On Optimal Anonymization for l^+-Diversity

Junqiang Liu[#*], Ke Wang[#]

[#]*Simon Fraser University*
Burnaby, British Columbia V5A 1S6, Canada
{jjliu, wangk} @ cs.sfu.ca
[*]*Zhejiang Gongshang University*
Hangzhou, Zhejiang 310018, China

Abstract — **Publishing person specific data while protecting privacy is an important problem. Existing algorithms that enforce the privacy principle called *l*-diversity are heuristic based due to the NP-hardness. Several questions remain open: can we get a significant gain in the data utility from an optimal solution compared to heuristic ones; can we improve the utility by setting a distinct privacy threshold per sensitive value; is it practical to find an optimal solution efficiently for real world datasets. This paper addresses these questions. Specifically, we present a pruning based algorithm for finding an *optimal solution* to an extended form of the *l*-diversity problem. The novelty lies in several strong techniques: a novel structure for enumerating all solutions, methods for estimating cost lower bounds, strategies for dynamically arranging the enumeration order and updating lower bounds. This approach can be instantiated with any reasonable cost metric. Experiments on real world datasets show that our algorithm is efficient and improves the data utility.**

I. Introduction

To protect individual privacy, data must be made anonymous by enforcing a certain privacy principle before publication. In the literature, the *k-anonymity principle* [22] and the *l-diversity principle* [14] are the most popular privacy principles. The former tackles linking attacks by ensuring that each record in the published data is *indistinguishable* from at least k–1 other records. The latter improves the former by preventing homogeneity attacks, i.e., ensuring that there are at least *l well-represented* values for a given sensitive attribute in each indistinguishable group of records. Various algorithms have been proposed to enforce these principles. Unfortunately, both the optimal k-anonymity problem and the optimal l-diversity problem are NP-hard [13][16]. Most algorithms are heuristic based or approximate ones, e.g., Mondrian [13], TDS [7], space mapping [8], space indexing [9], and approximate k-anonymity [16]. Other algorithms, such as Incognito [12], MinGen [23], and *BinarySearch* [21], find an optimal solution with a quite restrictive anonymization model.

So far, k-Optimize [3] is the only algorithm that finds an *optimal* k-anonymization with a flexible anonymization model. To our knowledge, no counterpart for an *optimal* l-diversity solution has been reported. Therefore, several questions remain unanswered regarding the optimal l-diversity problem: Is there a significant gain in the data utility from having an optimal solution compared to heuristic solutions, and from searching a flexible large solution space compared to a restricted solution space? Can we improve the utility by setting a distinct privacy threshold per sensitive value? Can a large portion of the search space be pruned without missing the optimal solution? Is it practical to find an optimal solution for real world datasets? This paper addresses these questions.

We consider an extended form of the classic l-diversity problem [14], called *optimal l^+-diversity problem*: find such an anonymization of a relational table, that ensures no individual can be linked to a sensitive value s_i with a probability higher than θ_i, and that incurs the minimum information loss. This problem extends the most used instantiation of l-diversity in that the latter is a special case of the former by setting $\theta_i = 1 / l$ for every sensitive value s_i. The anonymization is achieved by generalization and suppression.

The main challenge for this problem is the huge search space consisting of all possible ways of generalization and suppression. This problem is much more difficult than the optimal k-anonymity problem in [3]. k-anonymity observes a natural pruning property, called *monotonicity of suppression*: if a indistinguishable group G of records violates k-anonymity (and should be suppressed), then all subgroups derived by partitioning G violate k-anonymity (and should all be suppressed). l^+-diversity does not conform to such a monotonicity, so the amount of suppression is not monotone in a top-down search. Hence, the pruning techniques in [3] are not applicable to our problem.

Our contribution is an efficient algorithm for finding the optimal l^+-anonymization. The novelties are as follows:

- It adopts the full subtree generalization model [10] with various suppression schemes, which results in a much larger search space than the full domain generalization model [22]. It handles suppression as an integral part of anonymization and finds the equilibrium of suppression and generalization, which is different from previous works [12][21] where suppression is handled by an external constraint.
- It allows a different privacy threshold for a different sensitive value based on its sensitivity [5], which helps preserve more data utility, and provides more protection for sensitive values.
- It enumerates all solutions by the depth-first traversal of a novel *cut enumeration tree*. It estimates a *cost lower bound* for the solutions in the subtree rooted at each node, and prunes the subtree if the lower bound exceeds the current best cost.
- It employs an enumeration order that well suits the cost based pruning: finding a solution with a small cost quickly and packing solutions with large costs into subtrees for pruning.

- It is generic in that it works for any reasonable cost metric [3][10] and can easily adapt to a new metric.

This paper presents several key findings not previously known: by considering a large search space, the optimal solution has a significant utility gain compared to heuristic solutions (and solutions with a restricted search space); by setting a different privacy threshold per sensitive value, we can incur less information loss, i.e., preserve more data utility, while providing more privacy protection; it is possible to find the optimal solutions efficiently for real world datasets by employing strong pruning techniques.

The rest of this paper is organized as follows. Section II reviews the related works, Section III defines the optimal l^+-diversity problem, Section IV presents our search strategy, Section V discusses cost lower bounding methods, Section VI proposes dynamic pruning techniques, Section VII instantiates our approach with cost metrics, Section VIII evaluates our approach, Section IX illustrates the choice of the full subtree generalization model, and Section X concludes the paper.

II. Related Work

In the literature, several famous generalization models were proposed. The first is the *full domain generalization* model [22], which is highly restrictive in that all generalized values of an attribute in the anonymized data must be on the same level of the taxonomy tree associated with the attribute's domain. Such a restriction could exclude many natural generalization solutions. The optimal *k*-anonymizations in MinGen [23], *BinarySearch* [21], and Incognito [12] adopt this model.

The second is the *full subtree generalization* model [10], with which the child values that share a common parent value are either all or none generalized, and each generalization step is applied to all records in the dataset. This model allows more natural generalization solutions where generalized values can be at different levels of taxonomies. With the first and the second model, all values in the anonymized data are exclusive of each other, which is called the *domain exclusiveness* property. Such a property is these models' advantage in that existing algorithms can be applied to the anonymized data without modification. *k*-Optimize [3] is the only optimal *k*-anonymization algorithm that keeps the domain exclusiveness with a flexible generalization model (comparable to the full subtree generalization model). But the techniques in [3] are not applicable to our optimal l^+-diversity problem.

The third is the *multi-dimensional generalization* model [8][13], where a generalization step can be locally applied to a multi-dimensional region. This model has a smaller cost than the full subtree generalization model. However, the generalized data does not observe the *domain exclusiveness*. It follows that many existing algorithms can not be used to process the generalized data, and new *customized* algorithms need to be developed (more in Section IX-A). In this paper, we adopt the full subtree generalization model.

As an alternative to generalization, *bucketization* is proposed by [27]. However, with bucketization, the adversary could easily confirm the *participation* of a target individual, which is a privacy threat [8][17]; bucketization does not consider the partition's extent in the QI space, which may cause high information loss [8]. [15] improved bucketization by making it safe with worse-case background knowledge. [8] relaxed the model in [13] by allowing overlapping among multi-dimensional regions, and improved bucketization [27] by considering QI values when grouping records. And [26] identified minimality attacks (more in Section IX-B).

There are also a lot of works on perturbation techniques [1][6][18][20][24][28] and on anonymizing continuously growing datasets [4][9][18][19]. In this paper, we aimed at anonymizing a single dataset and preserving data semantics *at the record level* by generalization and suppression.

III. Problem Definition

A data holder wants to publish a person-specific table T which contains records $\{t_1,..,t_n\}$ and attributes $\{A_1,..,A_m\}$. As in the literature, we assume that each record represents a distinct individual. One attribute is called the *sensitive attribute*, denoted by SA, whose value must be kept secret for any individual. A subset of the attributes is called the *quasi-identifier*, denoted by QI. The QI values of an individual are publicly available. For any $t \in T$, $t[A_i,..,A_j]$ denotes the value sequence of t on attributes $A_i,..,A_j$. t[QI] is called the QI *value sequence* of t, and $t[A_k]$ is called a QI *value* of t for $A_k \in$ QI.

A. Privacy Requirement

The data holder publishes an *anonymized* version T^* of T. An adversary has access to T^*, but not T. The adversary knows a target individual's QI value sequence *qi* and wants to infer his/her SA value by analysing T^*. The set of the records in T whose *anonymized* QI value sequences are matched with *qi* is called the *partition* of *qi*, denoted by $<qi>$. Let $<qi>^*$ denote the set of the records in $<qi>$ whose anonymized records are published in T^*. Then, the adversary can identify $<qi>^*$.

The *confidence* of the adversary in inferring that s_i is the target individual's SA value is bounded by $conf(s_i \mid <qi>^*)$, where $conf(s_i \mid part)$ denotes the percentage of the records that contain s_i in a partition *Part*, and we also call it the *confidence* of s_i in *Part*. Our privacy goal is to limit $conf(s_i \mid <qi>^*)$.

Definition 1 (l^+-diversity): Let $\theta_i \in [0, 1]$ be a privacy threshold for each SA value s_i. We say that T^* satisfies l^+-*diversity* if for every QI value sequence *qi* in T and every SA value s_i, $conf(s_i \mid <qi>^*) \leq \theta_i$. We say that s_i is violated in a partition *part* if $conf(s_i \mid part) > \theta_i$. ∎

The anonymized version T^* is derived by performing generalization and suppression on T.

