 from the GSS, then the corresponding mental image of adversaries will contain all the 10 tables (from t_{01} to t_{10}). It can be shown that now the privacy property will always be satisfied regardless of which partition is selected during utility optimization. Taking P_1 as an example, from its corresponding generalized table, adversaries may further refine their mental image about t_0 as the first six tables (from t_{01} to t_{06}), but the highest ratio of a sensitive value is still $\frac{1}{2}$. We call such a subset of identifier partitions the *strongly globally safe set (SGSS)*. The SGSS allows us to optimize utility without worrying about violating the privacy property.

Therefore, the key problem in applying the privacy streamliner approach is to find the SGSS. The naive solution of directly following the above example to compute the LSS, GSS, and eventually SGSS is clearly impractical due to the large solution space. In the rest of this paper, we will present more efficient ways to directly construct the SGSS without first generating the LSS or GSS.

2. THE MODEL

We first give the basic model in Section 2.1. We then introduce the concept of *l-candidate* and *self-contained property* in Section 2.2. Finally, we prove that the SGSS can be efficiently con-

structed using those concepts in Section 2.3. Table 2 summarizes our notations.

$t_0, t, t(id, q, s)$	Micro-data table
$\mathcal{I}$, $\mathcal{Q}$, $\mathcal{S}$	Projection $\Pi_{id}(t)$, $\Pi_q(t)$, $\Pi_s(t)$
R_{iq}, R_{qs}, R_{is}	Projection $\Pi_{id,q}(t)$, $\Pi_{q,s}(t)$, $\Pi_{id,s}(t)$
$C(.\|t)$, $C_i(.\|t)$	A color of table t
$S^C(.\|t)$	The set of colors in t
$P(.\|t)$, $P_i(.\|t)$	A identifier partition of table t
$S^P(.\|t)$	A set of identifier partitions of t
$ss^l(.\|t)$	Locally safe set (LSS) of t
$ss^g(.\|t)$	Globally safe set (GSS) of t
$ss^s(.\|t)$	Strongly globally safe set (SGSS) of t

Table 2: The Notation Table

2.1 The Basic Model

We denote a micro-data table as $t_0(id, q, s)$ where id, q, and s denote the *identifier*, *quasi-identifier*, and *sensitive value*, respectively (each of which may represent multiple attributes). Denote by $\mathcal{I}$, $\mathcal{Q}$, $\mathcal{S}$ the set of identifier values $\Pi_{id}(t_0)$, quasi-identifier values $\Pi_q(t_0)$, and sensitive values $\Pi_s(t_0)$ (all projections preserve duplicates, unless explicitly stated otherwise). Also, denote by R_{iq}, R_{qs}, R_{is} the projections $\Pi_{id,q}(t_0)$, $\Pi_{q,s}(t_0)$, $\Pi_{id,s}(t_0)$, respectively.

As typically assumed, $\mathcal{I}$, $\mathcal{Q}$, and their relationship R_{iq} may be known through external knowledge, and $\mathcal{S}$ is also known once a generalization is released. Further, we make the worst case assumption that each tuple in t_0 can be linked to a unique identifier value through the corresponding quasi-identifier value. Therefore, both R_{is} and R_{qs} need to remain secret to protect privacy. Between them, R_{is} is considered as the private information and R_{qs} as the utility information.

We say a micro-data table t_0 is *l-eligible* if at most $\frac{|t_0|}{l}$ tuples in t_0 share the same sensitive value. We call the set of all identifier values associated with the same sensitive value s_i a *color*, denoted as $C(t_0, s_i)$ or simply C_i when t_0 and s_i are clear from the context. We use $S^C(t_0)$ or simply S^C to denote the collection of all colors in t_0.

EXAMPLE 1. *The left-hand side of Table 3 (the right-hand side will be needed for later discussions) shows a micro-data table t_0 in which there are two colors: $C_1 = \{id_1, id_2\}$ and $C_2 = \{id_3, id_4\}$, so $S^C = \{C_1, C_2\}$.* ⊡

R_{qs}

id	q	s
id_1	q_1	s_1
id_2	q_2	s_1
id_3	q_3	s_2
id_4	q_4	s_2
$\mathcal{I}$	$\mathcal{Q}$	$\mathcal{S}$

$P_1 = \{\{id_1, id_3\}, \{id_2, id_4\}\}$
$P_2 = \{\{id_1, id_4\}, \{id_2, id_3\}\}$
$P_3 = \{\{id_1, id_2, id_3, id_4\}\}$
$P_4 = \{\{id_1, id_2\}, \{id_3, id_4\}\}$

Table 3: An Example

We denote by $ss^l(t_0)$, $ss^g(t_0)$, and $ss^s(t_0)$ the *locally safe set (LSS)*, *globally safe set (GSS)*, and *strongly globally safe set (SGSS)* for a given t_0, respectively (those concepts have been illustrated in Section 1).

EXAMPLE 2. *Continuing Example 1 and assuming the privacy property to be 2-diversity [19], it can be shown that $ss^l(t_0) = \{P_1, P_2, P_3\}$ and $P_4 \notin ss^l$ where P_1, P_2, P_3, P_4 are shown on the right-hand side of Table 3. Further, $\{P_1, P_3\}$ and $\{P_2, P_3\}$ are both GSS and SGSS.* ⊡

We have previously given a sufficient condition for the SGSS, namely, the *l-cover* property [31]. In other words, a set of identifier partitions S^P is a SGSS with respect to l-diversity if it satisfies l-cover (however, no concrete method is given there to satisfy this property, which is the focus of this paper). Intuitively, l-cover requires each color to be indistinguishable from at least $l-1$ other sets of identifiers in the identifier partition. If no ambiguity is possible, we also refer to a color C together with its $l-1$ covers as the l-cover of C. As these concepts are needed later in the proofs of our main results discussed in Section 2.3, we repeat them in Definition 4 and 5 shown in Appendix A (note the remaining content of this paper can be understood without those definitions).

2.2 l-Candidate and Self-Contained Property

We first give a necessary but not sufficient condition for l-cover, namely, *l-candidate*. As formally stated in Definition 1, subsets of identifiers can be candidates of each other, if there exists one-to-one mappings between those subsets that always map an identifier to another in a different color. We will prove later that any collection of subsets of identifiers can be l-cover for each other only if they form an l-candidate.

DEFINITION 1 (l-CANDIDATE). *Given an l-eligible micro-data table t_0, we say*

- *$ids_1 \subseteq \mathcal{I}$ and $ids_2 \subseteq \mathcal{I}$ are* candidate *for each other, if*
  - *$ids_1 \cap ids_2 = \emptyset$ and $|ids_1| = |ids_2|$, and*
  - *there exists a bijection $f : ids_1 \rightarrow ids_2$, such that every $id \in ids_1$ and $f(id) \in ids_2$ are from different colors.*
- *$ids_1, ids_2, \ldots, ids_l \subseteq \mathcal{I}$ form a l-*candidate*, if for all $(1 \leq i \neq j \leq l)$, ids_i and ids_j are candidates for each other.*
- *Denote by $Can^l(.|t_0) = (can_1, can_2, \ldots, can_{|S^C|})$ a sequence of $|S^C|$ l-candidates each can_i of which is the l-candidate for the color C_i in t_0 (note that there is exactly one l-candidate for each color in the sequence, and $Can^l(.|t_0)$ is not necessarily unique for t_0).*

EXAMPLE 3. *In the table shown on the left-hand side of Table 3, the two colors $C_1 = \{id_1, id_2\}$ and $C_2 = \{id_3, id_4\}$ are candidates for each other, and they together form a 2-candidate $\{C_1, C_2\}$. Also, we have that $Can^l(.|t_0) = (\{C_1, C_2\}, \{C_1, C_2\})$ (note that $Can^l(.|t_0)$ denotes the sequence of l-candidates and we use the indices in the multiset to present the order in the remainder of this paper, and if no ambiguity is possible, we shall not distinguish the notations between a collection and a sequence). In this special case, it has two identical elements, the first one for C_1 and the second one for C_2, since both colors have the same l-candidate.* ⊡

Next we introduce the *self-contained* property in Definition 2. Informally, an identifier partition is self-contained, if the partition does not break the one-to-one mappings used in defining the l-candidates. Later we will show that the self-contained property is sufficient for an identifier partition to satisfy the l-cover property and thus form a SGSS.

