Anonymizing Set-Valued Data by Nonreciprocal Recoding*

Mingqiang Xue* Panagiotis Karras# Chedy Raïssi◊ Jaideep Vaidya# Kian-Lee Tan*
*National University of Singapore #Rutgers University ◊INRIA, Nancy Grand-Est, France

ABSTRACT

Today there is a strong interest in publishing set-valued data in a privacy-preserving manner. Such data associate individuals to sets of values (e.g., preferences, shopping items, symptoms, query logs). In addition, an individual can be associated with a sensitive label (e.g., marital status, religious or political conviction). Anonymizing such data implies ensuring that an adversary should not be able to (1) identify an individual's record, and (2) infer a sensitive label, if such exists. Existing research on this problem either perturbs the data, publishes them in disjoint groups disassociated from their sensitive labels, or generalizes their values by assuming the availability of a generalization hierarchy. In this paper, we propose a novel alternative. Our publication method also puts data in a generalized form, but does *not* require that published records form disjoint groups and does not assume a hierarchy either; instead, it employs *generalized bitmaps* and recasts data values in a *nonreciprocal* manner; formally, the bipartite graph from original to anonymized records does *not* have to be composed of disjoint complete subgraphs. We configure our schemes to provide popular privacy guarantees while resisting attacks proposed in recent research, and demonstrate experimentally that we gain a clear utility advantage over the previous state of the art.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*Security, integrity, and protection*; H.2.8 [**Database Management**]: Database applications—*Data mining*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

Keywords

privacy, anonymization, set-valued data, nonreciprocal recoding

1. INTRODUCTION

Assume a data vendor wants to publish a data set $\mathcal{D}$ of *set-valued* data, where a record $r_i \in \mathcal{D}$ consists of a set of items, $r_i = \{o_1, \ldots, o_n\}$, drawn from a universe $\mathcal{I}$. Moreover, each record r_i can potentially be associated with a *sensitive label* s_i, denoting a piece of information such as marital status, sexual orientation, political conviction, or income group. Several real-world data sharing problems can be formulated by this model, even when the data does not originally arise in a set-valued form; the set-valued data may describe data originally presented as a bipartite graph matching, e.g., users to preferences, or even relational database data, where each r_i contains a tuple's attribute values.

Publishing such data in their original form, even without identifiers, compromises privacy. Thus, there is a need to transform the data in a way that preserves information while alleviating privacy threats. There are two desiderata: First, a record r_i should not be clearly distinguishable from other records, leading to direct exposure of its subject's identity. Second, a sensitive label, when present, should not be easily associable to a certain individual.

ID	Jogging	Swimming	Tennis	Soccer	Religion
r_1	1	1	0	0	Christian
r_2	0	1	1	0	Christian
r_3	1	1	0	1	Muslim
r_4	0	1	1	1	Buddhist
r_5	1	1	1	0	Buddhist
r_6	1	0	1	1	Muslim

Table 1: Original set-valued data

Table 1 shows an example of set-valued data about the sport preferences and religious affiliation of certain individuals. For each record r_i, a value of 1 at position j indicates that item j is present in r_i, whereas a 0 indicates absence. Each record in Table 1 is uniquely identifiable by its bitmap. Thus, an adversary who is aware of the itemset this bitmap represents can infer an individual's presence in the data and sensitive label. For example, if Alex knows that Barbara likes *only* jogging and swimming, he can identify her record as r_1, and also infer that she is a Christian. We aim to publish the data in a form that prevents such disclosures.

Previous research has noted the importance of transforming set-valued data for privacy-preserving publication [8, 7, 9, 25, 23, 14, 2, 4], but employed transformation operations mostly unsuitable for the nature of the data at hand. Works such as [8, 7, 2] employ *random perturbation*, adding noise to the data. Yet random perturbation provides no information about the extent to which a particular record has been perturbed, and renders *outliers* vulnerable to an adversary with external knowledge [9]. On the other hand, *syntactic* data transformations, such as those in [9, 25, 23, 14, 4], recast the data so that they maintain a consistency to their original form, despite the obfuscation they undergo [17, 3]. Among them, [25] strives for a privacy objective by selectively *suppressing* some items (i.e., withholding them from publication); more refined *generalization* methods are employed in [23, 14, 4], based on the

*Work supported by Singapore's A*STAR grant 1021580037 and a Rutgers Business School Research Resources Committee grant.

KDD'12, August 12–16, 2012, Beijing, China.

assumption that a *generalization hierarchy* is applicable on the data items in $\mathcal{I}$. However, such hierarchies are not always available in practice; for example, in the case where the set-valued data represent query logs, their construction is, by itself, a non-trivial problem [14]. The experimental studies of [23, 14, 4] use ad hoc hierarchies, which are clearly arbitrary. Another suggestion [15] adds and suppresses query log objects so as to render users indistinguishable by a loose measure of user similarity. Last, [9] publishes exact (public) itemsets in groups, along with a separate summary table of (private) sensitive labels for each group. Unfortunately, this *transparent publication* method is vulnerable to attacks by adversaries with background knowledge of some sensitive associations: an adversary who sees the *exact* items in a record can carry out a chain of reasoning leading to an inference of a sensitive label, which would be hindered if these items were obfuscated by generalization [4]. Besides, the publication model of [9] does not provide protection against identity disclosure as generalization does [10].

ID	Jogging	Swimming	Tennis	Soccer	Religion
r_1	1	1	*	*	Christian
r_3	1	1	*	*	Muslim
r_5	1	1	*	*	Buddhist
r_2	*	*	1	*	Christian
r_4	*	*	1	*	Buddhist
r_6	*	*	1	*	Muslim

Table 2: Data anonymized by suppression

A conventional syntactic anonymization method may partition records in distinct groups, where all records in a group are interchangeable with each other. Table 2 shows an example along these lines, applied on the data of Table 1. The privacy objective is that, for each original record r_i, there should be (at least) *three* records that may be an obfuscated form of (or *match*) r_i's bitmap, and three different sensitive labels that may be associated to r_i. To achieve this objective, one can suppress some bit values, so that it is not disclosed whether the item in question is present or not, and form two distinct groups, with records in the same group having indistinguishable bitmaps and different sensitive labels. Yet even in this simple example, a significant number of suppressions is required to achieve the desired privacy, compromising the utility of the data.

However, it is *not* necessary that privacy be achieved via the formation of distinct groups, as above, while the obfuscation mechanism does *not* have to be suppression (or generalization along an arbitrary hierarchy) either. In this paper, we propose an alternative model. Our scheme ensures that each original record *matches* a group of generalized records, yet this effect is *not* brought about by creating groups of records recast so as to be identical to each other; instead, original records match anonymized ones in a *nonreciprocal* manner: when an original record s matches the anonymized form t' of another record t, then *it is not necessary* that t also matches s'. We recast each record's bitmap r_i by only altering some of its bits (i.e., adding or deleting items), and publish a *base bitmap* r'_i along with a *distance bitmap* d_i, and an *edit-distance threshold* t_i. In order to detect pairs of records of small Hamming distance, which can be easily recast so as to match each other, we employ the *Gray order* of bitmaps, enhanced by applying an approximation algorithm for the Traveling Salesman Problem (TSP).

Table 3 shows a way of publishing the data of Table 1 by our method that achieves the same privacy as the publication in Table 2, but much higher utility. For each original record r_i, the table shows its anonymized bitmap r'_i, a sensitive label, a distance bitmap d_i that indicates the positions where an error may occur in r'_i, and an edit-distance threshold t_i that indicates the maximum possible number of errors among the positions indicated in d_i.