B. Generalization Model

With the full subtree generalization model, a generalization solution corresponds to one *cut* through every QI attribute's taxonomy tree associated with the attribute's domain, where *exactly one* node on each root-to-leaf path in the taxonomy trees is contained in the cut.

Let *Cut* be such a cut. Then, the generalized data is obtained by replacing every QI value in T with its taxonomical ancestor in *Cut*. For a QI value v, $g(v, Cut)$ denotes its taxonomical ancestor in *Cut*, and for a record $t \in T$, $g(t, Cut)$ denotes t's generalized version by replacing t's QI values with their taxonomical ancestors in *Cut*. The generalized QI value sequence of t is given by $qi = g(t, Cut)[\mathrm{QI}]$. The partition $<qi>$ is the set of records in T that share the same generalized QI value sequence qi, i.e., $<qi> = \{t \mid t \in T \wedge g(t,Cut)[\mathrm{QI}] = qi\}$. We say $<qi>$ is *induced* by *Cut*, and use *Cut.parts* to denote the set of all partitions induced by *Cut*. Partitions in *Cut.parts* are pair-wise disjoint and their union equals to T.

For a taxonomy tree with a root R, the number of possible cuts can be recursively computed by $\#cuts(R) = 1 + \prod_{r \in children(R)} \#cuts(r)$, where $children(R)$ is the set of all child nodes of R. Clearly, the number of cuts grows exponentially with the number of internal nodes on the taxonomy tree.

C. *Suppression Schemes*

If a partition *Part* violates the privacy requirement by Definition 1, we can remove the violation by suppressing some records or SA values from *Part*. Suppression is independently applied to each partition. We consider several suppression schemes.

- The first scheme, denoted by *vioSA*, deletes *all violating SA values* (i.e., values whose confidence exceeds the threshold) and replaces each deleted occurrence with the special *unknown* value. In the extreme case that all SA values but one occur in *Part*, the adversary can infer that the *unknown* value stands for the only missing SA value.
- The second scheme, denoted by *allSA*, deletes *all SA values* from *Part* if any SA value' confidence in *Part* exceeds its threshold. This scheme incurs more distortion than *vioSA*, but the adversary can not infer what the *unknown* values stand for since all SA values in *Part* are suppressed.
- The third scheme, denoted by *vioRec*, deletes a *minimum number of records* for a violating SA value so as to remove the violation in *Part*. If the adversary knows the size of *Part*, he/she may know that some records were deleted.
- The fourth scheme, denoted by *allRec*, deletes *all records* from a violating *Part*. This scheme is very safe but it causes the most distortion.

allSA and *allRec* are coarse-grained suppression schemes that provide more privacy protection, whereas *vioSA* and *vioRec* are fine-grained suppression schemes that provide less protection. We include all schemes so that the data holder can make a choice based on his/her trade-off between privacy and data utility, or choose no suppression at all, denoted by *NoSupp*, in which case only generalization will be performed.

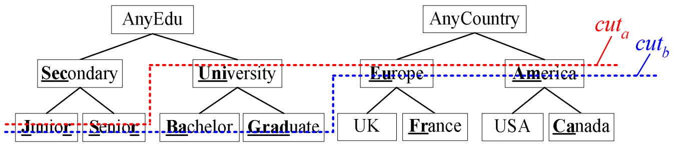
Example 1: Consider T in Fig. 1(a), QI = {Education, Country}, SA = {Disease}, and the QI taxonomy trees in Fig. 1(c)-(d). Cut_a = {Jr, Sr, Uni, Eu, Am} induces 4 partitions <Jr, Eu>, <Sr, Eu>, <Uni, Eu>, and <Uni, Am>. Given a uniform privacy threshold $\theta = 50\%$, partition <Uni, Eu> contains only record 5 and violates the privacy requirement. We derive a more specific Cut_b from Cut_a by specializing Uni to Ba and Grad, which splits the partition <Uni, Am> into sub-partitions <Ba, Am> and <Grad, Am>. <Grad, Am> contains records 8, 9 and 10, where Cancer has a confidence = 2/3 > 50%. With the *vioSA* scheme, Cancer is suppressed from records 9 and 10. With the *allSA* scheme, Cancer and Asthma are suppressed from records 8, 9 and 10. With the *vioRec* scheme, it suffices to suppress either record 9 or 10. With the *allRec* scheme, all 3 records are deleted. ■

#	Education	Country	Disease
1	Junior	France	Flu
2	Junior	UK	Asthma
3	Senior	France	Flu
4	Senior	UK	Cancer
5	Bachelor	UK	Asthma
6	Bachelor	Canada	Asthma
7	Bachelor	USA	Flu
8	Graduate	USA	Asthma
9	Graduate	Canada	Cancer
10	Graduate	Canada	Cancer

(a) Original table *T*

#	Education	Country	Disease
1	Junior	Europe	Flu
2	Junior	Europe	Asthma
3	Senior	Europe	Flu
4	Senior	Europe	Cancer
~~5~~	~~University~~	~~Europe~~	~~Asthma~~ *del*
6	University	America	Asthma
7	University	America	Flu
8	University	America	Asthma
9	University	America	Cancer
10	University	America	Cancer

(b) Anonymized table



(c) Education taxonomy (d) Country taxonomy

Fig. 1 Tables and taxonomies

D. *Optimal Anonymization*

Given *Cut* and a suppression scheme, we can produce the *anonymized* T^* from T in two steps. First, we generalize all QI values in T to their taxonomical ancestors in *Cut*. Then, for every QI value sequence qi in the generalized data, we apply the chosen suppression scheme to $<qi>^*$ to remove violations.

For *Cut* and the chosen suppression scheme, $cost(\bullet, Cut)$ denotes the *anonymization cost*, i.e., the information loss by generalization and suppression, where • can be a table, a partition, a record, or a value. The exact value of $cost(\bullet, Cut)$ depends on the cost metric employed, which we will discuss in Section VII. Notice that costs can be decomposed as

$$cost(T, Cut) = \textstyle\sum_{Part \in Cut.parts} cost(Part, Cut), \text{ and}$$

$$cost(Part, Cut) = \textstyle\sum_{t \in Part} cost(t, Cut).$$

Definition 2 (Optimal l^+-anonymization): Given a table T, taxonomy trees H for QI attributes, and a chosen suppression scheme, find a cut Cut_{best} on H such that T^* defined by Cut_{best} and the chosen suppression scheme satisfies l^+-diversity by Definition 1 and $cost(T, Cut_{best}) = min_{cut}\ cost(T, cut)$ for all possible cuts. The chosen suppression scheme and Cut_{best} comprise an *optimal solution*. ■

To find an optimal solution is hard as the search space of all possible anonymizations is huge. What make our approach practical are the three key elements: the search strategy, cost lower bounding methods, and dynamic pruning techniques, which are presented in the next three sections.

IV. SEARCH STRATEGY

In this section, we focus on how to organize and search all possible anonymizations, and present the generic algorithm.

A. *Cut Enumeration Tree*

Given a chosen suppression scheme, an anonymization is solely represented by a cut through the taxonomy trees of all QI attributes, which is a set of generalized values. We introduce the *cut enumeration tree* to enumerate all cuts. In the cut enumeration tree, each node represents a cut and is denoted by the set of its constituent values. The root represents the most generalized cut consisting of only the root values of the QI taxonomy trees. There is an edge from a parent node *Cut* down to a child node Cut_{child} if Cut_{child} is derived by specializing exactly one constituent value of *Cut*. *subtree*(*Cut*) denotes the subtree rooted at the node *Cut*. The term "cut" and the representing "node" are interchangeable.

To avoid duplicate enumeration of cuts, constituent values of *Cut* are divided into two types. An open value will be further specialized in *subtree*(*Cut*) and a locked value will not. The list of open values is denoted by *Cut.openVs*, and the list of locked values is denoted by *Cut.lockedVs*.

The cut represented by the root of the cut enumeration tree contains only open values. Let *Cut.openVs* = $\{v_1, ..., v_m\}$. Then, the *i*-th child of *Cut*, denoted by Cut_i, is derived by specializing the *i*-th open value v_i to its taxonomical child values, *children*(v_i). For Cut_i, Cut_i*.lockedVs* contains all values in *Cut.lockedVs*, all values in $\{v_1, .., v_{i-1}\}$, and values in *children*(v_i) that are leaf values on the QI taxonomy trees; Cut_i*.openVs* contains all values in $\{v_{i+1}, .., v_m\}$ and values in *children*(v_i) that are non-leaf values on the QI taxonomy trees. Section VI-A will discuss the impact of value ordering on the search space pruning.

Example 2: The cut enumeration tree in Fig. 2 represents all 25 of the taxonomy trees in Fig. 1(c)-(d). All values in a cut are listed in the left-to-right order on the taxonomy trees. Open values are in italic and locked values are underlined. All constituent values of Cut1 are open. Cut2 is derived from Cut1 by specializing AnyEdu to Sec and Uni. In Cut2, Sec and Uni are open since they are non-leaf values on the Education taxonomy tree, and AnyCountry remains open. Cut22 is derived from Cut1 by specializing AnyCountry to Eu and Am. In Cut22, AnyEdu is locked since it is listed before AnyCountry in Cut1*.openVs*. ■

Theorem 1 (Completeness and compactness): *All* cuts are enumerated exactly *once* by the cut enumeration tree. ■

B. *Search Strategy*

A complete traversal of the cut enumeration tree would yield an optimal cut, but it is computationally infeasible because the search space is huge. For example, the entire search space of the Adult dataset in Section VIII consists of 356 million cuts. It is critical to prune any subspace if it contains no optimal cut. We observe the following property of the cut enumeration tree. First, cuts are arranged from general to specific in the top-down direction. Second, cuts are arranged from specific to general in the left-to-right direction because for any cut, its *i*-th subtree specializes one more open value than its (*i*+1)-th subtree, e.g., in Fig. 2, Cut2 and Cut22 are children of Cut1, *subtree*(Cut2) specializes one more open value, AnyEdu, than *subtree*(Cut22).