DEFINITION 2 (SELF-CONTAINED PROPERTY AND FAMILY SET). *Given a micro-data table t_0 and a collection of l-candidates Can^l, we say*

- *an anonymized group G in an identifier partition P is* self-contained *with respect to Can^l, if for every pair of identifiers $\{id_1, id_2\}$ that appears in any bijection used to define Can^l, either $G \cap \{id_1, id_2\} = \emptyset$ or $G \cap \{id_1, id_2\} = \{id_1, id_2\}$ is true.*
- *an identifier partition P is self-contained if for each $G \in P$, G is self-contained.*
- *a set S^P of identifier partitions is self-contained, if for each $P \in S^P$, P is self-contained; we also call such a set S^P a* family set *with respect to Can^l.*

Next we introduce the concept of *minimal self-contained identifier partition* in Definition 3 to depict those identifier partitions that not only satisfy the self-contained property but have anonymized groups of minimal sizes. Intuitively, for any given collection of l-candidates Can^l, a minimal self-contained identifier partition may yield optimal data utility under certain utility metrics (we will discuss this in more details later).

DEFINITION 3 (MINIMAL SELF-CONTAINED PARTITION). *Given a micro-data table t_0 and a collection of l-candidates Can^l, an identifier partition P is called the* minimal self-contained partition *with respect to Can^l, if*

- *P satisfies the self-contained property with respect to Can^l, and*
- *for any anonymized group $G \in P$, no $G' \subset G$ can satisfy the self-contained property.*

EXAMPLE 4. *In Example 3, assume the bijections used to define l-candidate for C_1 in Can^l are $f_1(id_1) = id_3$ and $f_1(id_2) = id_4$ while for C_2 are $f_2(id_3) = id_1$ and $f_2(id_4) = id_2$, then the identifier partitions P_1 and P_3 shown in the left-hand side of Table 3 satisfy the self-contained property, whereas P_2 does not. Also, P_1 is the minimal self-contained identifier partition, and $\{P_1\}$, $\{P_3\}$, $\{P_1, P_3\}$ are all family sets.* □

Similarly, assume the bijections used to define l-candidate for C_1 in Can^l are $f_1(id_1) = id_4$ and $f_1(id_2) = id_3$ while for C_2 are $f_2(id_3) = id_2$ and $f_2(id_4) = id_1$, then the identifier partitions P_2 and P_3 satisfy the self-contained property, whereas P_1 does not. Also, P_2 is the minimal self-contained identifier partition, and $\{P_2\}$, $\{P_3\}$, $\{P_2, P_3\}$ are all family sets. Finally, assume $f_1(id_1) = id_3$, $f_1(id_2) = id_4$ and $f_2(id_3) = id_2$, $f_2(id_4) = id_1$, then in this case only P_3 satisfies self-contained property, whereas P_1 and P_2 do not. It is clearly evidenced by this example that, given micro-data table, its minimal self-contained partition is determined not only by the Can^l, but also the corresponding bijections. In this paper, we focus on deriving Can^l and constructing minimal self-contained partitions as well as family sets based on the bijections. Therefore, unless explicitly stated otherwise, Can^l is referred to itself together with the corresponding bijections in the remainder of this paper.

2.3 Main Results

In this section, we first prove that the self-contained property and l-candidate provide a way for finding identifier partitions that satisfy the l-cover property, and then we prove results for constructing l-candidates. All the proofs can be found in the Appendix B due to space limitations.

First, in Lemma 1, we show that a minimal self-contained identifier partition always satisfies the l-cover property.

LEMMA 1. *Given an l-*eligible *micro-data table t_0, every minimal self-contained partition satisfies the l-*cover *property. Moreover, for each color C, its corresponding l-candidate in Can^l is also an l-cover for C (that is, C together with its $l-1$ covers).*

In Lemma 2, we prove that an anonymized group in any self-contained identifier partition must either also be a group in the minimal self-contained partition, or be a union of several such groups. This result will be needed in later proofs.

LEMMA 2. *Given any l-*eligible *t_0, a collection of l-candidates Can^l and its corresponding minimal self-contained partition $P^{lm} = \{ids_1, ids_2, \ldots, ids_k\}$, any self-contained identifier partition P satisfies that $\forall(G \in P)$, either $G \cap ids_i = \emptyset$ or $G \supseteq ids_i$ ($i \in [1, k]$) is true.*

Based on Lemma 1 and 2, we now show that similar results hold for any self-contained identifier partition and any family set, as formulated in Theorem 1.

THEOREM 1. *Given an l-*eligible *t_0 and the l-candidates Can^l, we have that*

- *any self-contained identifier partition P satisfies the l-cover property. Moreover, for each color in t_0, the corresponding l-candidate in Can^l is also the l-cover for P.*
- *any family set S^{fs} satisfies the l-cover property. Moreover, for each color in t_0, the corresponding l-candidate in Can^l is also the l-cover for S^{fs}.*

Based on the above results, once the collection of l-candidates is determined, we can easily construct sets of identifier partitions to satisfy the l-cover property. Therefore, we now turn to finding efficient methods for constructing l-candidates. First, Lemma 3 and 4 present conditions for subsets of identifiers to be candidates for each other.

LEMMA 3. *Given an l-eligible t_0, any $ids \subseteq \mathcal{I}$ that satisfies $|ids| = |C|$ and $ids \cap C = \emptyset$ is a candidate for color C.*

LEMMA 4. *Given an l-eligible t_0, any $ids_1, ids_2 \subseteq \mathcal{I}$ satisfying following conditions are candidates for each other:*

- *$|ids_1| = |ids_2|$ and $ids_1 \cap ids_2 = \emptyset$, and*
- *the number of all identifiers in $ids_1 \cup ids_2$ that belong to the same color is no greater than $|ids_1|$.*

Based on Lemma 3 and 4, we now present conditions for constructing l-candidates of each color in Theorem 2. We will apply those conditions in the next section to design practical algorithms for building the SGSS.

THEOREM 2. *Given an l-eligible t_0, each color C together with any $(l-1)$ subsets of identifiers $\{ids_1, ids_2, \ldots, ids_{l-1}\}$ that satisfy following conditions form a valid l-candidate for C:*

- *$\forall(x \in [1, l-1])$, $|ids_x| = |C|$ and $ids_x \cap C = \emptyset$;*
- *$\forall((x, y \in [1, l-1]) \wedge (x \neq y))$, $ids_x \cap ids_y = \emptyset$;*
- *the number of all identifiers in $\cup_{x=1}^{l-1} ids_x$ that belong to the same color is no greater than $|C|$.*

3. THE ALGORITHMS

In this section, we design three algorithms for constructing l-candidates for colors and analyze their complexities. It is important to note that there may exist many other ways for constructing l-candidates based on the conditions given in Theorem 2. This flexibility allows us to vary the design of algorithms to suit different needs of various applications, because different l-candidates will also result in different SGSSs and hence algorithms more suitable for different utility metrics. We demonstrate such a flexibility through designing three algorithms in the following.