For example, the distance bitmap for r'_5 is 0011, denoting that an

ID	Jogging	Swimming	Tennis	Soccer	Religion	d_i	t_i
r'_1	1	1	0	1	Christian	1 0 1 1	2 bits
r'_2	1	1	1	0	Christian	1 1 0 1	2 bits
r'_3	0	1	1	1	Muslim	1 0 1 1	2 bits
r'_4	0	1	1	1	Buddhist	1 1 0 1	2 bits
r'_5	1	1	0	0	Buddhist	0 0 1 1	1 bit
r'_6	1	1	1	0	Muslim	0 1 1 1	2 bits

Table 3: Data anonymized by our method

original record r represented by r'_5 *may* differ from it at the 3rd or 4th bit. The error threshold indicates that r may only differ from r'_5 by at most 1 bit, which reduces our options to *either* the 3rd bit, or the 4th, or none. Thus, three *possible worlds* [5] are defined, as r may be either 1100, or 1110, or 1101. In the first case, r is r_1, in the second case it is r_5, and in the third case it is r_3. We emphasize that *some* possible worlds might not correspond to any real record, yet *all* real records that have to match an anonymized one by our scheme are always found among the possible worlds.

Original	Matches	Anonymized	Matches
r_1	r'_1, r'_5, r'_6	r'_1	r_1, r_3, r_4
r_2	r'_2, r'_3, r'_4	r'_2	r_2, r_5, r_6
r_3	r'_1, r'_3, r'_5	r'_3	r_2, r_3, r_4
r_4	r'_1, r'_3, r'_4	r'_4	r_2, r_4, r_6
r_5	r'_2, r'_5, r'_6	r'_5	r_1, r_3, r_5
r_6	r'_2, r'_4, r'_6	r'_6	r_1, r_5, r_6

Table 4: Original/anonymized data correspondence

Table 4 shows the correspondence between original and anonymized records, i.e., the anonymized records in Table 3 that each original record in Table 1 is *compatible* with, and vice versa. As each anonymized record matches three original records, and *vice versa*, a privacy guarantee of 3-*anonymity* is achieved [20].

2. BACKGROUND AND RELATED WORK

Our work draws from two sources. The former involves the privacy-preserving sharing of set-valued data in general, while the latter is about the transformation by nonreciprocal recoding.

2.1 Anonymizing Set-valued Data

Research on preserving privacy in set-valued data has recently focused on transforming the data in a way that provides a generic privacy guarantee. The pioneering work in the field [11] transforms the data into a band matrix by permutating rows and columns in the original table, and forms anonymized groups on this matrix, offering the privacy guarantee that the probability of associating a record with a particular sensitive label does not exceed a threshold $\frac{1}{p}$. This method is augmented by two more approaches in [9]. The best performer in terms of both data utility and execution time is a scheme that interprets itemsets as Gray codes and sorts them by their Gray-code rank, so that consecutive records have low Hamming distance, facilitating group formation. Still, the publication model of [11, 9] publishes *exact* public items together with a summary of the frequencies of sensitive labels per group; this transparency renders it vulnerable to attacks by adversaries who are already aware of some associations and wish to infer others [4].

Another alternative [25] opts to selectively *suppress* some items, and ensures that an adversary can link an individual to (none, or) at least k records, with at most $h\%$ thereof sharing the same sensitive label; the h parameter is thus equivalent to $\frac{1}{p}$ in [11, 9]. However, in contrast to [11, 9], [25] assumes that an adversary's knowledge is limited to at most p items in a record. Besides, the suppression technique of [25] results in high information loss [4, 23].

More recently, [23, 14, 4] use hierarchy-based generalization to anonymize set-valued data, and provide privacy guarantees against

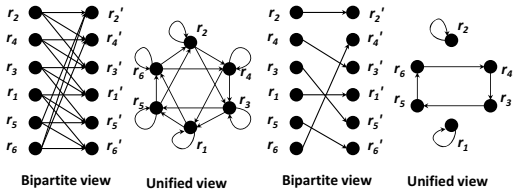
an adversary's capacity to link an individual to a small number of records [23, 14], or to confidently infer any sensitive item among the items in a record themselves [4]. However, a generalization hierarchy is not always applicable and/or available, and its construction is by itself a non-trivial problem [14]. In their experimental studies, [23, 14, 4] construct synthetic hierarchies. Under such a synthetic hierarchy, [14] applies its proposal on the anonymization of query logs. On the other hand, [15] anonymizes query logs, without assuming a generalization hierarchy over query objects; users are rendered indistinguishable according to a loose similarity measure, by adding and suppressing query objects.

All methods discussed above use *syntactic* transformations. Another line of research uses *random perturbation* to anonymize data [6, 21, 8, 7, 19, 2]. However, perturbation techniques can expose the privacy of *outliers* in a way that syntactic methods do not [9]. The sketch-based method of [1] tries to avoid such drawbacks, providing a guarantee that renders records hardly distinguishable from their k nearest neighbors. However, as it may not always be possible to satisfy this privacy condition, [1] resorts to suppressing outliers. Besides, perturbation-based transformations provide no information on how much a given record has been perturbed; in other words, they render data in an *inaccurate* form, hence limit the purposes they can be useful for [17]. On the other hand, syntactic transformations hamper the data's *precision*, but not its *accuracy*.

2.2 Nonreciprocal Recoding

As discussed, a syntactic transformation recasts the data by a still accurate representation, albeit imprecise and coarse, with an explicit margin of error. Past research [25, 11, 9, 23, 14] applied syntactic transformations under the premise that, for any two records s, t, if s is recoded so as to match t, then t should also be recoded so as to match s. This premise alone leads to *reciprocal recoding*. Past research has also assumed that the published records need to form disjoint groups, so that (the public parts of) *all* records in the same group are recoded so as to mutually match each other.

Nevertheless, this *reciprocity* assumption is not required by a privacy condition; it is redundant. This redundancy was noted by [3], observing that "there is no privacy reason" therefor. Contemporaneously, [12] revisited this question in the context of microdata anonymization, and noted that dropping the reciprocity assumption allows for improved data utility; the model of global $(1, k)$-anonymity [12] guarantees, by nonreciprocal recoding, that an individual is associated with at least k recoded records, as the popular k-anonymity model conventionally does using reciprocal recoding. Later, [24] observed that the techniques of [12] do not ensure that each such association is equiprobable, and provided a nonreciprocal-recoding algorithm that guarantees k associations of probability $\frac{1}{k}$, using randomization.



(a) Generalization graph (b) Sample assignment

Figure 1: Nonreciprocal recoding in graph view

We illustrate nonreciprocal recoding with two kinds of directed graphs. A *generalization graph* shows how the values of original records match those of anonymized records. A directed edge (r_i, r'_j) in a generalization graph indicates that the anonymized record r'_j should include original record r_i among its possible worlds.

Figure 1(a) shows the generalization graph for the example in the previous section. We present two views of this graph: a *bipartite* view, as well as as a *unified* view where a single node represents both the original record r_i and the anonymized record r'_i. For instance the fact that r'_1 matches r_1, r_3, and r_4 is represented by the edges (r_1, r'_1), (r_3, r'_1) and (r_4, r'_1), respectively.