Therefore, specific cuts are at the lower left portion of the cut enumeration tree. Usually, specific cuts have smaller costs than general cuts. To get a smaller best running cost early, we should search specific cuts as early as possible. The depth-first search suits exactly this strategy.

The depth-first search with pruning. We start from the most generalized cut. Let *Cut* be the cut that we are currently visiting, and Cut_{best} be the best cut examined so far. We first estimate a *lower bound* on the costs of the cuts in *subtree*(*Cut*), denoted by $LB_{cost}(T, Cut)$. The estimation of $LB_{cost}(T, Cut)$ will be discussed in Section V. If the *pruning condition*

$$LB_{cost}(T, Cut) \geq cost(T, Cut_{best})$$

holds, we can prune the entire *subtree*(*Cut*) without missing an optimal cut. If the condition does not hold, we update Cut_{best} if applicable, and then get to the next child Cut_{child} of *Cut*, in the left-to-right order, in a specialization step. After visiting *subtree*(Cut_{child}), we return to *Cut* in a backtracking step. Importantly, in Section VI-A, we will see that a proper order of the values in a cut helps reduce $cost(T, Cut_{best})$ and also helps tighten up $LB_{cost}(T, Cut)$, and hence maximizes the chance to satisfy the pruning condition.

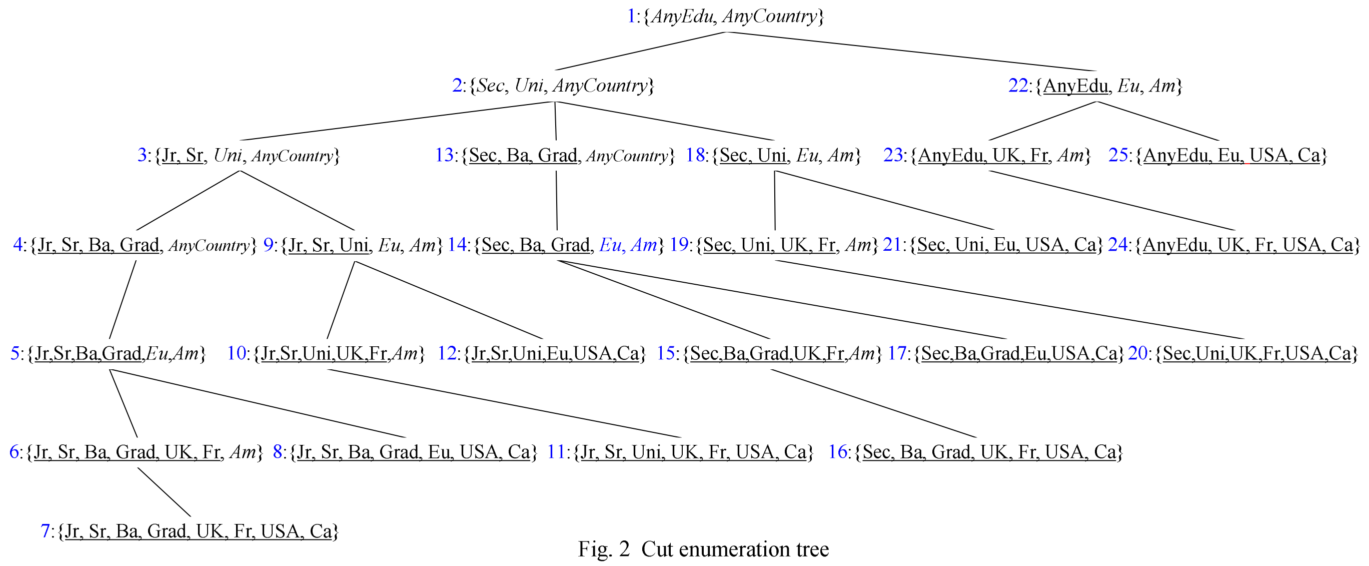

1:{AnyEdu, AnyCountry}
2:{Sec, Uni, AnyCountry}
22:{AnyEdu, Eu, Am}
3:{Jr, Sr, Uni, AnyCountry}
13:{Sec, Ba, Grad, AnyCountry}
18:{Sec, Uni, Eu, Am}
23:{AnyEdu, UK, Fr, Am}
25:{AnyEdu, Eu, USA, Ca}
4:{Jr, Sr, Ba, Grad, AnyCountry}
9:{Jr, Sr, Uni, Eu, Am}
14:{Sec, Ba, Grad, Eu, Am}
19:{Sec, Uni, UK, Fr, Am}
21:{Sec, Uni, Eu, USA, Ca}
24:{AnyEdu, UK, Fr, USA, Ca}
5:{Jr,Sr,Ba,Grad,Eu,Am}
10:{Jr,Sr,Uni,UK,Fr,Am}
12:{Jr,Sr,Uni,Eu,USA,Ca}
15:{Sec,Ba,Grad,UK,Fr,Am}
17:{Sec,Ba,Grad,Eu,USA,Ca}
20:{Sec,Uni,UK,Fr,USA,Ca}
6:{Jr, Sr, Ba, Grad, UK, Fr, Am}
8:{Jr, Sr, Ba, Grad, Eu, USA, Ca}
11:{Jr, Sr, Uni, UK, Fr, USA, Ca}
16:{Sec, Ba, Grad, UK, Fr, USA, Ca}
7:{Jr, Sr, Ba, Grad, UK, Fr, USA, Ca}

Fig. 2 Cut enumeration tree

1) Specialization Step

A specialization step brings us from *Cut* down to the next child Cut_{child} by specializing the next open value *v* in *Cut.openVs* into *children*(*v*). The partitions (induced by *Cut*) containing value *v* are split into sub-partitions (induced by Cut_{child}) containing values in *children*(*v*). And *cost*(*T*, Cut_{child}) can be incrementally calculated based on *cost*(*T*, *Cut*) and the costs of those sub-partitions. $LB_{cost}(T, Cut_{child})$ is computed incrementally, too. More details are in Section V.

2) Backtracking Step

After searching *subtree*(Cut_{child}), the search backtracks to the parent *Cut*. Recall that Cut_{child} was derived by specializing an open value *v* of *Cut*. In the backtracking, we generalize *children*(*v*) to *v* and restore the partitions induced by *Cut*. In addition, the value *v* becomes locked in the rest of *subtree*(*Cut*). This change helps tighten up $LB_{cost}(T, Cut)$. The detail will be given in Section VI-B.

Example 3: Consider Fig. 2. Cut1 induces one partition, <AnyEdu, AnyCountry>, consisting of all records of *T*. The step from Cut1 down to Cut2 specializes AnyEdu to Sec and Uni, and splits <AnyEdu, AnyCountry> into sub-partitions <Sec, AnyCountry> and <Uni, AnyCountry>. After traversing *subtree*(Cut2), the depth-first search backtracks to Cut1 by generalizing Sec and Uni to AnyEdu, and merging the sub-partitions <Sec, AnyCountry> and <Uni, AnyCountry> to restore the partition <AnyEdu, AnyCountry>. In the rest of *subtree*(Cut1), AnyEdu becomes locked. ■

C. Search Algorithm

Our search algorithm is given in Fig. 3. It searches *subtree*(*Cut*) and updates Cut_{best}. *Cut* and Cut_{best} are initialized to the topmost cut, *Cut.openVs* contains all root values on the QI taxonomy trees, and *Cut.lockedVs* is empty. Without confusion, we abbreviate *Cost*(*T*, *Cut*) to *Cost*(*Cut*), and abbreviate $LB_{cost}(T, Cut)$ to $LB_{cost}(Cut)$ in the algorithm. As the algorithm is self-contained, we omit further explanation.

```
$l^+$-Optimize($Cut$, $Cut_{best}$)
1. for each value $v$ in $Cut.openVs$ do
2.     if $LB_{cost}(Cut) \geq cost(Cut_{best})$ then goto 8
3.     $Cut_{child} \leftarrow$ Specialization($Cut$, $v$)
4.     if $LB_{cost}(Cut_{child}) \geq cost(Cut_{best})$ then Backtracking($Cut_{child}$)
5.     else
6.         if $cost(Cut_{child}) < cost(Cut_{best})$ then $Cut_{best} \leftarrow Cut_{child}$
7.         $Cut_{best} \leftarrow l^+$-Optimize($Cut_{child}$, $Cut_{best}$)
8. Backtracking($Cut$)
9. return $Cut_{best}$
```

Fig. 3 Algorithm l^+-Optimize

One key to the efficiency of l^+-Optimize is to estimate a tight lower bound $LB_{cost}(T, Cut)$. This is the topic in the next two sections.

V. Cost Lower Bounding

In this section, we first present a generic lower bound, and then tighten it up for individual suppression schemes.

A. Introduction to Cost Lower Bounding

A lower bound on costs of *subtree*(*Cut*), $LB_{cost}(\bullet, Cut)$, must be no more than the actual cost, *cost*(•,*U*), for *every* cut *U* in *subtree*(*Cut*), where • is a set of records in *T*. Let *Cut.parts* be the set of partitions induced by *Cut*. Then,

$$LB_{cost}(T, Cut) = \sum_{Part \in Cut.parts} LB_{cost}(Part, Cut).$$

In other words, we determine a cost lower bound for each partition *individually* and add up all partitions' lower bounds. To estimate the lower bound, we exploit the following property satisfied by most reasonable cost metrics.