To simplify our discussions, we say an identifier is *complete* (or *incomplete*) if it is (or is not) included in any l-candidate; similarly, we say a color is *complete* (or incomplete) if it only includes complete identifiers (or otherwise); we also say a set of identifiers is *compatible* (or *incompatible*) with an identifier id, if there does not exist (or exists) identifier in that set that is from the same color as id; finally, given any color, an identifier from other colors is said to be *unused* with respect to that color if it has not yet been selected as a candidate for any identifier in that color. Table 4 summarizes the notations used in the algorithms.

n	The number of (incomplete) tuples in t_0
C_i	The i^{th} color, or the set of (incomplete) identifiers in the i^{th} color
n_c	The number of (incomplete) colors in t_0
S^C	The sequence of (incomplete) colors in t_0
n_i	The number of (incomplete) tuples in color C_i
can_{ia}	The set of $(l-1)$ identifiers selected for identifier id_{ia} in color C_i
can_i	l-candidate for color i
Can^l	The collection of l-candidates

Table 4: Notations for Algorithms

3.1 The RIA Algorithm (Random and Independent)

The main intention in designing the RIA algorithm is to show that, based on our results in Theorem 2, l-candidate can actually be built in a very straightforward way, although its efficiency and utility is not necessarily optimal. In the RIA algorithm, to construct the l-candidates for each color C_i, $(l-1)$ identifiers can_{ia} are selected randomly and independently for each identifier id_{ia} in C_i. The only constraint in this selection process for any color is that the same identifier will not be selected more than once. Clearly, designing such an algorithm is very straightforward. Roughly speaking, for each identifier id_{ia} in any color C_i, RIA randomly selects $(l-1)$ identifiers from any other $(l-1)$ colors that are not selected by other identifiers in C_i, and then form l-candidate can_i for C_i from the can_{ia} of each identifier.

The RIA algorithm is shown in Table 5. RIA first set $Can^l = \phi$ (line 1). Then, Given the l-eligible table t_0, RIA iteratively constructs l-candidate for all its colors (line 2-9). In each iteration, RIA first repeatedly selects $(l-1)$ identifiers can_{ia} for each identifier $id_{i,a}$ in color C_i. These identifiers are from $(l-1)$ different colors and not be used yet by the other identifier in current color. Then RIA builds the $(l-1)$ candidates for current color. To construct the w^{th} candidate, RIA selects the w^{th} identifier from each can_{ia} for each identifier $id_{i,a}$ in color C_i. Consequently C_i, together with its $(l-1)$ candidates, form the l-candidate, can_i, for color C_i. Finally, all the can_i for each color form the set Can^l of l-candidates, and RIA terminates and returns Can^l.

The computational complexity of RIA algorithm is $O(l \cdot n)$ since: since: first, for each color, each of its identifiers costs exactly $(l-1)$ many constant times (line 4-6) to select its $(l-1)$ identifiers, and there are n_i identifiers in the color, so totally $(l-1) \times n_i$. Then,

Input: an l-eligible Table t_0, the privacy property l;
Output: the set Can^l of l-candidates for each color;
Method:
1. **Let** $Can^l = \emptyset$;
2. **For** $i = 1$ to n_c
   // Iteratively construct l-candidate for each color C_i
3. **For** $a = 1$ to n_i
   // Iteratively select the $l-1$ number of identifiers for each identifier $id_{i,a}$ in color C_i
4. Randomly select $l-1$ different colors S^C_{ia} from $S^C \backslash \{C_i\}$;
5. Randomly select one *unused* identifier from each color in S^C_{ia};
6. Form can_{ia} by collecting the previously selected $l-1$ identifiers in any order;
7. **For** $i = 1$ to n_c
8. **For** $w = 1$ to $l-1$
   // Create the l-candidate can_i for C_i based on the $can_{ia}(a \in [1, n_i])$
9. Create in can_i its w^{th} candidate: $\bigcup_{a=1}^{n_i}$ (the w^{th} identifier in can_{ia});
10. **Let** $Can^l = \{can_i : 1 \le i \le n_c\}$;
11. **Return** Can^l;

Table 5: The RIA Algorithm

based on these identifiers, it takes $(l-1) \times n_i$ many times to create its l-candidate. There are totally n_c many colors in the micro-table. Finally it takes n_c many times to create the set of l-candidates. Therefore, in totally its computational complexity is $O(\sum_i^{n_c}(2 \times (l-1) \times n_i) + n_c) = O(l \times n)$, because the size of all colors adds up to be n, and $n_c \le n$. Note that once an identifier select same identifier which was selected by the previously considered identifier in the color, RIA must reselect other identifier for that identifier. During the analysis of computational complexity, we ignore the time of solving such conflicts in colors and identifiers in line 4 and line 5 respectively. It is reasonable for most cases in the real life that $n_i \times (l-1) \ll n$, since in such case the probability of conflicts is very low. Note that the RIA algorithm only builds the l-candidates. In order to obtain the self-contained identifier partition and hence the SGSS (as shown in Theorem 1), we still need to merge the can_{ia}'s that share the common identifiers (which actually has a higher complexity than $O(l \times n)$, but we will not further discuss it since our intention of introducing the RIA algorithm is not due to its efficiency).

3.2 The RDA Algorithm (Random and Dependent)

The RDA algorithm aims at general-purpose data utility metrics that only depends on the size of each anonymized group in an identifier partition, such as the well known *Discernibility Metric (DM)* [4]. As we shall show through experiments, our RDA algorithm will produce solutions whose data utility by the DM metric is very close to that of the optimal solution, since the RDA algorithm can minimize the size of most anonymized groups in the chosen identifier partition.

Roughly speaking, for the color C_i that has the most incomplete identifiers, the algorithm randomly selects $(l-1)$ identifiers can_{ia} for each of its identifiers id_{ia}, one from each of the next $(l-1)$ colors with the most incomplete identifiers, until the number of incomplete colors is less than l. For the remaining identifiers, the

Input: an l-eligible Table t_0, the privacy property l;
Output: the set Can^l of l-candidates for each color;
Method:
1. **Let** n_c be the number of colors in t_0;
2. **Let** S^C be the sequence of the colors in the non-increasing order of their cardinality;
3. **Let** C_i, n_i ($i \in [1, n_c]$) be the i^{th} color and its cardinality;
4. **While** ($n_c \geq l$)
   //Construct l-candidate for the color in which most number of incomplete identifiers
5. Determine the color C_i which has most number of incomplete identifiers;
6. **For** $a = 1$ to n_i
7. **If**($id_{i,a}$ is *complete*)
8. **Skip** to check the next identifier in current color;
   // Iteratively select the $l-1$ number of identifiers for each identifier $id_{i,a}$ in color C_i
9. Randomly select $l-1$ *incomplete* identifiers from $l-1$ different colors in S^C with most *incomplete* identifiers;
10. Form can_{ia} by collecting the previously selected $l-1$ identifiers in any order;
11. Remove the *complete* colors from S^C, and recalculate n_c;
12. Reorder the colors in S^C based on their number of *incomplete* identifiers;
13. **If** ($n_c < l$) **Break**;
14. **While** ($S^C \neq \emptyset$)
15. Select any *incomplete* identifier $id_{j,b}$ from the color $C_j \in S^C$ with the most number of *incomplete* identifiers;
16. Select any $l-1$ identifiers from the *compatible* can_{ia} with the minimal cardinality;
17. Form can_{jb} by collecting the previously selected $l-1$ identifiers in any order;
18. **If** (color C_i is *complete*) Remove it from S^C;
19. **For** $i = 1$ to n_c
20. **For** $w = 1$ to $l-1$
    // Create the l-candidate can_i for C_i based on the $can_{ia}(a \in [1, n_i])$
21. Create in can_i its w^{th} candidate: $\bigcup_{a=1}^{n_i}$(the w^{th} identifier in can_{ia});
22. **Let** $Can^l = \{can_i : 1 \leq i \leq n_c\}$;
23. **Return** Can^l;

Table 6: The RDA Algorithm

algorithm simply selects any $l-1$ identifiers as their candidates from any compatible can_{ia}. The key difference from the RIA algorithm is that the RDA algorithm will not consider an identifier once it has selected its candidates, or been selected as a candidate, in most cases. This difference not only improves the data utility by minimizing the size of anonymized groups in the identifier partition, but also ensures the sets of candidates selected for different identifiers to be disjoint, which eliminates the need for the expensive merging process required by the RIA algorithm.