The privacy principle of k-anonymity [20] requires that each original record r_i have *at least* k *equally probable* matches among anonymized records R'. Under the conventional reciprocity assumption, this property is easily satisfied by forming groups of k records mutually matching each other within each group. However, when we drop the reciprocity assumption, we need to spell out the requirements for k-anonymity to be satisfied. It has been shown by [12, 24] that, to achieve k-anonymity by nonreciprocal recoding, it suffices to ensure that each original record r_i has exactly k matches in R' (i.e., k outgoing edges in the generalization graph), and each anonymized record r'_i also has exactly k matches in R (i.e., incoming edges); of course the same effect can be achieved with any $k' > k$, but then k'-anonymity is attained. In other words, it suffices to ensure that the data's generalization graph is k*-regular*. From such a graph we can generate k disjoint assignments [24]. The generalization graph in Figure 1(a) is 3-regular, hence ensures 3-anonymity. In order to create a k-regular generalization graph, [24] suggests the method of *ring generalization*: given k, a generalization graph is constructed as a ring, linking each of n records, r_i, to itself and its $k-1$ successors by a given cyclical order. The generalization graph in Figure 1(a) is a ring generalization graph for the order $\{r_2, r_4, r_3, r_1, r_5, r_6\}$.

On the other hand, an *assignment graph* shows a particular one-to-one correspondence between original and anonymized records (i.e., an assignment); it provides the assumed identities of anonymized records, and may be used as a guide when assigning non-generalized attributes (e.g., sensitive labels) to them. An assignment graph is a subset of the generalization graph. Figure 1(b) shows a possible assignment for our example in bipartite and unified view. To satisfy the equal probability requirement of k-anonymity, we should ensure that each edge in a generalization graph is equally likely to participate in a chosen assignment. This result can be achieved by selecting one of k disjoint assignments uniformly at random. Furthermore, in order to resist attacks based on knowledge of some anonymized tuples' identities and/or of the algorithm itself, the set of k disjoint assignments to choose from is generated by a randomization scheme [24]. Each assignment is generated by iteratively extracting cycles from the generalization graph (in unified view) via random walks, until all records are covered. We illustrate a simple example of this process in Figure 2.

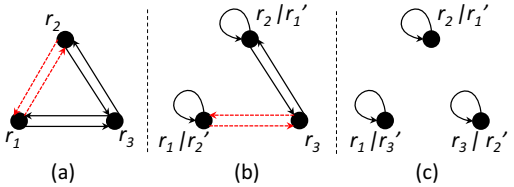


Figure 2: Iterative cycle extraction

Figure 2(a) depicts a 2-regular generalization graph for a data set of 3 records, and the first step of the process, which extracts the cycle $r_1 \rightarrow r_2 \rightarrow r_1$ by random walk. Then, the graph is updated to reflect the matching choices made so far. The node that originally stood for r_1 (and r'_1) now stands for r_1 and its match, r'_2, while the node that stood for r_2 (and r'_2) now stands for r_2 and r'_1; thus, the two chosen matches now appear as self-loops (Figure 2(b)). We can now proceed to extract another cycle, potentially destroying some of the previously selected matches (i.e., de-selecting

the self-loops of those nodes as matches of our choice) in case the new cycle passes through the same nodes, yet without ever reducing the number of matched nodes; some matches may be replaced by others, but no previously matched node is left orphan. Such a new cycle, namely $r_3 \rightarrow r_1/r'_2 \rightarrow r_3$, is shown in Figure 2(b). This cycle replaces the match of r_1 to r'_3, while it matches r_3 to r'_2 instead. With these new matches our task is completed, as all records in the generalization graph have been covered. Figure 2(c) shows the chosen *assignment graph*, composed of self-loop singletons.

Unfortunately, the time complexity of the randomization-based scheme in [24] is $O(kn^2)$. Noting this complexity, [24] suggests that this scheme be applied *on top* of a partitioning scheme, so as to improve the utility within each partition. We aim to provide a general-purpose anonymization scheme by nonreciprocal recoding that can be applied on a full set-valued data set (or any data set amenable to bitmap representation) and achieve high data utility.

3. DEFINITIONS AND PRINCIPLES

We consider a set-valued dataset $\mathcal{D} = (R, S)$ of n records. $R = \{r_1, \ldots, r_n\}$, where r_i is the non-sensitive part of record i and $S = \{s_1, \ldots, s_n\}$ is a set of sensitive labels of records, with the sensitive label of record i denoted as s_i. Each r_i is represented as a bitmap of b bits, where b is the cardinality of the universe I a record draws items from. The value of the bit at position j, $r_{i,j}$, denotes the presence or absence of the j^{th} item in I in/from r_i. We aim to obfuscate the non-sensitive parts of records, producing $R' = \{r'_1, \ldots, r'_n\}$, where r'_i is the anonymized version of r_i.

We say that an original record r_i and an obfuscated record r'_j *match* each other when r'_j is possibly an obfuscated from of r_i. We then define the privacy guarantees of k-anonymity [20] and ℓ-diversity [18] in the context of set-valued data as follows:

DEFINITION 1. *An* anonymized *set-valued data set $\mathcal{D}' = (R', S)$ satisfies* k-anonymity *with respect to the original data $\mathcal{D} = (R, S)$ iff each original record $r_i \in \mathcal{D}$ matches at least k published records in $\mathcal{D}'$, each having, from an adversary's perspective, equal probability (at most $\frac{1}{k}$) to be the* true match *of r_i. $\mathcal{D}'$ satisfies* ℓ-diversity *with respect to $\mathcal{D}$ iff each $r_i \in \mathcal{D}$ matches at least ℓ published records, each associated with a different sensitive label $s \in S$ and having equal probability (at most $\frac{1}{\ell}$) to be the* true match *of r_i.*

These guarantees ensure that an adversary knowing the non-sensitive part of all records, i.e. R, shall not be able to identify the *true match* of a record r_i (and its sensitive value) with probability higher than $\frac{1}{k}$ $\left(\frac{1}{\ell}\right)$. The twin problems of k-anonymization and ℓ-diversification for set-valued data call for satisfying these guarantees with a low reduction of the utility of the original data:

PROBLEM 1. *Given a data set $\mathcal{D} = (R, S)$,* transform *$\mathcal{D}$ to an anonymized form $\mathcal{D}'$ that satisfies k-anonymity (ℓ-diversity), maintaining as much of the data utility as possible.*

We describe a collection of matches encompassing a complete set of original and anonymized records as an *assignment*.

DEFINITION 2. *Given a set-valued data set $\mathcal{D} = (R, S)$ and an anonymized version thereof, $\mathcal{D}' = (R', S)$, an* assignment *α from $\mathcal{D}$ to $\mathcal{D}'$ is an one-to-one mapping from $\mathcal{D}$ to $\mathcal{D}'$, denoted as $\alpha = \{(r_{i_1}, r'_{j_1}), \ldots, (r_{i_n}, r'_{j_n})\}$, such that each $r_i \in \mathcal{D}$ is mapped to exactly one $r'_j \in \mathcal{D}'$, where r_i* matches *r'_j. In each pair $(r_i, r'_j) \in \alpha$, we say that r_i is the* preimage *of r'_j and r'_j is the* postimage *of r_i. Two assignments α_p and α_q are* disjoint *if $\alpha_p \cap \alpha_q = \emptyset$.*

In order to achieve k-anonymity, we need to ensure that there exist k disjoint assignments from original records in $\mathcal{D}$ to records in $\mathcal{D}'$. After we have constructed a set of k such desired assignments, we can determine the values of records in $\mathcal{D}'$ therefrom, such that each record $r'_i \in \mathcal{D}'$ is indeed compatible to (i.e., matches) the records mapped to it. Last, we can select one of these k assignments as the one that defines the *true matches* between $\mathcal{D}$ and $\mathcal{D}'$ and publish any other attributes of our data accordingly. This reasoning extends to the case of ℓ-diversity, with the additional provision that the ℓ matches assigned to a record r in ℓ different assignments should have different sensitive labels from each other.

A set of m disjoint assignments defines m distinct matches in $\mathcal{D}'$ for each $r_i \in \mathcal{D}$ (i.e., one by each assignment), and *vice versa*, i.e., m distinct matches in $\mathcal{D}$ for each $r'_i \in \mathcal{D}'$. The net result can be represented by means of a *generalization graph* [24].