Observation 1 (Metric monotonicity): We say that a cost metric satisfies the *metric monotonicity* if (a) the generalization cost decreases when specializing constituent values of a cut; (b) the cost on generalizing a record is no more than the cost on suppressing the record. ■

From now on, we estimate $LB_{cost}(Part, Cut)$ with a cost metric satisfying the metric monotonicity. A simple case is that, for every open value *v* in *Cut*, all taxonomical leaf descendants of *v* occurring in *Part* are the same. In this case, we say that *Part* is *homogeneous*. Essentially, a homogeneous *Part* never splits if specialized to any cut in *subtree*(*Cut*), therefore, the suppression cost for *Part* is the same at every cut in *subtree*(*Cut*). So we can estimate $LB_{cost}(Part, Cut)$ by *cost*(*Part*, Cut_{spec}), where Cut_{spec} denotes the *most specific cut* in *subtree*(*Cut*) and is derived by specializing all open values in *Cut* to the leaf level on the QI taxonomy trees. More explanation is given by Example 5 in Section VI-B.

Now we consider the case that *Part* induced by *Cut* is not homogeneous. Then, *Part* may have a different suppression cost at each cut in *subtree*(*Cut*). So, it does not work to just consider Cut_{spec}. To estimate $LB_{cost}(Part, Cut)$, we estimate the lower bound for each record in *Part* and sum up the lower bound over all records in *Part*.

Consider a single record *t* in *Part*. Table I summarizes the components of *cost*(*t*, *U*) for a cut *U* in *subtree*(*Cut*). *cost_s*(*t*,*U*) denotes the suppression cost related to *t*. For the *vioSA* or *allSA* scheme, *cost_s*(*t*, *U*) is the cost for suppressing *t*'s SA value, and for the *vioRec* or *allRec* scheme, *cost_s*(*t*, *U*) is the cost for suppressing the record *t*. *cost_g*(*t*, *U*) denotes the cost for generalizing *t* to the cut *U*. With the *vioSA* or *allSA* scheme, *t* may be generalized while *t*'s SA value is suppressed, and with the *vioRec* or *allRec* scheme, *t* is generalized only if *t* is not suppressed. We estimate the lower bound on generalization cost and suppression cost separately.

Table I Possible Components of *cost*(*t*, *U*)

Suppression scheme	Condition	Components
vioSA or allSA	SA value of *t* not suppressed	*cost_g*
	SA value of *t* is suppressed	*cost_g + cost_s*
vioRec or allRec	Record *t* not suppressed	*cost_g*
	Record *t* is suppressed	*cost_s*

B. Lower Bound on Generalization Cost

The metric monotonicity (b) implies that generalizing a record has no more cost than suppressing the record. So, we can estimate $LB_{cost}(Part, Cut)$ by treating each suppressed record in *Part* as if it were generalized, and examining the most specific cut in *subtree*(*Cut*), i.e., Cut_{spec}. The next lemma formalizes this idea.

Lemma 2 (LB on generalization cost): For *Part* induced by *Cut*, $cost\_g(Part, Cut_{spec})$ is an estimate of $LB_{cost}(Part, Cut)$.

Proof: For any cut *U* in *subtree*(*Cut*) and any record $t \in Part$, as summarized by Table I, *cost_g*(*t*, *U*) is always a component of *cost*(*t*, *U*) except when *t* is suppressed with the *vioRec* or *allRec* scheme. In this exceptional case, *cost_s* is the only component, and by the metric monotonicity (b), *cost_s*(*t*, *U*) ≥ *cost_g*(*t*, *U*). For all cases, *cost*(*t*, *U*) ≥ *cost_g*(*t*, *U*), so *cost*(*Part*,*U*) ≥ *cost_g*(*Part*,*U*). By the metric monotonicity (a), *cost_g*(*Part*, *U*) ≥ $cost\_g(Part, Cut_{spec})$. The lemma holds. ■

C. Lower Bound on Suppression Cost

Similarly, let us estimate $LB_{cost}(Part, Cut)$ by only counting the suppression cost. For this goal, given the unit cost for suppressing an occurrence of a SA value s_i, it suffices to estimate the minimum number of occurrences of s_i that must be suppressed by any cut in *subtree*(*Cut*), in order to satisfy l^+-diversity. The following property [14][25] enables us to do so.

Observation 2 (Confidence monotonicity): If the confidence of a SA value s_i in *Part* (as termed in Section III-A) exceeds the threshold θ_i, then s_i's confidence in some sub-partitions split from *Part* must exceed θ_i. ■

Lemma 3 (Minimum number of suppression): For a partition *Part* of size *n*, induced by *Cut*, if a SA value s_i with n_i occurrences in *Part* violates the privacy threshold θ_i, i.e., $n_i / n > \theta_i$, then to satisfy θ_i, at least

$$\#MinRm(Part, s_i) = \lceil (n_i - \theta_i \cdot n) / (1 - \theta_i) \rceil$$

occurrences of s_i in *Part* must be suppressed in *subtree*(*Cut*).

Proof: Consider any cut *U* in *subtree*(*Cut*) and *Part*'s sub-partitions induced by *U*. Let $\#rm(s_i)$ be the total number of occurrences of s_i that must be suppressed from these sub-partitions in order to satisfy the privacy threshold θ_i.

First, let us consider the *vioSA* and *allSA* schemes. Let $SPs^{(1)}$ denote the union of the sub-partitions that violate the threshold θ_i, and $SPs^{(2)}$ denote $Part - SPs^{(1)}$. Then, $SPs^{(2)}$ holds $n_i - \#rm(s_i)$ occurrences of s_i. Since every sub-partition in $SPs^{(2)}$ satisfies the threshold θ_i, so does their union by Observation 2, i.e., $(n_i - \#rm(s_i)) / |SPs^{(2)}| \le \theta_i$. Note that $|SPs^{(2)}| = n - |SPs^{(1)}|$ and $|SPs^{(1)}| \ge \#rm(s_i)$. So $|SPs^{(2)}| \le n - \#rm(s_i)$ and $(n_i - \#rm(s_i)) / (n - \#rm(s_i)) \le \theta_i$. Solving this equation, we get $\#rm(s_i) \ge \lceil (n_i - \theta_i \cdot n) / (1 - \theta_i) \rceil$.

Now, let us consider the *vioRec* and *allRec* schemes. The occurrences of s_i after the suppression is $n_i - \#rm(s_i)$, and the total size of all sub-partitions after suppressing records holding s_i is $n - \#rm(s_i)$ for the *vioRec* scheme and at most $n - \#rm(s_i)$ for the *allRec* scheme. Since after suppression, all remaining sub-partitions satisfy θ_i, $(n_i - \#rm(s_i)) / (n - \#rm(s_i)) \le \theta_i$. Therefore, we get $\#rm(s_i) \ge \lceil (n_i - \theta_i \cdot n) / (1 - \theta_i) \rceil$. ■

With the *vioRec* or *allRec* scheme, suppressing a record holding s_i increases confidences of other SA values. Lemma 3 should be recursively applied until there is no violation.

Example 4: Consider again *T* in Fig. 1(a). Partition <Uni, AnyCountry> consists of records 5 to 10, with Asthma occurring 3 times, Cancer twice, and Flu once. Given a uniform threshold $\theta = 1 / 3$, Asthma violates the threshold. By Lemma 3, *#MinRm* for Asthma is $\lceil (3 - 6 \cdot 1 / 3) / (1 - 1 / 3)) \rceil = 2$. If the *vioSA* or *allSA* scheme is used, the estimation is done. If the *vioRec* or *allRec* scheme is used, suppressing 2 records holding Asthma causes Cancer to violate the threshold. So, we recursively apply Lemma 3 for Cancer, and get its $\#MinRm = \lceil (2 - 4 \cdot 1/3) / (1 - 1/3)) \rceil = 1$. ■

D. Integrated Lower Bounds

Putting Lemma 2 and 3 together yields the first lower bound integrating both generalization and suppression costs.

Theorem 4 (Generic LB): Let $\#MinRm(Part, s_i)$ be determined by Lemma 3, and c_i be the unit cost for suppressing one occurrence of s_i. Then,

$$max \{ \textstyle\sum_i \#MinRm(Part, s_i) \cdot c_i,\ cost\_g(Part, Cut_{spec}) \}$$

is an estimate of $LB_{cost}(Part, Cut)$. ■

This lower bound applies to all suppression schemes. Below, we derive a tighter lower bound for each suppression scheme. With the *vioSA* or *allSA* scheme, *cost_g*(*t*, *Cut*) is *always* a component of *cost*(*t*, *Cut*), and if the SA value in *t* is suppressed, *cost_s*(*t*, *Cut*) is an *additional* component. So, we can add up these two components to get a tighter lower bound.

Theorem 5 (LB for *vioSA* and *allSA*): Let $\#MinRm(Part, s_i)$ be determined by Lemma 3, and c_i be the unit cost for suppressing one occurrence of s_i. If the *vioSA* or *allSA* scheme is used, then

$$\textstyle\sum_i \#MinRm(Part, s_i) \cdot c_i + cost\_g(Part, Cut_{spec})$$

is an estimate of $LB_{cost}(Part, Cut)$. ■

With the *vioRec* or *allRec* scheme, for any record *t*, *cost*(*t*, *Cut*) is equal to *cost_g*(*t*, *Cut*) if *t* is not suppressed, or equal to *cost_s*(*t*, *Cut*) if *t* is suppressed. In other words, *t* has *exactly one* cost component, not both. Theorem 5 does not apply to this case. Theorem 4 applies, but gets a lower bound that is not tight enough for some cost metrics.