The RDA algorithm is shown in Table 6. Compared to RIA algorithm, RDA simply skips and does not reselect the $l-1$ identifiers for the $l-1$ candidates if the identifiers have been selected (line 7-8), and ensures that each identifier is not selected as candidates (line 9). Specifically, RDA algorithm first sets n_c, C_i, n_i, and S^C to be the number of colors, the i^{th} color and its cardinality, and the sequence of colors in the non-increasing order of cardinality in t_0 respectively (line 1-3). Then, RDA iteratively selects $l-1$ identifiers $can_{i,a}$ for each identifier $id_{i,a}$ in color C_i until the number of incomplete colors is less than l (line 4-13) . Here C_i is the color which has the most number of incomplete identifiers in S^C. In each iteration, RDA first selects one incomplete color with most incomplete identifiers (line 5). Then for each of its incomplete identifiers, RDA forms can_{ia} by randomly selecting $(l-1)$ incomplete identifiers from $(l-1)$ different colors in S^C (line 9-10), and removes the completed colors from S^C, recounts n_c, and reorders the colors in S^C in the non-increasing order of the number of incomplete identifiers (line 11-12). Next, RDA forms can_{ia} for the remainder identifiers (line 14-18). In each iteration, RDA first selects any incomplete identifier $id_{j,b}$ from the color C_j with the most number of incomplete identifiers (line 15) , and then forms can_{jb} by collecting any $l-1$ identifiers from any compatible can_{ia} with smallest size (line 16-17). Finally, all the can_i for each color form the set Can^l of l-candidates, and RDA terminates and returns Can^l (line 19-23).

Note that, we can derive the minimal self-contained partition directly through the bijections in the l-candidates. In other words, each can_{ia} is a transient group (see proof of Lemma 1) for minimal self-contained partition, furthermore, it is the anonymized group in minimal self-contained partition when the intersection between any two can_{ia} is empty. Actually, the construction of the set of l-candidate based on can_{ia}s (Line 19-22 in RDA algorithm) is only used to prove its existence. Therefore, in order to ensure that can_{ia}s are disjoint, line 16-17 can be replaced by: Append $id_{j,b}$ to its compatible can_{ia} with the minimal cardinality. Since can_{ia}s are disjoint, the merge process in Table 9 can be bypassed. This will reduce the computational complexity and improve the data utility under certain type of utility measures based on the size of the QI-groups, such as DM.

Furthermore, we show that the computational complexity of Line 9-12 is linear in l. First, the remainder colors in S^C are incomplete, and we can also design certain additional data structure to store the incomplete identifiers in each incomplete color and record the cardinality. Therefore, Line 9-10 can be processed in time linear in l. Second, since after Line 9-10, only $l-1$ colors (besides color C_i) are affected and their cardinality is only reduced by 1, Line 11-12 also can be processed in time linear in l with the assistance of addi-

tional structure. Based on previous discussions, the computational complexity of RDA algorithm is $O(n)$. First, Line 1-3 runs in $O(n)$ time by applying bucket sort (Additionally, $n_c << n$ holds for general cases in real world). Second, from Line 4-17, each identifier in the micro-data table is considered once all through the process with the assistance of additional data structure. We will evaluate utility of the RDA algorithm through experiments in the next section.

3.3 The GDA Algorithm (Guided and Dependent)

For both the RIA and RDA algorithms, we have assumed that the utility metric is independent of the actual quasi-identifier values. Our intention of designing the GDA algorithm is to demonstrate how our approach also allows designing algorithms that optimize data utility based on actual quasi-identifier values. For this purpose, assuming the quasi-identifier is composed of attributes $q_1, q_2, \ldots, q_d$, we assign an integer *weight* $weight_i$ to each attribute $q_i (i \in [1, d])$, and a rank $rank \in [1, |q_i|]$ to each value of the attribute q_i. Given any tuple t_a in the micro-data table t_0 and its value of each quasi-identifier attribute $t_a[q_i]$, we define its *weighted-rank* as $wr_a = \sum_{i=1}^{d}(weight_i \times rank(t_a[q_i]))$. Given any two tuples t_a and t_b, we define their *QI-distance* as $d_{ab} = |wr_a - wr_b|$. Also, given a tuple t_a and a set of tuples t_B, we define the *average QI-distance* as $d_{aB} = \frac{\sum_{b \in t_B}(d_{ab})}{|t_B|}$. Intuitively, a smaller QI-distance indicates that placing the two tuples into the same anonymized group will produce better data utility (for example, patients from the same geographical region should be grouped together).

Roughly speaking, for each incomplete identifier $id_{i,a}$ in the color C_i with the most incomplete identifiers, the algorithm determines $l-1$ incomplete colors that can minimize the QI-distance between their first incomplete identifier with the largest weighted-rank and $id_{i,a}$, and then selects these $l-1$ identifiers to be the $l-1$ candidates for $id_{i,a}$, until the number of incomplete colors is less than l. For each remainder identifier $id_{j,b}$, GDA selects $(l-1)$ identifiers from its compatible can_{ia} which has the smallest average QI-distance from $id_{j,b}$.

The GDA algorithm is shown in Table 7. Given a micro-data table t_0 and an integer l, GDA first initialize the following: Set n_c, C_i, n_i, and S^C to be the number of colors, the i^{th} color and its cardinality, and the sequence of colors in the non-increasing order of cardinality in t_0 respectively (line 1-3); Compute the weighted-rank for each identifier (tuple) based on its quasi-identifier information (line 4); Sort the identifiers inside a color in ascending order of their weighted-rank values (line 5). After that, GDA iteratively constructs can_{ia} for each identifier in the micro-table t_0(line 6-11). In each iteration, GDA repeatedly selects $l-1$ identifiers can_{ia} for each identifier $id_{i,a}$ in color C_i. For each identifier $id_{i,a}$, we select the $l-1$ best colors among the whole set of colors other than C_i itself. To judge the best colors, we compare the QI-distance between the QI-attributes of $id_{i,a}$ and the first identifier in each color which is not yet mapped to any identifier in C_i. The less the QI-distance is, the better the identifier is. Finally, all the can_i for each color forms the set Can^l of l-candidates, and GDA terminates and returns Can^l (line 12-16). From the description above, the selection of l-candidate for each color is further decided by the selection of $l-1$ identifiers for each of its identifier, which in turn are selected based on the QI-distance, it is, the local optimization. Therefore, the transient groups are expected to be closer with regard to the QI-attributes, which may increase the data utility. However, this approach cannot assure the size of the anonymized group since there may exists many merges when construct the locally-minimal partition based on such set of l-candidates.

The computational complexity of GDA algorithm is $O(n \log n)$ since after sorting each color based on the weighted-rank values, each identifier is processed only once throughout the process of building l-candidates. Since this algorithm aims at minimizing the average QI-distance inside each anonymized group, we will evaluate its data utility in the next section based on such a quasi-identifier value-dependent metric.

3.4 The Construction of SGSS

Remind that our ultimate objective is to construct strongly globally safe set (SGSS) in which the data utility is optimized later. Once Can^l has been constructed by RIA, RDA, or GDA algorithm, in this paper we adopt the approach based on the corresponding bijections in Can^l to building the minimal self-contained partition and then the SGSS.

More specifically, for RDA and GDA algorithms, each can_{ia}, created in step 10 in Table 6 and in step 9 in Table 7 respectively, forms an anonymized group. Then we simply append the $id_{j,b}$, in step 15 in Table 6 and in step 11 in Table 7 respectively, to the selected can_{ia}. Similarly, for RIA algorithm, each can_{ia} created in step 6 in Table 5 forms an anonymized group, and we then merge the resultant anonymized groups which have common identifiers to be disjoint sets. The algorithms in the literature to achieve disjoint sets are applicable for our problem and the details are omitted here.

For the experiments in Section 4, we integrate the process in building the minimal self-contained partitions into the algorithms of constructing Can^l for RDA and GDA algorithms.