DEFINITION 3. *Given a set-valued data set $\mathcal{D} = (R, S)$ and its anonymized version $\mathcal{D}' = (R', S)$, a* generalization graph *$G = (V, E)$ is a directed graph in which each vertex $v \in V$ stands for an original/anonymized record $r_i \in \mathcal{D}$ and $r'_i \in \mathcal{D}'$, and an edge $(v_i, v_j) \in E$ is present iff r_i matches r'_j.*

Our definition corresponds to the *unified* view of such a graph (see Figure 1(a)). In a *bipartite* view, the vertex standing for an original record r_i is separate from that standing for its anonymized form r'_i. A set of m disjoint assignments defines a generalization graph in which each vertex has exactly m outgoing and m incoming edges, i.e., an m-regular generalization graph. As [24] has shown, the reverse is also true, that is, an m-regular generalization graph effectively defines m disjoint assignments.

In our publication model, we publish the anonymized data $\mathcal{D}' = (R', S)$, while for each anonymized record r'_i we also publish a distance bitmap d_i, which denotes with value 1 the bits where r'_i may differ from any of its matches, and a distance threshold t_i, which upper-bounds the number of different bits between r'_i and its matches, hence t_i does not exceed the number of 1 bits in d_i. Taken together, d_i and t_i compactly define a set of *possible worlds* [5], one of which corresponds to the true match of r'_i.

4. METHODOLOGY

Our overall methodology consists of the following five steps.

First, we order the data records by a cyclical order, the **Gray-TSP order**. The utility achieved by our anonymization scheme depends on the extent to which neighboring records in this order are close to each other by some distance metric, hence limit the afflicted information loss. In order to achieve such an outcome, we start out with the order defined by the Gray code of itemset bitmaps, also used by [9], and *enhance* it further via a local approximate solution to the Traveling Salesman Problem (TSP). For the same purpose [24] uses the order defined by a Hilbert curve in the space of attribute value domains; this approach is unsuitable for the high-dimensional space defined by set-valued data.

Second, we create an m-regular generalization graph, where m is k or ℓ. For k-anonymization, we build such a graph as a ring over the Gray-TSP order. On the other hand, in the case of ℓ-diversification, a ring generalization graph would not satisfy the privacy requirements. Therefore, we propose a **Greedy Assignment Extraction** algorithm, which extracts ℓ disjoint assignments from the dataset's complete graph under a constraint derived from the ℓ-diversity requirement, guided by the Gray-TSP order, and forms an ℓ-regular generalization graph from their union; the greedy character of this process aims to achieve low information loss.

Third, we extract m random disjoint assignments from the generalization graph, employing a *random walk*, as in [24] (see Section 2.2). Yet, in contrast to [24], we do not aim to create *cycles* via the

walk, *backtracking* whenever we reach a dead-end. We propose a **Closed Walk** method instead: we allow the followed path to revisit vertices and continue, unobstructed by dead-ends. Thus, we gain a significant efficiency advantage that enables our algorithm to scale and run smoothly over a *full* data set. The random character of this process renders our scheme resistant to adversaries having knowledge of the algorithm [5] or of some anonymized tuples' identities[1] [24]. Besides, this same closed walk is employed for greedy assignment extraction (see above) as well.

Fourth, we pick up one of the extracted k (ℓ) assignments uniformly at random. This assignment defines the putative identity and (when such exists) sensitive label of each anonymized record r'_i. The non-deterministic nature of this step guarantees that each preimage of r'_i has the same probability to be chosen.

Fifth, for each anonymized record, we set its base bitmap r'_i, distance bitmap d_i, and distance threshold t_i, as a function of its m preimages. Let $\mathcal{P}(r'_i)$ be the set of m preimages of r'_i. For the sake of data utility, the values in r'_i should be similar to those of its preimages. To achieve this result, we employ a *bit voting* method: the p^{th} bit of r'_i is set as the most common p^{th} bit value among its preimages (ties are resolved arbitrarily). For example, if $\mathcal{P}(r'_i) = \{1100, 1011, 0101\}$, then r'_i is set to be 1101; while r'_i is not identical to any of its preimages, each one of its bits has the most common value among those in $\mathcal{P}(r'_i)$. Thus, the value of r'_i minimizes the sum of Hamming distances among r'_i and its preimages. We emphasize that there is *no privacy loss* caused by this provision. The match of a record is chosen with equal probability among all the matches in the generalization graph. The bit voting method has no effect on this choice; it only reveals information on what *single items* are frequent in the data, which is the kind of information we wish to give. Next, the value of the p^{th} bit of d_i is set to 0 iff the p^{th} bit is the same among all preimages of r'_i; otherwise it is set to 1, denoting that at least one preimage differs from r'_i in that position. Last, the distance threshold t_i is measured as the maximum Hamming distance among r'_i and its preimages, $t_i = \max\{\mathcal{H}(r'_i, r_j) \forall r_j \in \mathcal{P}(r'_i)\}$. Eventually, d_i and t_i define a set of *possible worlds* that is a superset of $\mathcal{P}(r'_i)$.

We now elaborate on the elements of our approach.

4.1 The Gray-TSP Order

The *Gray code*, or *reflected binary code* [13], is a binary numeral system where two successive values differ in only one bit, i.e. their *Hamming distance* is 1. Table 5 depicts an example of Gray encoding for the decimals from 0 to 7.

Decimal	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111
Gray	000	001	011	010	110	111	101	100

Table 5: An example of Gray coding

An itemset drawing items from a universe $\mathcal{I}$ of b items may take one of 2^b values. A *Gray order* defined over these values, expressed as bitmaps, provides a guide for *sorting* a dataset $\mathcal{D}$ of records drawing items from $\mathcal{I}$. Nevertheless, a typical real-world data set $\mathcal{D}$ contains much fewer records than the 2^b possible records (bitmaps) of size b. In effect, even after the records in $\mathcal{D}$ are sorted by the Gray order of their bitmaps, there will still be large gaps, i.e. large Hamming distances, between consecutive records.

To mitigate this drawback, we use the Gray order only as an initialization step, and then enhance it via a local application of an approximation algorithm for the Traveling Salesman Problem (TSP). In particular, we first sort $\mathcal{D}$ by its Gray order, to obtain a sorted version, $\sigma(\mathcal{D})$. Then we divide $\sigma(\mathcal{D})$ into segments. In each segment S_i, we fix the position of the first and last record, r_f and r_l, and treat each record $r_i \in S_i$ as a node v_i in a complete weighted graph $\mathcal{G}(V, E)$, where each edge $(v_i, v_j) \in E$ is weighted by the Hamming distance among the records corresponding to its incident nodes, $\mathcal{H}(r_i, r_j)$. We aim to locally reorder the internal records in S_i so as to reduce the total sum of Hamming distances among consecutive records. This problem amounts to solving the TSP on $\mathcal{G}$. As the TSP is NP-hard, we apply an efficient genetic algorithm therefor [22], with v_f as origin and v_l as destination.