To derive a tighter lower bound for the *vioRec* or *allRec* scheme, we estimate the lower bound for the suppressed records and the remaining records *separately* and *add them up*. For any cut *U* in *subtree*(*Cut*), at least $d = \sum_i \#MinRm(Part, s_i)$ records in *Part* will be suppressed by Lemma 3. Let $n = |Part|$, for the remaining $n - d$ records in *Part*, their *cost* is no less than their *cost_g*, which reaches the minimum at Cut_{spec} by the metric monotonicity (a). As each record may have a different *cost_g* at Cut_{spec}, we consider the set of $n - d$ records in *Part* that have smallest *cost_g* at Cut_{spec}. This set is defined by

$$SP_{spec} = arg\ min_{SP \subset Part\ \wedge\ |SP| = n-d}\ cost\_g(SP, Cut_{spec}).$$

The *cost_g* of SP_{spec} gives a lower bound on *cost_g* for the remaining $n - d$ records in *Part*.

Theorem 6 (LB for *vioRec* and *allRec*): Let $n = |Part|$, $d = \sum_i \#MinRm(Part, s_i)$ be determined by Lemma 3, and c_i be the unit cost for suppressing one record holding s_i. If the *vioRec* or *allRec* scheme is used, then

$$\textstyle\sum_i \#MinRm(Part, s_i) \cdot c_i + cost\_g(SP_{spec}, Cut_{spec})$$

is an estimate of $LB_{cost}(Part, Cut)$. ■

In general, to determine SP_{spec} requires examining every size $n - d$ subset of *Part*, which is computationally expensive. However, if the following property holds, we have an efficient computation.

Observation 3 (Independence property): We say that the *independence property* is observed if suppressing a record from *Part* induced by *Cut* does not affect *cost_g*(t, *Cut*) for any other record t in *Part*. ■

The independence property is always observed with the *allRec* scheme because suppressing any record t entails suppressing all records in t's partition. More discussion is in Section VII. With the independence property, *cost_g*(SP_{spec}, Cut_{spec}) equals to the sum of the smallest $n - d$ terms in *cost_g*(*Part*, Cut_{spec}) for $n - d$ records.

Corollary 7 (Simplifying LB for *vioRec* and *allRec*): If the independence property holds, *cost_g*(SP_{spec}, Cut_{spec}) in Theorem 6 can be replaced by the sum of the smallest $n - d$ terms in *cost_g*(*Part*, Cut_{spec}) for $n - d$ records, where $n = |Part|$ and $d = \sum_i \#MinRm(Part, s_i)$. ■

VI. Dynamic Techniques

The previous section presented methods for estimating a cost lower bound with a *given* order for enumerating cuts. The pruning strength by lower bounding critically depends on such an order for enumerating cuts and the way for updating the lower bound. In this section, we discuss these issues.

A. Ordering Values in Cuts

Recall that *subtree*(*Cut*) is pruned if $LB_{cost}(T, Cut) \geq cost(T, Cut_{best})$. We can maximize this pruning in two ways: (1) having a small $cost(T, Cut_{best})$, which can be achieved by examining cuts with small costs as early as possible, and (2) having a large $LB_{cost}(T, Cut)$, which can be achieved by packing cuts with large costs into *subtree*(*Cut*). Notice that the order of values in *Cut.openVs* determines the order of examining cuts in *subtree*(*Cut*), and hence has a great impact on pruning the search space.

First, let us consider (1). Roughly speaking, the more specific a cut is, the smaller its cost is. Thus, to reduce $cost(T, Cut_{best})$ we should first specialize general values which usually have high occurrences. So, we should order the values of *Cut.openVs* in the descending order of their occurrences. For example in Fig. 4, value v_1 is arranged before value v_2 in Cut1. So, Cut2 = {children(v_1), v_2, …} is examined before Cut3 = {v_1, children(v_2), …}. Let $O(v)$ denote the occurrences of a value v. The cost of Cut2 and Cut3 is proportional to $O(v_2)$ and $O(v_1)$ respectively. If $O(v_1) > O(v_2)$, examining Cut2 first has a better chance to reduce $cost(T, Cut_{best})$.

Now we consider (2). If a child cut is derived by specializing a value v in *Cut.openVs*, v becomes locked when backtracking to *Cut* from the child cut. Subsequently, in the rest of *subtree*(*Cut*), v remains locked and hence contributes to $LB_{cost}(T, Cut)$ a cost component proportional to $O(v)$. Therefore, if we arrange the values v in *Cut.openVs* in the descending order of O(v), v with a larger O(v) will become locked earlier, thus contribute a larger cost component to $LB_{cost}(T, Cut)$ earlier. For example, at Cut3 in Fig. 4, value v_1 changes from open to locked. With all other things being equal, $LB_{cost}(T, \text{Cut3})$ is proportional to $O(v_1)$. If $O(v_1) > O(v_2)$, $LB_{cost}(T, \text{Cut3})$ is likely tighter than if $O(v_1) < O(v_2)$. We summarize the above discussion as follows.

Observation 4 (Ordering values in a cut): Arranging the values of *Cut.openVs* in the descending order of occurrences helps produce a small $cost(T, Cut_{best})$ and a large $LB_{cost}(T, Cut)$, thus improving the cost lower bounding based pruning. ■

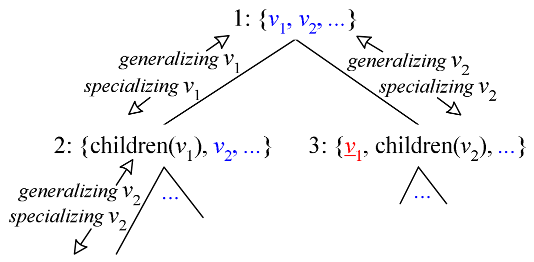


Fig. 4 Value ordering vs. pruning, and partition splitting / merging

B. Updating Lower Bounds

Upon backtracking to *Cut*, one more constituent value of *Cut* becomes locked, making the most specific cut Cut_{spec} in the rest of *subtree*(*Cut*) more general. Also, having more locked values will increase the chance for partitions to be *homogenous* (Section V-A). We can use this chance to tighten up the lower bound for the rest of *subtree*(*Cut*).

Example 5: Consider Fig. 2 again. When first visiting Cut18 = {Sec, Uni, Eu, Am}, Eu is open and the most specific cut in *subtree*(Cut18) is Cut20. The partition <Sec, Eu> consists of records 1 to 4 and is not homogenous. When backtracking from Cut19 to Cut18, Eu becomes locked, so the most specific cut in the rest of *subtree*(Cut18) is Cut21, more general than Cut20. The partition <Sec, Eu> is now *homogeneous* and its lower bound is its actual cost. The rest of *subtree*(Cut18) may be pruned without visiting Cut21. ■

Observation 5 (Updating the lower bound when backtracking): Starting from *Cut*, we get to a child cut by specializing an open value v of *Cut*. When backtracking to *Cut*, v becomes locked in the rest of *subtree*(*Cut*). The lower bound for each partition containing v induced by *Cut* can be tightened up as v is locked. ■

In general, *Cut* has multiple children, $Cut_1, \ldots, Cut_k$. Each time backtracking to *Cut* from a child Cut_i, $LB_{cost}(T, Cut)$ is updated as discussed above. In these updates, $LB_{cost}(T, Cut)$ is monotonically increasing, and $cost(T, Cut_{best})$ is monotonically decreasing. So, each update improves the chance to satisfy the pruning condition $LB_{cost}(T, Cut) \geq cost(T, Cut_{best})$. This idea is incorporated in the backtracking step of I^+-Optimize where $LB_{cost}(T, Cut)$ are incrementally updated by recalculating the lower bound of each partition containing value v, and the pruning condition is always rechecked before specializing to a new child. Experiments show that with all other techniques combined, 99% of the search space is pruned, and with this technique in addition, 99.97% to 99.99% is pruned. This additional 0.97% to 0.99% pruning rate is critical for pruning a huge search space.

C. Reducing Data Scan

The cut enumeration tree is just a conceptual vehicle for describing the search space and strategy; it is never materialized in its entirety in the memory. In the depth-first search, we only materialize the partitions induced by the current cut. So, each specialization step is accompanied by splitting partitions, and each backtracking step is accompanied by merging partitions. Consider the example in Fig. 4 again, where value v_1 involves 1 specialization / generalization step, while value v_2 involves 2 specialization / generalization steps. Thus the overhead of splitting / merging partitions is proportional to $1 \cdot O(v_1) + 2 \cdot O(v_2)$, where $O(v)$ denotes the occurrences of value v. Clearly, $1 \cdot O(v_1) + 2 \cdot O(v_2)$ is smaller if $O(v_1) > O(v_2)$ than if $O(v_1) < O(v_2)$. Therefore, with the values in *Cut.openVs* being arranged in the descending order of occurrences, the overhead of splitting / merging partitions is reduced.

Observation 6 (Reducing data scan): Examining the constituent values of a cut in the descending order of occurrences reduces partition splitting / merging overhead. ■

VII. Instantiation with Cost Metrics

The lower bounds in Theorem 4 to 6 require the metric monotonicity (Observation 1), and the tighter lower bound in Corollary 7 requires the independence property (Observation 3). In this section, we instantiate these lower bounds with several cost metrics by establishing these properties.