4. EXPERIMENTS

In this section, we evaluate the efficiency and utility of our proposed algorithms through experiments. To compare our results to that reported in [28], our experimental setting is similar to theirs. We adopt two real-world datasets, OCC and SAL, at the Integrated Public Use Micro-data Series [21]. Each dataset contains 600k tuples. The domain sizes of the six chosen attributes of both datasets are shown in Table 8. Among these, we select four attributes, *Age*, *Gender*, *Education*, and *Birthplace*, as the QI-attributes for both datasets, and we select *Occupation* and *Income* as the sensitive attribute for OCC and SAL, respectively. For our experiment, we adopt the *MBR* (*Minimum Bounding Rectangle*) function (similar to that in [28]) to generalize QI-values within the same anonymized group once we obtain an identifier partition using our algorithms. As mentioned before, the RIA algorithm is only introduced to demonstrate how simple an algorithm can be by following our approach, we will not evaluate its performance, but only focus on the RDA and GDA algorithm. In fact, in these two algorithms, each can_{ia} forms an anonymized group (transient group), and for the remainder identifiers shown in step 6 in Table 6 and step 7 in Table 7 are simply appended in the selected compatible anonymized groups (Step 19-22 in Table 6 and step 12-15 in Table 7 are used to represent the l-candidates). All experiments are conducted on a computer equipped with a 1.86GHz Core Duo CPU and 1GB memory.

Attribute	Age	Gender	Education	Birthplace	Occupation	Income
Domain Size	79	2	17	57	50	50

Table 8: Description of OCC and SAL Datasets

We evaluate computational complexity using execution time, and evaluate data utility of the released table using two measurements:

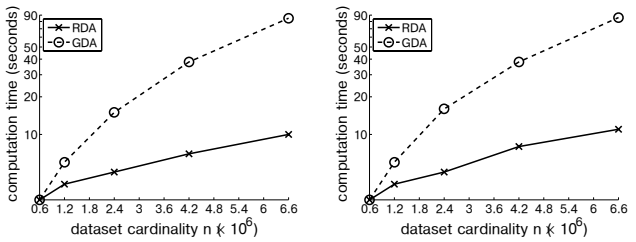
Input: an l-eligible table t_0, the privacy property l;
Output: the set Can^l of l-candidates for each color;
Method:

1. **Let** n_c be the number of colors in t_0;
2. **Let** S^C be the sequence of the colors in the non-increasing order of their cardinality;
3. **Let** C_i, n_i ($i \in [1, n_c]$) be the i^{th} color and its cardinality;
4. Compute the weighted-rank for each tuple in the table t_0;
5. Sort the tuples in each color in ascending order of their weighted-rank values;
6. **While** $(n_c \geq l)$
7. Let C_i be the color with the most incomplete identifiers;
8. **For** each incomplete identifier $id_{i,a}$ in C_i
9. Create can_{ia} by selecting $l-1$ *incomplete* identifiers from the first $l-1$ colors that minimize the QI-distance between their first and $id_{i,a}$;
10. **For** each incomplete identifier $id_{j,b}$
11. Create can_{jb} by selecting $l-1$ identifiers with minimal QI-distance from compatible can_{ia} with the least average QI-distance;
12. **For** $i = 1$ to n_c
13. **For** $w = 1$ to $l-1$
// Create the l-candidate can_i for C_i based on the $can_{ia}(a \in [1, n_i])$
14. Create in can_i its w^{th} candidate: $\bigcup_{a=1}^{n_i}($ the w^{th} identifier in $can_{ia})$;
15. **Let** $Can^l = \{can_i : 1 \leq i \leq n_c\}$;
16. **Return** Can^l;

Table 7: The GDA Algorithm

Discernibility Metric (DM) [4] and Query Workload Error (QWE, which is a utility metric that depends on quasi-identifier values) [14].

4.1 Computation Overhead



(a) SAL Dataset (b) OCC Dataset

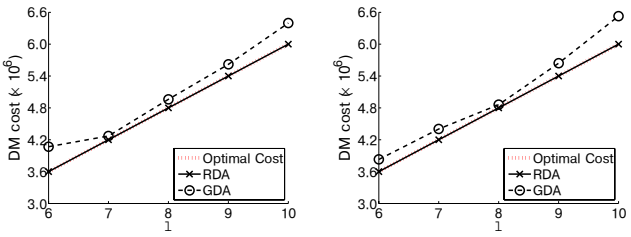
Figure 1: Execution Time vs. Dataset Cardinality n

Figure 1 illustrates the computation time of both of our algorithms on both datasets against the dataset cardinality n. We generate n-tuple datasets by synthesizing $\frac{n}{600k}$ copies of OCC, SAL respectively (Reminder that both OCC and SAL contain 600k tuples). We set $l = 8$ for this set of experiments, and conduct the experiment 100 times and then take the average. From the results, it is clear that both of our algorithms are practically efficient, and the computation time increases slowly with n. The RDA algorithm is slightly more efficient than GDA. This is because, when selecting candidates for each identifier, RDA considers the $l-1$ colors with the most incomplete identifiers while GDA considers the $l-1$ colors whose incomplete identifiers have the least QI-distances. Therefore, the more complex computation required by the GDA algorithm results in slightly more overhead than RDA.

Comparing to Results in [28] In contrast to the results reported in [28], both of our algorithms are more efficient, while the RDA algorithm requires significantly less time than that in [28]. Although not reported here due to space limitations, we have also investigated the computation time against l as well as the number of QI-attributes. Both algorithms are insensitive to these two parameters. This is as expected since the computation complexity of both algorithms only depends on the cardinality of dataset n.

4.2 Data Utility

We first conduct a set of experiments on the original SAL and OCC dataset to evaluate the utility of released tables measured by the *DM* metric. Figure 2 shows the DM cost (the lower cost the better utility) of each algorithm against l. From the results, we can see that the DM cost of our RDA algorithm is very close to the optimal cost (calculated using a separate algorithm), while the DM cost of the GDA algorithm is only slightly higher than the optimal cost. This is as expected, because the RDA algorithm is specifically designed for a general-purpose utility metric that aims to minimize the size of each anonymized group regardless of actual quasi-identifier values, whereas the GDA algorithm will attempt to minimize the QI-distance (the assignment of weight and rank for the GDA algorithm is described below).

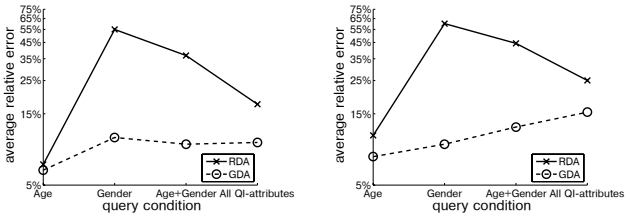


(a) SAL Dataset (b) OCC Dataset

Figure 2: Data Utility Comparison: DM Cost vs. l

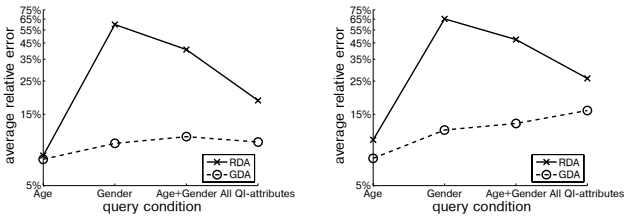
Following [28], we then evaluate the query workload error (QWE) by answering count queries. The intention is to compare our algorithms with a utility metric that depends on the actual quasi-

identifier values. For this purpose, predicates on QI-attributes are constructed on *Age*, *Gender*, with an *and* operations between them, and with an *and* operations between all the QI-attributes, respectively. We set *weight* to be 1,10000,1, and 1 for *Age*, *Gender*, *Education*, and *Birthplace*, respectively. By processing 1000 randomly-generated queries for each type of predicates, we intend to investigate how well the released table preserves the R_{qs} relation. For each query, we first obtain its accurate answer *acc* from the original micro-data table, and then adopt the approximation technique in [14] to compute the approximate answer *app* from the released table output by our algorithms. The error of an approximate answer is formulated as $\frac{|acc-app|}{max\{acc,\delta\}}$ [28], where δ is set to 0.5% of the dataset cardinality. Then, the average error of all queries is taken as the QWE.