We divide $\sigma(\mathcal{D})$ into segments so as to avoid applying the TSP algorithm on the full size of the data. We emphasize that our strategy does not aim to acquire the optimal TSP solution, but only to leverage a TSP algorithm in order to improve upon the Gray order. We fix the first and last record in each segment so as to facilitate the transitions among segments, preserving the Hamming distances provided by the Gray order at these breakpoints. Ideally, these breakpoints should be placed at positions where the Hamming distance between consecutive records in the Gray order is small. To achieve this effect, we design a dynamic programming (DP) algorithm that finds appropriate breakpoints. This DP algorithm receives as parameters the minimum and maximum segment size allowed, m and M respectively, and detects the *optimal* way of partitioning $\mathcal{D}$ into segments under these constraints, so that the sum of Hamming distances at breakpoints is minimized. Let $C(i)$ be the minimum sum of Hamming distances for partitioning the first i records in $\sigma(\mathcal{D})$. $C(i)$ is recursively computed as:

$$C(i) = \min_{j \in [i-M, i-m]} \{C(j) + \mathcal{H}(r_j, r_{j+1})\}, \quad C(0) = 0 \quad (1)$$

In Equation 1, the j variable goes through all the allowed positions for the last breakpoint in the examined prefix of $\sigma(\mathcal{D})$, and chooses the best among them. The overall solution is obtained by computing $C(n)$ in $O\big((M-m)n\big) = O(n)$. Eventually, after partitioning $\sigma(\mathcal{D})$ into segments and locally enhancing each of them by TSP, we arrive at a Gray-TSP order of $\mathcal{D}$, denoted as $\phi(R)$.

4.2 The Closed Walk

We now describe our Closed Walk algorithm for assignment extraction. We apply this algorithm in both our k-anonymization and ℓ-diversification algorithms. In the former, we use it only to extract k disjoint assignments from a k-regular ring generalization graph over the Gray-TSP order, using all its $n \times k$ edges and making walk choices in a *random* manner. In the latter, we use it first to extract ℓ assignments from the *complete* directed graph of the data so as to define our ℓ-regular generalization graph of $n \times \ell$ edges, making walk choices in a *greedy* manner and assisted by the Gray-TSP order (see Section 4.4); then, we reapply a *random* closed walk to extract ℓ disjoint assignments from this ℓ-regular graph, so as to render our scheme proof against sophisticated adversaries.

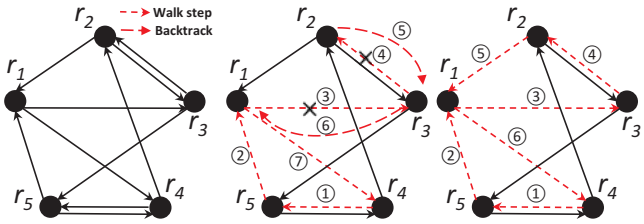
Our algorithm works in m rounds. Each round generates an assignment A_i, disjoint from previously generated ones, by iterative cycle extraction (see Section 2.2); it repetitively starts from a random node, takes a walk to build a cycle along edges that have not been traversed before (neither in previous rounds nor in the current one); when the walk is closed, the graph is updated so as to render all selected edges as self-loops; this process is repeated until all nodes are covered; the final set of selected edges represents the generated assignment; m rounds generate m disjoint assignments.

The algorithm in [24], which we call WMC, obeys the constraint that a node cannot be revisited by the same random walk. Thus, WMC encounters a *dead-end* in situations where there is no avail-

[1]Furthermore, our scheme impedes attacks that calculate the statistical likelihood of possible worlds, such as the deFinetti attack [16]; to launch such an attack, an adversary would have to calculate the likelihood of each possible assignment, and therefor find the number of possible assignments, a #P-complete problem.

able next hop to move to, as it has previously traversed all nodes adjacent to its current position; then it *backtracks* and attempts to correct a previous decision. Such backtracking may occupy most of its running time, manifesting its worst-case $O(kn^2)$ complexity.

In contrast, when our algorithm arrives at a position where all next hops have been visited by the current walk, it *revisits* one of them, say u, anyway; thereby, it creates a *deviant cycle* starting from and ending at u. This deviant cycle is henceforward ignored, and the walk proceeds until it closes by reaching the node it started from. In graph theory terms, while WMC strives to directly build a *cycle*, i.e., a closed walk that does not revisit any vertex, our algorithm takes a plain *closed walk* and simply ignores deviant cycles.



(a) Generalization graph (b) Backtracking (c) Closed-walking

Figure 3: Backtracking vs. Closed-walking

We illustrate the difference between backtracking and closed-walking by extracting a cycle from the 2-regular generalization graph in Figure 3(a). Let the first round start from r_4, and randomly pick its first 4 hops as in Figure 3(b). Then WMC encounters a *dead-end*, as r_1 and r_3, the two adjacent nodes of r_2, are already in the walk; thus, it *backtracks* to r_3 (Step 5). At r_3, there is still no other available next hop, as r_5 has been visited. In effect, WMC backtracks onto r_1 (Step 6), where it eventually selects a legitimate alternative next hop, r_4, and thus completes a cycle (Step 7). Altogether, it takes 7 steps to detect cycle $r_4 \rightarrow r_5 \rightarrow r_1 \rightarrow r_4$.

Figure 3(c) shows how our closed-walk algorithm resolves the same conflict. At Step 5, instead of backtracking, we revisit r_1, thereby creating *deviant cycle* $r_1 \rightarrow r_3 \rightarrow r_2 \rightarrow r_1$. Nodes r_3 and r_2, are duly removed from the cycle. In step 6, we move on to r_4 and close cycle $r_4 \rightarrow r_5 \rightarrow r_1 \rightarrow r_4$, constructed in only 6 steps. The step difference between backtracking and closed-walking can be arbitrarily large; for a deviant cycle of p edges, backtracking may have to perform p extra steps so as to annul the deviation; as we show in Section 5.3, such $O(p)$ differences, accumulated over many deviations, translate to a significant efficiency advantage.

Algorithm 1: Assignment extraction by Closed Walk

Data: The dataset $\phi(R)$ and $\phi(R')$ sorted in Gray-TSP order; The privacy level m; Current round r
Result: An assignment A_r
1 $A_r \leftarrow \{(u_i, u'_i)\}$ where $u_i \in \phi(R)$ and $u'_i \in \phi(R')$;
2 $L \leftarrow R$;
3 **while** $L \neq \emptyset$ **do**
4 $visited \leftarrow$ new empty list;
5 Pick $u_i \in L$ at random;
6 $u'_j \leftarrow$ Pick(u_i, $\phi(R)$, $\phi(R')$, $visited$);
7 add (u_i, u'_j) to $visited$;
8 **while** $u'_j \neq u'_i$ **do**
9 $u_x \leftarrow u$ s.t. (u, u'_j) in A_r;
10 $u'_y \leftarrow$ Pick(u_x, $\phi(R)$, $\phi(R')$, $visited$);
11 add (u_x, u'_y) to $visited$; set the child of u'_j to be u'_y;
12 $u'_j \leftarrow u'_y$;
13 Update A_r with the matchings in the cycle;
14 Remove nodes matched with nodes in the cycle from L;
15 **return** A_r;

Algorithm 1 provides the pseudo-code for assignment extraction by closed walk. It first initializes an assignment A_r (Line 1), in which each u_i is matched with u'_i. This assignment does not need to be valid; some of the matches (edges) in it may have already been used by previous assignments. Our task is to *update* A_r with valid matchings. We set L as the list of unprocessed nodes, initially all nodes in the graph (Line 2). After a cycle is found, its nodes are removed from L, hence $|L|$ is monotonically decreasing. Then we enter a cycle-discovery loop, to be terminated when all nodes have been assigned to a cycle, i.e. when $L = \emptyset$ (Lines 3-14). For each cycle to be created, we initialize a *visited* data structure (Line 4), which keeps track of each visited node and its previously traversed next hops. This structure facilitates two objectives: (i) we prefer visiting previously unvisited nodes, so that we do not create deviant cycles; we only do so when no unvisited next hop node is available; (ii) when we do revisit a node v, we should not reselect a previously traversed next hop, lest we reiterate the same deviant cycle.