A. Loss Metric, LM

The *loss metric* LM [10] quantifies information loss when a leaf value is generalized. If a leaf value v is generalized to a value v^* in the QI taxonomy tree rooted at R, $cost\_g(v, Cut) = (\#leaves(v^*) - 1) / (\#leaves(R) - 1)$, where $\#leaves(v^*)$ and $\#leaves(R)$ denote the numbers of leaves in $subtree(v^*)$ and *subtree*(R). And $cost\_g(t, Cut)$ is the sum of $cost\_g(t[A], Cut)$ for all QI attributes A. For concreteness, we set such a unit cost c_i for suppression that each SA value s_i is as important as all QI values in the record t holding s_i. Therefore, with the *vioSA* or *allSA* scheme, $cost\_s(t, Cut) = c_i = |\text{QI}|$; with the *vioRec* or *allRec* scheme, $cost\_s(t, Cut) = c_i = 2|\text{QI}|$.

LM observes the metric monotonicity (a) because *#leaves* increases when a value is generalized, and observes the metric monotonicity (b) because $cost\_g(t, Cut) \leq |\text{QI}| \leq cost\_s(t, Cut)$. LM observes the independence property since *cost_g* of t is independent of other records.

B. Discernibility Metric, DM

The *discernibility metric* DM [3] attempts to capture the ability to maintain *discernibility* between records. Since DM considers only suppression of records, we only use DM with the *vioRec* and *allRec* schemes. DM assigns a cost to a record t based on the number of records indistinguishable from t on QI. If t is not suppressed, $cost\_g(t, Cut) = |Part^*|$, where $Part^*$ is the anonymized partition holding t. If t is suppressed, DM assigns a cost equal to the size of T. So $cost\_s(t, Cut) = c_i = |T|$.

DM observes the metric monotonicity (a) because the partition size, $|Part^*|$, increase when generalizing values, so does $cost\_g(t, Cut)$, and observes the metric monotonicity (b) because $cost\_g(t, Cut) = |Part^*| \leq |T| = cost\_s(t, Cut)$. DM with the *vioRec* scheme does not observe the independence property since suppressing a record from a partition reduces the size of the partition, which affects the generalization cost of the remaining records in the partition. However, as discussed in Section V-D, the independence property always holds for the *allRec* scheme regardless of the cost metric. So Corollary 7 is applicable to DM if the *allRec* scheme is used.

C. Classification Metric, CM

The *classification metric* CM [10] measures the information loss for classification. For this metric, T contains an additional *class attribute*. Like DM, CM is designed for suppressing records, so we consider only the *vioRec* and *allRec* schemes for CM. Let $minority(Part^*)$ be the set of records in $Part^*$ belonging to minority classes. Suppose that t holds a SA value s_i, if t is suppressed, $cost\_s(t, Cut) = c_i = 1$, otherwise if $t \in minority(Part^*)$, $cost\_g(t, Cut) = 1$, else $cost\_g(t, Cut) = 0$.

CM satisfies the metric monotonicity (a) because $|minority(P_1 \cup P_2)| \geq |minority(P_1)| + |minority(P_2)|$ (proved by [3]), which implies that the generalization cost never increases by splitting a partition. CM satisfies the metric monotonicity (b) because $cost\_g(t, Cut) \leq 1 = cost\_s(t, Cut)$. CM with the *vioRec* scheme does not observe the independence property because suppressing a record from a majority class could cause the minority class to become a majority class. So, Corollary 7 is applicable to CM if the *allRec* scheme is used.

VIII. Experimental Evaluation

Our goal is to study the utility gain of optimal solutions and the feasibility of finding optimal solutions for a very large search space. We used the widely adopted benchmark Adult dataset [2]. This dataset consists of 45,222 records after removing records with missing values. Table II describes the QI attributes (the first 7), the sensitive attribute, and the class attribute. The taxonomy tree for each QI attribute is from [10]. The full search space consists of 356 million (356,440,500) cuts, which makes a complete search infeasible. We further amplified the size of the search space and the size of the dataset to assess the efficiency and scalability of our algorithm. The dataset, the QI taxonomy trees, and the l^+-Optimize executable can be downloaded from the first author's personal website.

TABLE II Description of the Adults Dataset

	Attribute	**|Domain|**	**Generalization Type**	**#Level**
1	Age	74	Ranges-5,10,20,40	5
2	Education	16	Taxonomy tree	5
3	Native Country	41	Taxonomy tree	4
4	Work Class	7	Taxonomy tree	3
5	Marital Status	7	Taxonomy tree	3
6	Gender	2	Taxonomy tree	2
7	Race	5	Taxonomy tree	3
8	Occupation	14	Sensitive attribute	
9	Income Class	2	Class column	

We implemented three algorithms in C++, including l^+-Optimize proposed in this paper, the l-diverse variant [14] of Incognito [12], and the simulated annealing algorithm [11]. As discussed in Section II and IX-A, multi-dimensional generalization models [8][13] do not ensure the domain exclusiveness of the anonymized data, so we did not compare with them since it is less meaningful. All experiments were run on an HP tablet PC with a 2GHz Intel Pentium M CPU and 1GB RAM running Windows XP.

We used the three cost metrics LM, DM, and CM. As explained in Section VII, the suppression schemes *NoSupp*, *allRec*, and *vioRec* are used with all cost metrics, and the suppression schemes *allSA* and *vioSA* are used with LM. For all sets of experiments but one, we used a uniform threshold θ for all SA values. In Subsection A-2) , we explored the utility gain by allowing a different θ_i for a different SA value s_i.

A. Utility Evaluation by Comparing with Incognito

Incognito finds the optimal solution under the restrictive full domain generalization model. Incognito does not incorporate suppression as an integral part rather as an external constraint because the amount of suppression is not monotone and no cost based pruning is employed. So, we only consider Incognito without suppression. We use *LO* as the abbreviation of l^+-Optimize. So, *LO-NoSupp* denotes l^+-Optimize without suppression, *LO-all Rec* denotes l^+-Optimize with the *allRec* suppression scheme, and so on.

1) Information Loss with a Uniform Threshold for SA Values

Fig. 5(a)-(c) show the cost of the anonymized data produced by Incognito and l^+-Optimize with different cost metrics. A uniform l ($1 / \theta$) is used for all SA values, which ranges from 2 to 6 (θ = 50%, 33.33%, 25%, 20%, 16.67%).

The cost of Incognito is always equal to or greater than the cost of l^+-Optimize. With LM as in Fig. 5(a), the gap between Incognito and l^+-Optimize and without suppression (i.e., *NoSupp*) is not significant. The gap increases when l^+-Optimize incorporates suppression. With DM in Fig. 5(b), the gap is quite significant even without suppression: for most cases, the cost of Incognito is 44% to 176% more than the cost of l^+-Optimize. The gap is further increased when l^+-Optimize integrates various suppression schemes. Fig. 5(c) shows that l^+-Optimize also outperforms Incognito with CM. In summary, integrating suppression with generalization helps reduce information loss; the finer the granularity of suppression, the more the reduction, where the order of granularity from coarse to fine is *NoSupp*, *allRec*, *allSA*, *vioSA*, and *vioRec*.

We also studied the usefulness of the anonymized data for classification on Income Class. We anonymized the data by running algorithms with CM, constructed the C4.5 classifier from the training set (the first 30,162 anonymized records), and reported the classification error on the test set (the last 15,060 anonymized records). The baseline of the classification error is 17.1% which is the error with the original data. In Fig. 5 (d), the classification error by Incognito is similar to l^+-Optimize with the *NoSupp* scheme: for a small $l \leq 4$, the classification error is pretty close to the baseline. When l becomes larger, the error is high. However, the error for l^+-Optimize with *vioRec* is always very close to the baseline (with a gap $\leq$ 1%).

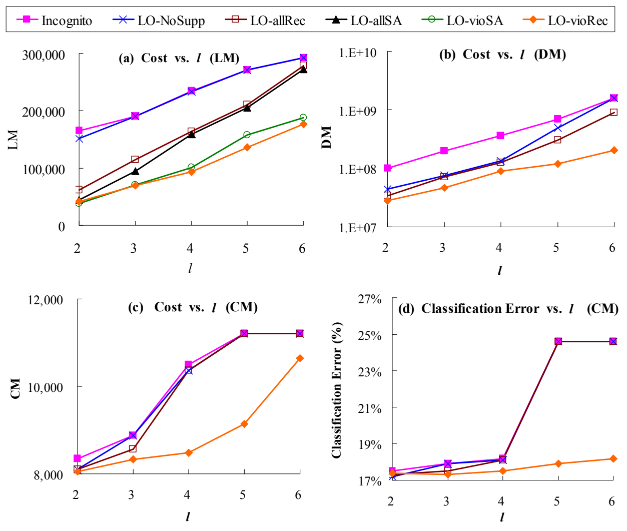


Fig. 5 Comparing with Incognito on optimal cost and classification error

2) Information Loss with a Distinct Threshold per SA Value

We explore the flexibility of our l^+-diversity principle, i.e., set a different θ_i for a different SA value s_i as follows. Givin θ be a baseline threshold, θ_i for s_i is determined by θ and the sensitivity of s_i. If the frequency of s_i in the original table, $fr(s_i)$, is below the average frequency fr_{avg}, s_i is deemed very sensitive and $\theta_i = \theta$, otherwise s_i is deemed less sensitive and $\theta_i = \theta \cdot fr(s_i) / fr_{avg}$. Let θ_{avg} is the average threshold, then $\theta_i \leq \theta_{avg}$ if $fr(s_i) \leq fr_{avg}$. In words, the more sensitive is s_i, the more restrictive is θ_i and the better the protection is for s_i.