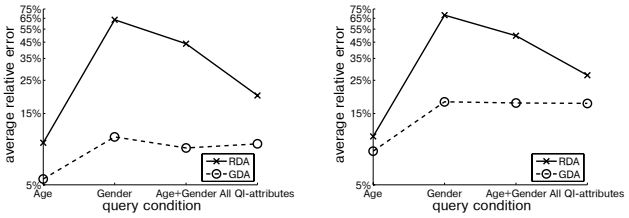
(a) SAL Dataset (b) OCC Dataset

Figure 3: Data Utility Comparison: Query Accuracy vs. Query Condition($l = 6$)



(a) SAL Dataset (b) OCC Dataset

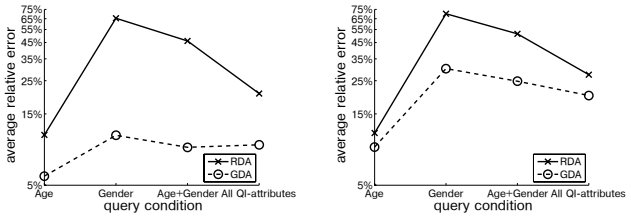
Figure 4: Data Utility Comparison: Query Accuracy vs. Query Condition($l = 7$)



(a) SAL Dataset (b) OCC Dataset

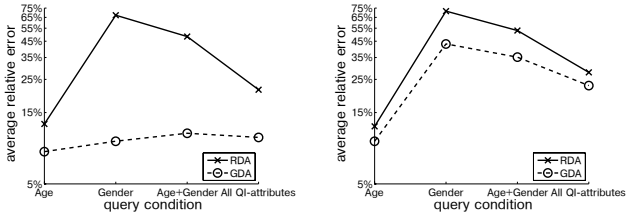
Figure 5: Data Utility Comparison: Query Accuracy vs. Query Condition($l = 8$)

Figures 3, 4, 5, 6, and 7 show the average relative error against different types of predicates for $l = 6, 7, 8, 9$ and 10 respectively. Compared to RDA, GDA now has better utility, which is as expected since GDA does consider the actual quasi-identifier values in generating the identifier partition, as mentioned in Section 3.



(a) SAL Dataset (b) OCC Dataset

Figure 6: Data Utility Comparison: Query Accuracy vs. Query Condition($l = 9$)



(a) SAL Dataset (b) OCC Dataset

Figure 7: Data Utility Comparison: Query Accuracy vs. Query Condition($l = 10$)

Particularly, the average relative error for querying on SAL and OCC with *Gender* as the only query condition for $l = 8$ is reduced from 64%, 69% (of RDA) to 10%, 18%, respectively. Finally, although not reported here due to space limitations, the utility result of our algorithms measured by QWE are close to the results reported in [28] (no result based on DM was reported there).

5. DISCUSSION

Possible Extensions In this paper, we have focused on applying the self-contained property on l-candidates to build sets of identifier partitions satisfying the l-cover property, and hence to construct the SGSS. However, there may in fact exist many other methods to construct the SGSS, which will lead to potential directions of future work. First, there are different ways for building the l-candidates for each color. As discussed above, theoretically any subset of $\mathcal{I}$ satisfying the constraints shown in Lemma 3 can be a valid candidate for a color, and $l - 1$ such subsets together with that color will form a valid l-candidate for that color if they satisfy the constraints shown in Theorem 2. Second, once l-candidates are given, there still exist different ways, including applying the self-contained property, for constructing sets of identifiers to satisfy the l-cover property. Third, even the l-cover property is not necessarily the only valid way for directly building the SGSS. Finally, although we have focused on l-diversity and the utility measures DM and QWE, the principle of decoupling utility optimization from privacy preservation can potentially be applied to other privacy applications to yield efficient solutions.

Practicality of Our Approach We have demonstrated the practicality of our approach by showing through complexity analysis and experiments that our proposed algorithms are efficient enough to be applied to real world applications. It is important to note that it would be unfair to compare the performance of our algorithms to many existing algorithms that ignore the issue of privacy breaches caused by adversarial knowledge about algorithms [23, 10]. As to utility, as discussed earlier, our proposed algorithms produce results comparable to existing methods. We believe the flexibility of our

approach may lead to other algorithms with further improved utility. For the QWE metric, note that our experiments only evaluate the QWE cost on the minimal self-contained partition. The utility may be increased by fine-tuning the weight information for each quasi-identifier, and by optimizing among the family set. We will conduct more experimental comparisons in terms of performance and utility between our algorithms and the traditional approaches in our future work.

The Focus on Syntactic Privacy Principles We have focused on syntactic privacy principles and methods, such as l-diversity and generalization, in this paper. However, the general approach of decoupling utility optimization from privacy preservation is not necessarily limited to such a scope. In particular, one interesting issue is to consider its applicability to differential privacy [6], which is being accepted as one of the strongest privacy models and extended to privacy preserving data publishing [17]. On the other hand, since most existing approaches that ensure differential privacy are random noise-based and are suitable for specific types of statistical queries, we have regarded this direction as future work.

6. RELATED WORK

The privacy preserving issue has received significant attentions in many different domains, including micro-data release [9, 22, 6], mobile network [8], social network [7, 20], and other web applications [3, 5]. In the context of privacy-preserving micro-data release, since the introduction of the k-anonymity concept [22], much effort has been made on developing efficient privacy-preserving algorithms [1, 2, 22, 13], Meanwhile many other models are also proposed to enhance the k-anonymity model, such as l-diversity [19], t-closeness [16], differential privacy [6], and so on. In particular, differential privacy [6] aims to achieve the goal that the probability distribution of any disclosed information should be similar enough regardless of whether that disclosed information is obtained using the real database, or using a database without any one of the existing records. Our future work will attempt to apply the proposed approach to other domains and privacy properties.

While many existing work assume the released generalization to be the only source of information available to an adversary, recent work [29, 25] show that this is actually not the case. In addition to the information that can be obtained from the actual released generalization, adversaries may also infer more information about private data by reasoning about how the generalization algorithm optimizes the data utility while satisfying the privacy property. With such extra inferred information, the adversary may eventually violate the privacy property. In the work of [29] [25], the authors discover the above problem and correspondingly introduce models and algorithms to address the issue. However, the method in [25] is still vulnerable to algorithm-based disclosure [11], whereas the one in [29] incurs a prohibitive complexity. To improve the efficiency, a so-called exclusive strategy is proposed in [30] to penalize the cases where a recursive process is required to compute the adversarial mental image about the micro-data table. More generally, a k-jump strategy is proposed in [18] to penalize such cases where with more control in the sense that only k, instead of all, generalization functions will be skipped. Despite the improved efficiency, most of those methods are still impractical due to the high complexity.

We have previously reported the preliminary idea of our proposed approach and the concept of l-cover in [31], which has been reviewed in Section 1 and Section 2.1 to make our paper more self-contained. However, no concrete methods for building identifier partitions that can satisfy the l-cover property was reported in [31], which is the main focus of this paper. Finally, the authors of [28] introduce algorithms that share the same spirit with our algorithms, and can achieve similar performance (more precisely, their algorithms are slightly less efficient than ours since their time complexity is $O(n^2 logn)$). In fact, under slight modification, their algorithms, such as ACE algorithm which is originally intended to publish dynamic datasets[27], can be regarded as another instantiation of our model and approach. This further confirms the correctness and flexibility of our approach.

7. CONCLUSION

In this paper, we have proposed a *privacy streamliner* approach for privacy-preserving applications. We reported theoretical results required for instantiating this approach in the context of privacy-preserving micro-data release using public algorithms. We have also designed three such algorithms by following the proposed approach, which not only yield practical solutions by themselves but also reveal the possibilities for a large number of algorithms that can be designed for specific utility metrics and applications. Our experiments with real datasets have proved our proposed algorithms to be practical in terms of both efficiency and data utility. Our future work will apply the proposed approach to other privacy-preserving applications and privacy properties in order to develop efficient algorithms.

Acknowledgment

The authors thank the anonymous reviewers for their valuable comments. The authors also thank Prof. Xiaokui Xiao (Nanyang Technological University, Singapore) for his help regarding experiments. This material is based upon work partially supported by Natural Sciences and Engineering Research Council of Canada under Discovery Grant N01035, Fonds de recherche sur la nature et les technologies, and Canada Graduate Scholarship.