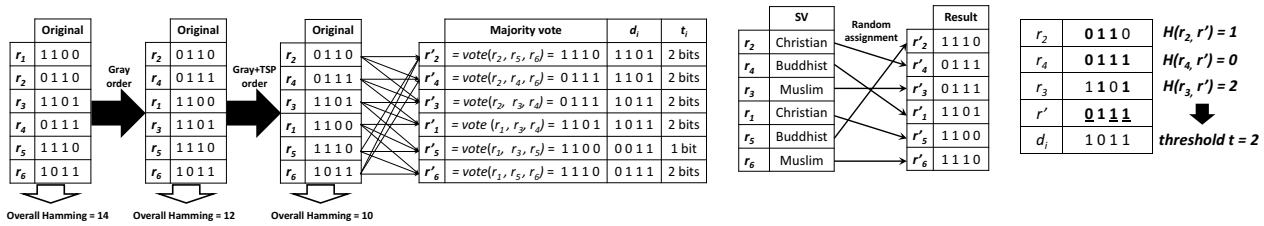
We initiate a cycle by picking up a node $u_i \in L$ at random (Line 5). Then we select a next hop, u'_j (Line 6); our method for picking up u'_j is either *random* or *greedy*; this point makes the difference between a *Random Walk*, used for extracting random assignments from a generalization graph with both our k-anonymization and ℓ-diversification schemes, and a *Greedy Walk*, used for extracting assignments from the complete graph to build an ℓ-regular generalization graph for ℓ-diversification. In both cases, we always choose a next hop not used in a previous assignment; the choice (either random or greedy) is preferably made among next hops not already visited in the current walk either; if such options are not available, then we choose among visited ones, creating a deviant cycle. We elaborate further on the *greedy* choice in the next section.

Once the next hop has been chosen, the pair (u_i, u'_j) is added to the $visited$ data structure (Line 7). Then a loop iterates until the cycle under construction is closed by reaching u'_i (Lines 8-12). At each iteration, we pick (Line 9) the current preimage u_x of the selected next hop u'_j in assignment A_r, choose a new next hop, u'_y, for u_x (Line 10), add the pair (u_x, u'_y) to the *visited* structure, and set u'_y as the child of u'_j, so as to retrieve the created cycle later (Line 11). The matching (u_x, u'_y) is *not* registered in the extracted assignment A_r at this point, as it may yet be updated by later steps of the same walk. We then assign u'_y to u'_j and proceed with the next hop (Line 12). When the internal **while** loop (Lines 8-12) terminates, a cycle has been discovered; then assignment A_r is *updated* with the matchings in that discovered cycle (Line 13). This update may annul some matchings created by a previous cycle. Yet the overall process is progressive, as *at least one new record* from L is added to the set of matched records with each cycle; previously matched records may re-orient their matches (i.e., their preimage and postimage), but they do not become unmatched. Last, the set of newly matched nodes is removed from L, hence cannot be selected as a starting point of a cycle again (Line 14). Once all nodes are removed from L, an assignment has been extracted.

While our algorithm's worst-case complexity is quadratic, it performs less redundant steps than WMC [24], and is therefore more efficient. This algorithm rests on the assumption that the walk can always be closed by returning to the starting node without reusing any edge. The following theorem justifies this assumption.

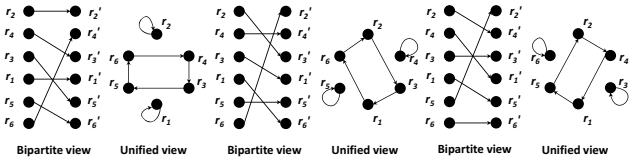
THEOREM 4.1. *In a directed graph G where each node u has the same number m_u of incoming and outgoing edges, if there is a path from node v to v', then there exists a path from v' to v that does not reuse any edge in the path from v to v' .*

PROOF. Consider the graph G' consisting of all nodes and edges in G *except* the edges along the path from v to v', and *with* an additional edge from v to v'. Each node in G' has the same number of incoming and outgoing edges, as we have deleted one incoming and one outgoing edge from each node along the path, and the added



(a) Workflow (b) SV Assignment (c) Bit voting

Figure 4: Workflow and publication details in our methodology



(a) Assignment 1 (b) Assignment 2 (c) Assignment 3

Figure 5: Extracted assignments in our example

edge from v to v' compensates for the edges these nodes have lost. If a path from v' to v exists in G', then it also exists in G, and, by the definition of G', does not reuse any edge in the path from v to v'. Thus, it suffices to prove that such a path exists in G'.

Assume there is no such path. Then consider W, the set of nodes in G' that can be reached from v'; v' is in W and has at least one outgoing edge (given that it has an incoming edge), hence W is non-empty. By definition, each outgoing edge from a node in W leads to a node in W, hence is an incoming edge to W. Since each node in G' has an equal number of incoming and outgoing edges, it follows that each edge incoming to W is also outgoing from W. By our assumption, v does not belong to W, hence the edge from v to v' is incoming to, but not outgoing from W. By reductio ad absurdum, it follows that there is a path from v' to v in G', hence in G, that does not reuse any edge in the path from v to v'. □

4.3 A Coherent Example

Figure 4 carries the example in the introduction forward by illustrating all elements of our methodology. The 3-regular ring generalization graph in Figure 1(a) is already defined on the Gray-TSP order over the dataset. Figure 4(a) shows how this order is created. The six records are first sorted by their Gray order, reducing the sum of their Hamming distances from 14 to 12. The application of the TSP algorithm further reduces this distance to 10. The Gray-TSP order $(r_2, r_4, r_1, r_3, r_5, r_6)$ is then used in the graph of Figure 1(a), whose edges are also shown in Figure 4(a). The values of anonymized records are defined by the *majority vote* of each record's preimages in the graph, as also shown in Figure 4(a). The details of voting are shown for record r'_3 as example, in Figure 4(c). Furthermore, Figure 5 depicts the three disjoint assignments we extract. Eventually, we randomly pick one of these; assume the one in Figure 4(c) is chosen. We use this assignment as a guide to assign presumed identities and any other attributes, such as sensitive labels, to our six records, as in Figure 4(b). The anonymized data we obtain are the same as those in Table 3. However, for the sake of simplicity, in that table we did not yet present the effect of assigning sensitive values according to a randomly selected assignment.

4.4 Greedy Assignment Extraction

The solution in our example applies our k-anonymization algorithm and satisfies 3-anonymity. By chance, it also happens to satisfy 3-diversity, as each original record matches three anonymized records of different sensitive values. However, in order to systematically address the ℓ-diversification problem, we need to ensure that the generalization graph we work with satisfies the ℓ-diversity requirement, i.e., matches each original record to ℓ anonymized postimages of different sensitive values. Such a generalization graph cannot be built by applying a simple rule over a given order. However, we can eschew the a priori construction of a generalization graph altogether. Instead, we start out by assuming a *complete generalization graph*, i.e. a graph where an edge exists from every preimage to every postimage, extract ℓ assignments therefrom, and build the generalization graph we eventually use as the union of these ℓ assignments. Assuming the full data set satisfies ℓ-diversity (is ℓ-eligible [18]), such ℓ assignments can be extracted, so that each record obtains ℓ-diverse matches. The burden falls upon our closed-walk algorithm to take sensitive values in consideration when *picking* next hops. We now outline our method for picking up next hops in a manner that satisfies the ℓ-diversity requirement, i.e., ensures that each postimage a record is matched to a sensitive value different from those it was previously matched to, while otherwise making greedy decisions for the benefit of utility. Algorithm 2 presents a pseudo-code for this greedyPick method, to be used in our closed-walk algorithm (Section 4.2).