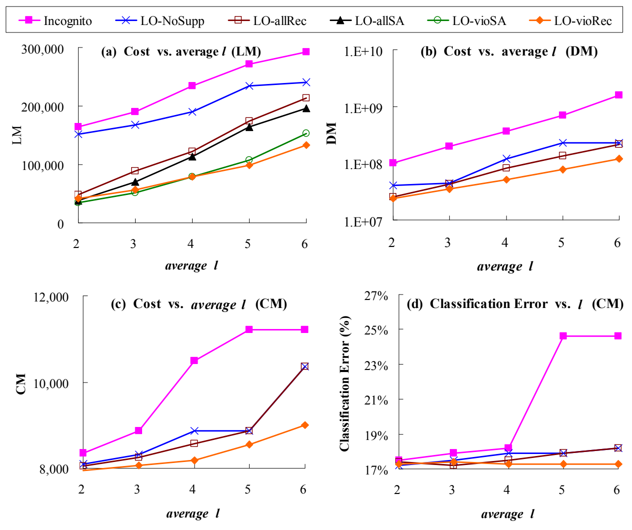


Fig. 6 Comparing with Incognito on optimal cost and classification error by exploring the flexibility of l^+-Optimize: a different θ_i per s_i (*average* $l = 1/\theta_{avg}$)

Fig. 6 shows that all cost curves of l^+-Optimize shift downwards compared to Fig. 5. That is, by making use of the flexibility of our l^+-diversity principle, l^+-Optimize greatly reduced the information loss and classification error with all cost metrics and all suppression schemes. E.g., comparing Fig. 6 with Fig. 5 shows that the information loss by l^+-Optimize is reduced by 13% to 22% with the LM cost metric and the *NoSupp* scheme, 9% to 600% with DM and *NoSupp*, and 8%

to 26% with CM and *NoSupp*, etc. The cost curves of Incognito remain the same as it only uses a uniform l, thus l^+-Optimize preserves even more utility than Incognito.

In summary, the utility gain by adopting the flexible full subtree generalization model is significant. l^+-Optimize further reduces information loss by incorporating suppression as an integral part of anonymization, and provides more protection for more sensitive values by allowing setting privacy thresholds based on sensitivities.

B. Utility Evaluation by Comparing with SimulatedAnnealing

Simulated annealing is a stochastic algorithm that employs a greedy heuristic to search for a good solution, while avoiding being trapped at local optima [11]. Simulated annealing, abbreviated as SimuAnneal, is used as a representative of greedy algorithms to compare with l^+-Optimize. We collected the average best cost at any given time point based on 6 random runs.

Fig. 7 shows the results with LM, all suppression schemes, and l = 5 (θ = 20%). SimuAnneal starts with some solutions not terribly bad and improves the quality *gradually*, and given a long enough time, it can produce a high quality solution. In contrast, l^+-Optimize starts with the most generalized solution and improves the quality *quickly* within a short time. l^+-Optimize finds the best solution by several orders of magnitude faster than SimuAnneal.

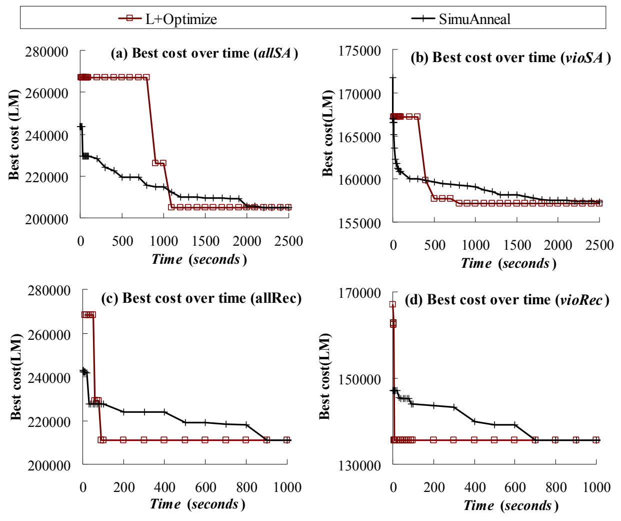


Fig. 7 Comparing with simulated annealing on best cost over time

In short, the gain from the optimal solution relative to heuristic solutions is also significant. This experiment also suggests another usage of l^+-Optimize as an *anytime* algorithm giving time is a constraint.

C. Efficiency and Scalability Evaluation

In this set of experiments, we evaluate the performance of l^+-Optimize by examining two key indicators of the efficiency, the runtime and the percentage of cuts examined.

Fig. 8 shows the runtime (left column) and the percentage of cuts examined (right column) by l^+-Optimize, for the three cost metrics. For most of the cases, 99.99% to 99.97% of cuts are pruned. In other words, typically only 0.01% to 0.03% of cuts are examined, out of which about 2/3 to 9/10 are examined before the optimal solution was found (according to statistics collected in addition). This indicates that our lower bound pruning is highly effective. We observed a strong correlation between the runtime and the percentage of cuts examined. The runtime usually is under a few hundred to a little bit more than a thousand seconds, which is highly efficient giving 356 million cuts in the search space.

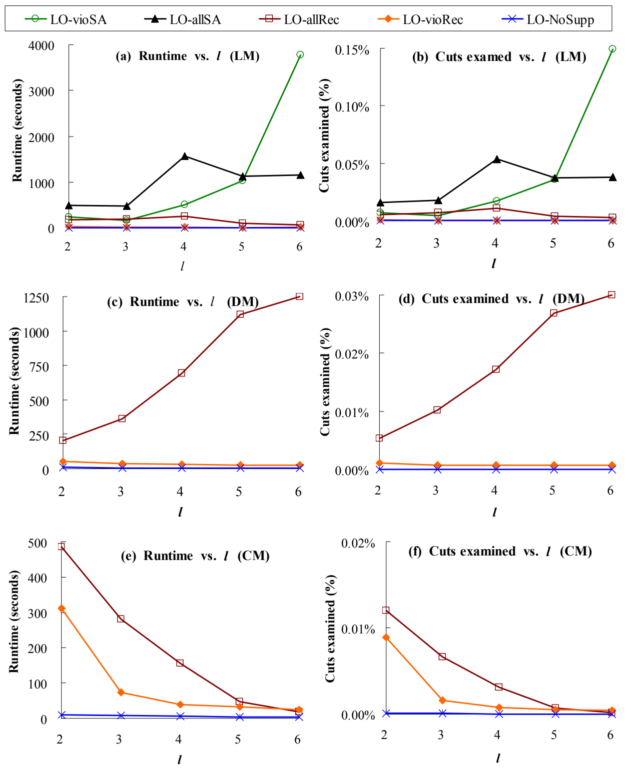


Fig. 8 Performance of l^+-Optimize

In the following, we did a series of experiments to explore the effects of individual pruning techniques, and the impacts of the QI taxonomy selection, the size of QI, and the size of the dataset on the performance. The default setting is as follows: cost metric DM, l = 4 (θ = 25%), and the suppression schemes *NoSupp*, *allRec*, and *vioRec*.

1) Individual Pruning Techniques

We evaluated the effectiveness of the each dynamic technique in Section VI *individually*. "full l^+-Optimize" denotes the algorithm using both techniques, i.e., arranging values in the descending order of occurrences, and updating lower bounds when backtracking. "partial (Asc)" denotes the partial algorithm with values being arranged in the reverse of the proposed order, and "partial (noUpdate)" denotes the partial algorithm without dynamically updating lower bounds on backtracking. It turned out that none of the two partial algorithms terminated within a reasonable time limit with the default QI taxonomy trees. To obtain results for partial algorithms, we reduced the taxonomy trees of the age and native-country attributes to the top two levels. Fig. 9 (a)-(b) shows the runtime and the percentage of cuts examined with DM and *allRec* (the results with the other cost metrics and suppression schemes are similar).

The efficiency of "partial (noUpdate)" drops by a factor of 2 to 3 from the full algorithm, and the efficiency of "partial (Asc)" is several orders of magnitude worse than the full algorithm. That is, the order of values in a cut has much larger impact on the efficiency. Nevertheless, both techniques are critical for achieving a strong pruning.

2) Amplifying the Search Space

We investigated the impact of the selection of QI taxonomies on the efficiency of l^+-Optimize. We created 3 taxonomies with the coarse, medium, and fine granularity respectively, based on the taxonomies in [7] and [10]. Three taxonomies have vastly different search space sizes: the coarse taxonomy yields 152 million (152,685,000) cuts, the medium one yields 7 billion (6,943,460,940) cuts, and the fine one yields 431 billion (431,934,597,900) cuts.

Fig. 9 (c)-(d) show the runtime and the percentage of cuts searched. For a finer taxonomy, the algorithm took a longer time because the search space is very large. However, the increase in the runtime is several orders less than the increase in the size of the search space, which means that the pruning is stronger with a larger search space. For example, with the *allRec* scheme, the search space of the medium taxonomy is 40 times of the coarse taxonomy, but only 0.0017% of cuts for the medium taxonomy were examined compared to 0.02% for the coarse taxonomy. In fact, a finer taxonomy provides more flexibility, which helps us get a smaller running best cost in the early stage of search, so we have a stronger pruning.

3) Varying the Size of QI

We studied the impact of the QI size on pruning. We selected the first *m* attributes listed in Table II to comprise the QI, for *m* = 1, 2, .., 7.