8. REFERENCES

[1] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. k-anonymity: Algorithms and hardness. Technical report, Stanford University, 2004.

[2] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Anonymizing tables. In *ICDT '05*, pages 246–258, 2005.

[3] Michael Backes, Goran Doychev, Markus Dürmuth, and Boris Köpf. Speaker recognition in encrypted voice streams. In *ESORICS '10*, pages 508–523, 2010.

[4] Roberto J. Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *ICDE '05*, pages 217–228, 2005.

[5] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *IEEE Symposium on Security and Privacy '10*, pages 191–206, 2010.

[6] Cynthia Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006.

[7] Philip W. L. Fong, Mohd Anwar, and Zhen Zhao. A privacy preservation model for facebook-style social network systems. In *ESORICS '09*, pages 303–320, 2009.

[8] Julien Freudiger, Mohammad Hossein Manshaei, Jean-Pierre Hubaux, and David C. Parkes. On non-cooperative location privacy: a game-theoretic analysis. In *CCS '09*, pages 324–337, 2009.

[9] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):14:1–14:53, June 2010.

[10] Benjamin C. M. Fung, Ke Wang, and Philip S. Yu. Top-down specialization for information and privacy preservation. In *ICDE '05*, pages 205–216, 2005.

[11] Xin Jin, Nan Zhang, and Gautam Das. Algorithm-safe privacy-preserving data publishing. In *EDBT '10*, pages 633–644, 2010.

[12] K.Kenthapadi, N.Mishra, and K.Nissim. Simulatable auditing. In *PODS*, pages 118–127, 2005.

[13] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: Efficient fulldomain k-anonymity. In *SIGMOD '05*, pages 49–60, 2005.

[14] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE '06*, page 25, 2006.

[15] L.H.Cox. Suppression, methodology and statistical disclosure control. *J. of the American Statistical Association*, pages 377– 385, 1995.

[16] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE '07*, pages 106–115, 2007.

[17] Ninghui Li, Wahbeh H. Qardaji, and Dong Su. Provably private data anonymization: Or, k-anonymity meets differential privacy. *CoRR*, abs/1101.2604, 2011.

[18] Wen Ming Liu, Lingyu Wang, and Lei Zhang. k-jump strategy for preserving privacy in micro-data disclosure. In *ICDT '10*, pages 104–115, 2010.

[19] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.

[20] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy '09*, pages 173–187, 2009.

[21] Steven Ruggles, Matthew Sobek, J. Trent Alexander, Catherine Fitch, Ronald Goeken, Patricia Kelly Hall, Miriam King, and Chad Ronnander. Integrated public use microdata series: Version 3.0. http://ipums.org, 2004.

[22] Pierangela Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, 2001.

[23] Ke Wang, Philip S. Yu, and Sourav Chakraborty. Bottom-up generalization: A data mining solution to privacy protection. In *ICDM '04*, pages 249–256, 2004.

[24] Raymond Chi-Wing Wong and Ada Wai-Chee Fu. *Privacy-Preserving Data Publishing: An Overview*. Morgan and Claypool Publishers, 2010.

[25] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality attack in privacy preserving data publishing. In *VLDB '07*, pages 543–554, 2007.

[26] Xiaokui Xiao and Yufei Tao. Anatomy: simple and effective privacy preservation. In *VLDB '06*, pages 139–150, 2006.

[27] Xiaokui Xiao and Yufei Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *SIGMOD '07*, pages 689–700, 2007.

[28] Xiaokui Xiao, Yufei Tao, and Nick Koudas. Transparent anonymization: Thwarting adversaries who know the algorithm. *ACM Trans. Database Syst.*, 35(2):1–48, 2010.

[29] Lei Zhang, Sushil Jajodia, and Alexander Brodsky. Information disclosure under realistic assumptions: privacy versus optimality. In *CCS '07*, pages 573–583, 2007.

[30] Lei Zhang, Lingyu Wang, Sushil Jajodia, and Alexander Brodsky. Exclusive strategy for generalization algorithms in micro-data disclosure. In *DBSec '08*, pages 190–204, 2008.

[31] Lei Zhang, Lingyu Wang, Sushil Jajodia, and Alexander Brodsky. L-cover: Preserving diversity by anonymity. In *SDM '09*, pages 158–171, 2009.

Appendix

A. Definitions of Cover, l-cover and Their Example

DEFINITION 4 (COVER). *We say* $ids_1, ids_2 \subseteq \mathcal{I}$ *are* cover *for each other with respect to a set* $S^P \subseteq ss^l$*, if*

- $ids_1 \cap ids_2 = \emptyset$*, and*
- *there exist a bijection* $f : ids_1 \rightarrow ids_2$ *such that for any* $ids_x \in P_i$*,* $P_i \in S^P$*, there always exists* $P_j \in S^P$ *satisfying* $ids_x \setminus (ids_1 \cup ids_2) \cup f(ids_x \cap ids_1) \cup f^{-1}(ids_x \cap ids_2) \in P_j$ *[31].*

DEFINITION 5 (L-COVER). *We say a set* $S^P \subseteq ss^l$ *satisfies the* l-cover *property, if every color* C *has at least* $l-1$ *covers* $ids_i (i \in [1, l-1])$ *with the bijections* f_i *satisfying that*

- *for any* $id \in C$*, each* $f_i(id)$ $(i \in [1, l-1])$ *is from a different color, and*
- *for any* $ids_x \in P$ *and* $P \in S^P$*, we have* $|ids_x \cap C| = |ids_x \cap ids_i| (i \in [1, l-1])$ *[31].*

EXAMPLE 5. *Continuing Example 2 and considering* $S^P = \{P_1, P_3\}$*, the colors* $C_1 = \{id_1, id_2\}$ *and* $C_2 = \{id_3, id_4\}$ *provide cover for each other, since for* C_1 *we have* $f_1(id_1) = id_3$ *and* $f_1(id_2) = id_4$*, and for* C_2 *we have* $f_2(id_3) = id_1$ *and* $f_2(id_4) = id_2$*. Further,* S^P *satisfies the* l*-cover property where* $\{C_1, C_2\}$ *is the* l*-cover of both* C_1 *and* C_2*.*

Similarly, for $S^P = \{P_2, P_3\}$*,* C_1 *and* C_2 *provide cover for each other since for* C_1 *we have* $f_1(id_1) = id_4$ *and* $f_1(id_2) = id_3$*, and for* C_2 *we have* $f_2(id_3) = id_2$ *and* $f_2(id_4) = id_1$*. Further,* S^P *also satisfies the* l*-cover property.* ⊡

B.1. Proof of Lemma 1

PROOF. To prove the lemma, we first show the procedure *l-candidate-to-*P^{lm} in Table 9 based on the self-contained property to construct its minimal self-contained partition.

Input: an l-eligible table t_0, a collection of l-candidates Can^l
Output: the minimal self-contained partition;
Method:

1. Create a set of anonymized groups $S^G = \emptyset$;
2. **For** each color C_i
3. **For** each $id_{i,a} \in C_i$
4. Create in S^G a anonymized group $G_{i,d} = \{id_{i,a}\} \bigcup_{u=1}^{l-1} \{f_{i,u}(id_{i,a})\}$;
5. Merge the anonymized groups which have common identifiers to build minimal self-contained partition (P^{lm});
6. **Return** P^{lm};

Table 9: procedure: l-candidate-to-P^{lm}

Then, we show that $P^{lm} \in ss^l$. As shown in Table 9, to satisfy the self-contained property, for each identifier $id_{i,a}$ in each color C_i, the identifiers to which $id_{i,a}$ is mapped in each of the l-1 candidates should be in the same final anonymized group. We call such set of identifiers ,$G_{i,a} = \{id_{i,a}\} \bigcup_{u=1}^{l-1} \{f_{i,u}(id_{i,a})\}$, for a^{th} identifier in color C_i is *transient group*. Obviously, each transient group itself satisfies entropy l-diversity. Furthermore, based on the Definition 2, for any color C_i in the micro-data table, if an identifier $id_{i,a}$ in C_i is in the final anonymized group, then its whole transient group $G_{i,a}$ will be in the final anonymized group. In other words, in any final anonymized group G, the ratio of any identifier in any C_i associated with the sensitive value S_i equals to $\frac{|n_{C_i}|}{|n_{C_i}| \times l + \delta}$ where $\delta \geq 0$ and $|n_{C_i}|$ is the number of identifiers from color C_i in the anonymized group. Therefore, it is less than or equal to $\frac{|n_{C_i}|}{|n_{C_i}| \times l} = \frac{1}{l}$. Thus, each anonymized group in minimal self-contained partition satisfies l-diversity, so does the minimal self-contained partition. We have thus proved that $P^{lm} \in ss^l$.