Algorithm 2: greedyPick(u, $\phi(R)$, $\phi(R')$, $visited$)

```
1  i ← the rank of u in φ(R);
2  last ← false;
3  for j ← 0 to n/2 + 1 do
4      u'_1 ← record at rank i + j  mod n in φ(R');
5      u'_2 ← record at rank i − j  mod n in φ(R');
6      S_u ← sens. labels of records previously matched with u;
7      for p ← 1 to 2 do
8          if (u, u'_p) ∈ any A_1 ... A_{r−1} or visited then
9              u'_p ← null;
10         if u'_p.s ∈ S_u then
11             u'_p ← null;
12     if u'_1 and u'_2 both are null then
13         continue loop;
14     u' ← u'_p ∈ {u'_1, u'_2} s.t. H(u'_p, u) is minimum;
15     if ∄ u_x, s.t. (u_x, u') ∈ visited then
16         u'' ← u'; break loop;
17     if last = false then
18         u'' ← u'; last ← true;
19 return u'';
```

In a nutshell, given a preimage u and the TSP-Gray-sorted node lists $\phi(R)$ and $\phi(R')$, greedyPick aims to return an eligible postimage for u that is close to u by Hamming distance, while preferring unvisited nodes to visited ones. The rank of u in $\phi(R)$ is denoted as i (Line 1). We use a boolean, initialized as false (Line 2), which indicates whether an option of *last resort* has been reached, so that a deviant cycle may be created by picking up as next hop a node already visited in the current walk. While such an option is not preferred, it can be opted for if no other choice is available. For the sake of utility, we prefer to select a postimage that is close to u in the Gray-TSP order. We progressively search for such a postimage in Lines 3-20. In each iteration we consider the next two candidate records, u'_1 and u'_2, that are one position further away from u (in two directions along the Gray-TSP order) than previously considered ones (Lines 4-5), and try to match ei-

ther u'_1 or u'_2 with u, while satisfying the following criteria: (i) u cannot be matched to a record it has been matched to in a previous assignment; (ii) in case u is being revisited (i.e., a deviant cycle is created), it cannot be matched again to a next-hop record it was already matched to in this walk, lest we reiterate the same deviant cycle indefinitely (see Section 4.2); (iii) for ℓ-diversity to be satisfied, u cannot be matched to a record having the same sensitive label as a match of u in a previous assignment. In case both u'_1 and u'_2 fail these criteria, the loop continues to the next iteration (Lines 6-13). Otherwise, we pick the one that has the lowest Hamming distance to u as u' (Line 14), while also ensuring that a node not previously visited in the current walk is preferred to an already visited one. If u' has *not* been previously visited, the loop terminates and u' is returned as u'' (Lines 15-16). Otherwise, if no option of last resort has been set before, u' is marked as the best such option (Lines 17-18). Thus, u' will be eventually returned, unless a more preferable option is found in a subsequent iteration.

We emphasize the greedy character of the process. As $\phi(R)$ is sorted by the Gray-TSP order, and we always pick up u' as close as possible to u, we expect their Hamming distance to be small; at the same time, we avoid u' with sensitive labels already picked up in previous assignments. To that end we maintain the set S_u of sensitive labels already assigned to u (Line 6). After u' is picked as a match for u, its label is also added to S_u. Eventually, our generalization graph is created as the union of ℓ disjoint assignments extracted by closed walk using our greedyPick method.

Dataset	# records n	Avg. size	Universe size $\|I\|$
Chess	3,196	37	75
Pumsb	49,046	75	7,117

Table 6: Dataset information

5. EXPERIMENTAL EVALUATION

We now evaluate our schemes experimentally. We use two real-life set-valued data: Pumbs and Chess, available at the UCI Machine Learning Repository.[2] Table 6 presents the data specifications. Pumsb contains transactions representing a sample of responses from the Los Angeles – Long Beach area census questionnaire. Such data sets are used in targeted marketing campaigns for identifying a population likely to respond to a particular promotion. Chess contains 37-attribute board-descriptions for chess endgames. The first 36 attributes describe the board, while the last attribute is the classification: "win" or "nowin". For the evaluation of our ℓ-diversification scheme, we have introduced sensitive labels in all data in a consistent manner. We obtain the empirical distribution of sensitive labels from the histogram of occupation values from the census data[3] of 1990, and assign to each record a randomly sampled sensitive label. The extracted census data has 3,030,728 records with 470 distinct occupation attribute values.

We evaluate our schemes in: (i) the information loss incurred by the anonymization process; (ii) the accuracy in answering aggregate queries over the data; and (iii) runtime efficiency and scalability. Our algorithms were implemented in Java and experiments ran on a 4 CPU, 2.4GHz Linux server with 8GB RAM.

5.1 Information Loss

We first assess the information loss caused by our techniques. We evaluate our k-anonymization scheme on the benefit brought about by the TSP-Gray sorting, in terms of reducing information loss. In the case of ℓ-diversification, we compare against CAHD (Correlation-aware Anonymization of High-dimensional Data), the most recommended anonymization scheme proposed in [9].

[2]Online at http://archive.ics.uci.edu/ml/datasets/

[3]Online at http://usa.ipums.org/usa/

CAHD partitions records in groups, assisted by a Gray order, so that the distribution of sensitive labels within groups satisfies a privacy requirement p, equivalent to ℓ-diversity for $p = \frac{1}{\ell}$. Eventually, the data is published by breaking the associations among individual records and their sensitive labels. In order to render CAHD comparable to our scheme, we apply our publication model on the data obtained from CAHD as well, i.e., we generalize the bitmaps of records within a group to their most representative bit values by our bit voting scheme. To measure the error inflicted by this model, we propose an Error Rate (ER) metric, defined as the average ratio of the number of bits flipped in the published base bitmap r'_i of an original record r_i to the number of bits valued 1 in r_i.

We measure ER for the anonymized Pumsb and Chess data. In all our experiments, we set the chunk-size range in Gray-TSP sorting to $[300, 350]$. To evaluate the benefit brought about by our sorting scheme, we prepare each data set in two different orders: one using the Gray order only and another using our Gray-TSP order, and apply ring-based nonreciprocal generalization on each. Figures 6(a,e) show our ER results as a function of the k parameter. Remarkably, lower ER values are achieved with the Gray-TSP order than with the plain Gray order; this result confirms that the TSP enhancement bears fruits in terms of containing information loss. We emphasize that the Gray-only technique is *also* using nonreciprocal recoding. Figures 6(b,f) show our results on ℓ-diversification, comparing our complete nonreciprocal method (NR) to CAHD, as a function of ℓ. The results show a clear utility advantage for NR, which is gained thanks to both nonreciprocal recoding and our TSP-based enhancement of the Gray order.

5.2 Answering Aggregation Queries

Next, we study the accuracy achieved with anonymized data over aggregation queries. We propose two types of queries, which count records based on whether a certain itemset is present in or absent from them. Given $In \subseteq I$ and $Ex \subseteq I$, these types are defined as:

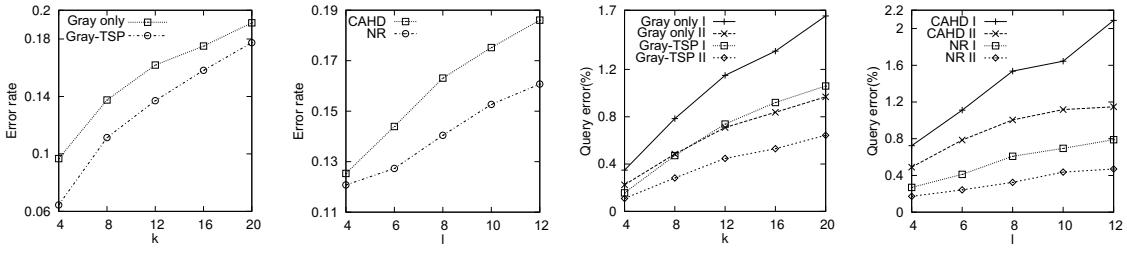
Type I: Select COUNT(r) FROM R' WHERE $In \subseteq IS(r_i)$;

Type II: Select COUNT(r) FROM R' WHERE $Ex \cap IS(r_i) = \emptyset$;

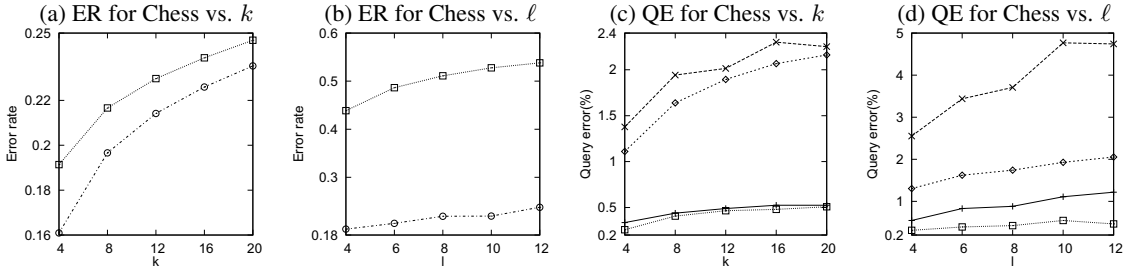
A Type I (II) query counts records with certain items present (absent). We first specify the size of In and Ex based on the average number of records and the universe size in each dataset. In particular, the values of $(|In|, |Ex|)$ for Pumsb and Chess are $(1, 5)$ and $(3, 4)$, respectively. We randomly select $|In|$ items from I to form In, and $|Ex|$ items from I to form Ex. For each tested value of k, we run 500 random queries, and measure the query error (QE), defined as $QE = \frac{|C_o - C_a|}{n}$, where C_o (C_a) is the result obtained from the original (anonymized) data and n the size of the dataset. Figures 6(c,d,g,h) show the average QE results. Again, our TSP-based method permits lower query error than the variant using only a Gray-code order, while our nonreciprocal ℓ-diversification scheme clearly outperforms CAHD.

5.3 Runtime Results

We now evaluate the benefit brought about by our closed-walk algorithm for assignment extraction as compared to the backtracking algorithm in [24]. Figure 7(a) presents the time needed for assignment generation in k-anonymization by both algorithms on the Pumsb data, as a function of k. Our closed walk offers a clear efficiency benefit. We also examine scalability in data size. We obtain data sets of size $2\times$, $4\times$, $8\times$ and $16\times$ that of Chess by duplication and random perturbation. We ran both compared methods on these data, with k set to 16. Figure 7(b) shows our results on *logarithmic* axes. Our closed-walk method maintains an advantage of almost one order of magnitude with increasing data size. We also evaluate

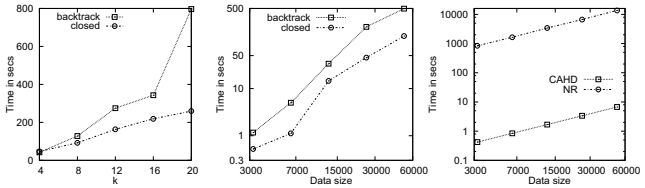


(a) ER for Chess vs. k (b) ER for Chess vs. ℓ (c) QE for Chess vs. k (d) QE for Chess vs. ℓ



(e) ER for Pumsb vs. k (f) ER for Pumsb vs. ℓ (g) QE for Pumsb vs. k (h) QE for Pumsb vs. ℓ

Figure 6: Bit Error Rate and Query Error in k-anonymization and ℓ-diversification



(a) Time with Pumsb (b) k-anonymization (c) ℓ-diversification

Figure 7: Runtime vs. k and size

the scalability of our ℓ-diversification technique vis-à-vis CAHD on the same data, setting ℓ to 6. Figure 7(c) shows our results. For our technique, the measured time includes the time for TSP-Gray sorting, greedy assignment generation for building the generalization graph, and random assignment generation for the sake of privacy (see Section 4). Expectedly, our method requires more time than CAHD, but presents a similarly scalable growth trend. Arguably, the extra time is a reasonable cost for the utility benefits it brings.

6. CONCLUSION

In this paper we revisited the problem of sharing set-valued data while conforming to k-anonymity-like and ℓ-diversity-like privacy guarantees. We proposed a novel *nonreciprocal* anonymization scheme for such data, whereby it is *not* required that original records match anonymized ones in groups. In the process, we brought the state of the art for nonreciprocal anonymization forward in terms of efficiency, applied it on whole data sets, and developed a special method for nonreciprocal ℓ-diversification. Our technique comes along with a novel way to devise a *cyclical order* over set-valued records, employing the Gray-code order and improving on it by applying a TSP algorithm. Our experimental study demonstrates that our schemes preserve data utility to a degree not achieved by previous methods; the extra runtime required compared to CAHD is an affordable price to pay for the benefits we gain. In the future, we plan to investigate how nonreciprocal anonymization techniques can be applied to other types of data, e.g. spatial data.

Acknowledgments

We thank Gabriel Ghiniţa who shared the CAHD code with us, and the anonymous reviewers for their constructive feedback.

7. REFERENCES

[1] C. C. Aggarwal and P. S. Yu. On privacy-preservation of text and sparse binary data with sketches. In *SDM*, 2007.

[2] S. Agrawal, J. R. Haritsa, and B. A. Prakash. FRAPP: A framework for high-accuracy privacy-preserving mining. *Data Min. Knowl. Discov.*, 18(1):101–139, 2009.

[3] J. Brickell and V. Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *KDD*, 2008.

[4] J. Cao, P. Karras, C. Raïssi, and K.-L. Tan. ρ-uncertainty: Inference-proof transaction anonymization. *PVLDB*, 3(1):1033–1044, 2010.

[5] G. Cormode, N. Li, T. Li, and D. Srivastava. Minimizing minimality and maximizing utility: Analyzing method-based attacks on anonymized data. *PVLDB*, 3(1):1045–1056, 2010.

[6] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding association rules by using confidence and support. In *IHW*, 2001.

[7] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, 2003.

[8] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, 2002.

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

[10] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints. *ACM TODS*, 34(2):1–47, 2009.

[11] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *ICDE*, 2008.

[12] A. Gionis, A. Mazza, and T. Tassa. k-anonymization revisited. In *ICDE*, 2008.

[13] F. Gray. Pulse code communication. US Patent 2632058, 1953.

[14] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *PVLDB*, 2(1):934–945, 2009.

[15] Y. Hong, X. He, J. Vaidya, N. R. Adam, and V. Atluri. Effective anonymization of query logs. In *CIKM*, 2009.

[16] D. Kifer. Attacks on privacy and deFinetti's theorem. In *SIGMOD*, 2009.

[17] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization techniques for large-scale datasets. *ACM TODS*, 33(3):17:1–17:47, 2008.

[18] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM TKDD*, 1(1):3, 2007.

[19] S. J. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *VLDB*, 2002.

[20] P. Samarati. Protecting respondents' identities in microdata release. *IEEE TKDE*, 13(6):1010–1027, 2001.

[21] Y. Saygin, V. S. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Rec.*, 30(4):45–54, 2001.

[22] H. Sengoku and I. Yoshihara. A fast TSP solver using GA on JAVA. In *AROB*, 1998.

[23] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *The VLDB Journal*, 20(1):83–106, 2011.

[24] W. K. Wong, N. Mamoulis, and D. W. L. Cheung. Non-homogeneous generalization in privacy preserving data publishing. In *SIGMOD*, 2010.

[25] Y. Xu, K. Wang, A. W.-C. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *KDD*, 2008.