Next, consider the $l-1$ covers for each color $C_i \in S^C$. Without loss of generality, we rewrite its corresponding l-candidate as $can_i^l = \{C_i, ids_{i,1}, ids_{i,2}, \ldots, ids_{i,l-1}\}$ so that C_i is the first element, we show that for the set of identifier partition P^{lm} ($|P^{lm}| = 1$), $ids_{i,1}, ids_{i,2}, \ldots ids_{i,l-1}$ are $l-1$ covers of C_i. By Definition 4, C_i and $ids_{i,u} (u \in [1, l-1])$ should satisfy following two conditions:

- $C_i \cap ids_{i,u} = \emptyset$, and
- there exists a bijection $f_{i,u} : C_i \rightarrow ids_{i,u}$ satisfying that for any $ids_x \in P^{lm}$, $ids_{x'} = ids_x \backslash (C_i \cup ids_{i,u})) \cup f_{i,u}(ids_x \cap C_i) \cup f_{i,u}^{-1}(ids_x \cap ids_{i,u}) \in P^{lm}$.

The first condition is satisfied by the definition of l-candidate. For the second condition, let the bijection $f_{i,u}$ be the corresponding bijection for $ids_{i,u}$ in the l-candidate can_i^l. It is obvious that $ids_{x'} = ids_x$. Therefore, the second condition also holds.

Finally, we further show that the previous $l-1$ covers of C_i satisfy the following three conditions defined in the definition of l-cover.

- $\forall (u \neq w), ids_{i,u} \cap ids_{i,w} = \emptyset$, and
- $\forall (id \in C_i)$, each $f_{i,u}(id)$ $(u \in 1, l-1])$ is from different color.
- $\forall (ids \in P^{lm}), |ids \cap C_i| = |ids \cap ids_{i,u}|$ $(u \in [1, l-1])$.

The first two conditions follow directly from the definition of l-candidate. The last condition is satisfied by the property of self-contained. In other words, given such P^{lm}, all colors have their l-covers, therefore, P^{lm} satisfies l-cover property. Thus we have proved the lemma. □

B.2. Proof of Lemma 2

Proof. We prove by contradiction. First assume that there exist $G \in P$ and $ids_i \in P^{lm}$, such that $G \cap ids_i \neq \emptyset$ and $ids_i - G \neq \emptyset$. Then, due to $ids_i - G \neq \emptyset$, there must exist identifier $id_o \in ids_i$ such that $id_o \notin G$. Assume that $id_o \in G'$, where $(G' \in P) \wedge (G' \neq G)$. Moreover, due to $G \cap ids_i \neq \emptyset$, there also exists identifier $id_i \in ids_i$ such that $id_i \in G$. Thus there exist id_o and id_i which is a pair of identifiers for some bijection in Can^l, and $G \cap \{id_o, id_i\} = \{id_i\}$ and $G' \cap \{id_o, id_i\} = \{id_o\}$. However, By definition of *self-contained*, it has the following transitive property. That is, if $\{id_1, id_2\}, \{id_2, id_3\}, \ldots, \{id_{a-1}, id_a\}$ each pair satisfies that there exists bijections for the set of l-candidates such that $f_{i-1,i}(id_{i-1}) = id_i$ or $f_{i,i-1}(id_i) = id_{i-1}$. Then for any self-contained anonymized group G, either $G \cap \cup_{i=1}^{a}(id_i) = \emptyset$ or $G \supseteq \cup_{i=1}^{a}(id_i)$. Thus by definition, since $id_o \in ids_i$ and $id_i \in ids_i$, $\forall (G \in P), G \cap \{id_o, id_i\} = \emptyset$ or $G \cap \{id_o, id_i\} = \{id_o, id_i\}$.

Therefore, neither G nor G' satisfies self-contained, so does P, leading to a contradiction. □

B.3. Proof of Theorem 1

Proof. *First, we prove that any self-contained identifier partition P satisfies l-cover property.*

We first show that $P \in ss^l$. Note that the privacy model l-diversity satisfies the monotonicity property. That is, for any two anonymized groups G_1 and G_2 satisfying l-diversity, the final anonymized group derived by merging all tuples in G_1 and in G_2 satisfies l-diversity [24]. Based on Lemma 2, each anonymized group G in P satisfies $G = \cup_{X \subseteq \{1,2,\ldots,k\}} ids_X$. Therefore, each anonymized group G satisfies l-diversity, so does P.

Then we have proved that, given the l-candidate can_i^l of certain color C_i, the $can_i^l \backslash \{C_i\}$ are the $l-1$ covers of C_i for P, similar to the proof of Lemma 1.

Finally, the set of $l-1$ set of identifiers $can_i^l \backslash \{C_i\}$ are $l-1$ covers of color C_i which satisfy the three conditions of l-cover definition.

Second, we prove any family set satisfies the l-cover property.

We first show that $\forall (P \in S^{fs}), P \in ss^l$. Since the privacy model l-diversity satisfies the monotonicity property [24], based on the definition of family set, it is clear that the table generalization corresponding to each identifier partition in S^{fs} satisfies l-diversity.

Similar with previous proofs, for each color $C_i \in S^C$ and its corresponding l-candidate $can_i^l = \{C_i, ids_{i,1}, ids_{i,2}, \ldots, ids_{i,l-1}\}$, we have proved that for the family set S^{fs}, $ids_{i,1}, ids_{i,2}, \ldots, ids_{i,l-1}$ are the $l-1$ covers of C_i. Moreover, these $l-1$ covers of C_i satisfy the three conditions of l-cover. This completes the proof. □

B.4. Proof of Lemma 3

Proof. By the definition 1, C and ids should satisfy the following two conditions:

- $C \cap ids = \emptyset$ and $|C| = |ids|$;
- there exists a bijection $f : C \rightarrow ids$, such that $\forall (id \in C)$, id and $f(id)$ are from different colors.

The first condition follows directly from the condition of the lemma. Since $|C| = |ids|$, there must exist bijection $f : C \rightarrow ids$. Moreover, since $C \cap ids = \emptyset$, $\forall (id_x \in ids)$, $id_x \notin C$, by the definition of color, id_x has sensitive value other than it of color C. In other words, id_x must belong to the other color C' other than C. Therefore, The second condition is also satisfied, which completes the proof. □

B.5. Proof of Lemma 4

Proof. The first constraint in the lemma respectively guarantees the first condition of definition 1. Consider the second condition. Since $|ids_1| = |ids_2|$, there must exist bijections between ids_1 and ids_2. Assume that the second condition of definition 1 does not hold. Then there must exist at least $|ids_1| + 1$ number of identifiers in $ids_1 \cup ids_2$ with identical sensitive value, which is in contradiction with the second constraint in lemma. Therefore, the second condition of definition 1 also satisfies. Since the two conditions both hold, the proof is complete. □

B.6. Proof of Theorem 2

Proof. To prove the theorem, we should show that any two sets of identifiers from the sets C and ids_x $(x \in [1, l-1])$ are candidate for each other. The fact that C and each ids_x $(x \in [1, l-1])$ are candidate follows the Lemma 3, while the fact any two ids_x, ids_y $((x, y \in [1, l-1]) \wedge (x \neq y))$ are candidate follows the Lemma 4. This completes the proof. □