Fig. 9 (e) shows that the runtime increases with the increase of the QI size since the search space size increases quickly with the QI size. Fig 8 (f) shows that the pruning becomes stronger with a larger QI size, which mitigates the impact of the QI size on performance.

4) Amplifying the Size of the Dataset

To study the scalability of l^+-Optimize, we amplified the size of the dataset by inserting α "variants" of each original record into the Adult dataset. A variant of an original record *t* was created by randomly selecting *q* attributes from QI, with *q* being uniformly distributed in the range [1, |QI|], and replacing *t*'s value on each selected attribute with a value randomly drawn from the attribute domain.

Fig. 9 (g) depicts the runtime of l^+-Optimize for 50K to 1000K data records. The runtime is linear to the data size. In fact, the search space depends on the taxonomies of the QI attributes rather than the size of the dataset. Fig. 9(h) shows that there is little change in the number of cuts to be examined. Notice that the overhead pertaining to each cut, mainly comes from partition split / merge operations, and is proportional to the size of the dataset. In short, l^+-Optimize is quite scalable.

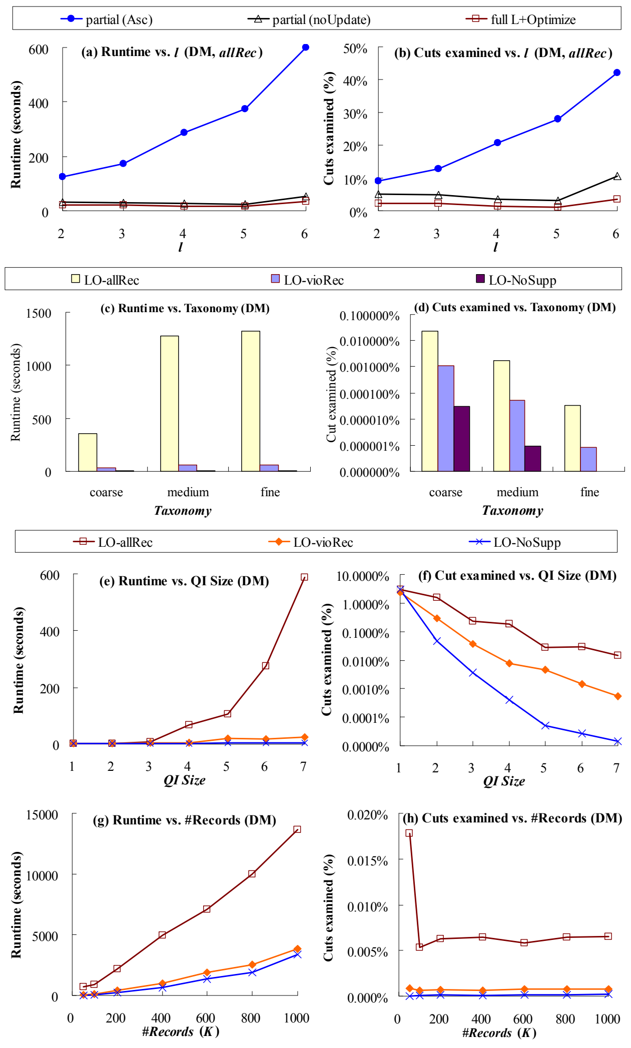


Fig. 9 Performance by (a)(b) different dynamic techniques, (c)(d) taxonomies of different granularities, (e)(f) different QI sizes, (g)(h)different dataset sizes

IX. Additional Issues

We illustrate our choice of the full subtree generalization model and how to handle minimality attacks [26].

A. Choosing the Full Subtree Generalization Model

The full subtree generalization model [10] is a single-dimensional global recoding model [12]. As discussed in Section II, this model ensures the *domain exclusiveness*, which implies that the anonymized data can be analysed by any existing algorithm without modification.

For example, given *T* in Fig. 1(a) and $\theta = 50\%$ ($l = 2$), this model may produce 3 partitions, <Junior, Europe>, <Senior, Europe>, and <University, America> in the anonymized T^*. The values in T^* are *exclusive* of each other. So, we can analyse T^* by any existing algorithm, e.g., mine (generalized) association rules in T^*.

In comparison, the latest multi-dimensional generalization model [8] may produce 5 multi-dimensional regions (partitions), <Junior-or-Senior, UK>, <Senior-or-Bachelor, France-or-Canada>, <Junior-or-Graduate, France-or-Canada>,

<Bachelor-or-Graduate, USA>, and <Bachelor-or-Graduate, UK-or-Canada>. Notice that [8] first maps the *m*-dimension data to 1-dimension, and then maps the *anonymized* 1-dimension data back to *m*-dimension. Now, the values in the anonymized T^* are not exclusive of each other, i.e., they overlap on the domains, which is a flexibility that helps reduce the information loss. However, this comes with a price: we have to either further transform the anonymized data or develop new customized algorithms to analyse the data, e.g., we can not directly run existing algorithms on T^* to mine association rules as we can not count the exact occurrences of values in T^*.

In summary, both the single dimensional generalization models and the multi-dimensional generalization models have their own strength. The latter has less generalization cost, whereas the former produces generalized data that can be processed by existing algorithms.

B. Handling Minimality Attacks

Minimality attacks, identified by [26], arise from the fact that knowledge of the anonymization algorithm, in particular the *minimality principle* employed by most algorithms, can yield clues about how to infer detailed information from the anonymized data. Note that both single dimensional and multi-dimensional generalization models may suffer from minimality attacks.

To prevent minimality attacks, a simple strategy is to apply an additional precaution step to the optimal solution, such as randomly introducing more generalization or suppression. Such a precaution step breaks the minimality principle and can be easily incorporated into our algorithm while the key elements of our approach still work.

Another strategy is to return a solution randomly selected from the top-*h* optimal solutions with *h* being a small integer, which also breaks the minimality principle. At the same time, the returned solution is guaranteed to be among the top-*h* optimal ones. To adopt this strategy, we maintain the current top-*h* best solutions and use the cost of the *h*-th best solution in the pruning condition. All techniques employed in our algorithm remain unchanged.

X. Conclusion

This paper presented an efficient algorithm, l^+-Optimize, for finding an optimal anonymization satisfying l^+-diversity under the full subtree generalization model with various suppression schemes. The algorithm is generic in the sense that it can be instantiated with any reasonable cost metric. Several novel techniques contribute to the efficiency of l^+-Optimize: a novel search strategy, cost lower bounding methods, and pruning techniques. The new findings are: by considering a large search space, the optimal solution has a significant utility gain compared to heuristic solutions; by setting each sensitive value's privacy threshold according to its sensitivity, we can provide better protection for more sensitive values and incur less information loss; it is possible to find the optimal solution efficiently on real world datasets by employing strong pruning strategies.

Acknowledgment

The research is supported in part by the Natural Sciences and Engineering Research Council of Canada, and in part by the Science and Technology Development Plan of Zhejiang Province, China (2006C21034).

We thank the anonymous reviewers for their helpful comments.

References

[1] R. Agrawal and R. Srikant. *Privacy preserving data-mining*. In *SIGMOD*, pages 439-450, 2000.

[2] A. Asuncion and D.J. Newman. *UCI Machine Learning Repository*, 2007. Irvine, CA: Uni of California, School of Information and Computer Science.

[3] R. J. Bayardo Jr. and R. Agrawal. *Data privacy through optimal k-anonymization*. In *ICDE*, pages 217-228, 2005.

[4] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. *Secure anonymization for incremental datasets*. In *VLDB* (*SDM*), 2006.

[5] B. Chen, K. LeFevre, and R. Ramakrishnan. *Privacy skyline: privacy with multidimensional adversarial knowledge.* In *VLDB*, 2007.

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

[7] B. Fung, K. Wang, and P. Yu. *Top-down specialization for information and privacy preservation*. In *ICDE*, 2005.

[8] G. Ghinita, P. Karras, P. Kalnis, N. Mamoulis. *Fast data anonymization with low information loss*. In *VLDB*, 2007.

[9] T. Iwuchukwu and J. Naughton. *k-Anonymization as spatial indexing: Toward scalable and incremental anonymization*. In *VLDB*, 2007.

[10] V. Iyengar. *Transforming data to satisfy privacy constraints*. In *SIGKDD*, pages 279-288, 2002.

[11] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi. *Optimization by simulated annealing*. Science, New Series 220 (4598): 671-680, 1983.

[12] K. LeFevre, D. DeWitt, and R. Ramakrishnan. *Incognito: Efficient full-domain k-anonymity*. In *SIGMOD*, 2005.

[13] K. LeFevre, D. DeWitt, and R. Ramakrishnan. *Mondrian multidimensional k-anonymity*. In *ICDE*, 2006.

[14] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. *l-Diversity: Privacy beyond k-anonymity*. In *ICDE*, 2006.

[15] D. J. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Y. Halpern. *Worst-case background knowledge for privacy preserving data publishing*. In *ICDE*, 2007.

[16] A. Meyerson and R. Williams. *On the complexity of optimal k-anonymity*. In *PODS*, pages 223-228, 2004.

[17] M. Nergiz, M. Atzori, and C. W. Clifton. *Hiding the presence of individuals from shared databases*. In *SIGMOD*, 2007.

[18] S. Papadimitriou, F. Li, G. Kollios, and P. Yu. *Time Series Compressibility and Privacy*. In *VLDB*, 2007.

[19] J. Pei, J. Xu, Z. Wang, W. Wang, and K. Wang. *Maintaining k-anonymity against incremental updates*. In *SSDBM*, 2007.

[20] V. Rastogi, S. Hong, and D. Suciu. *The Boundary Between Privacy and Utility in Data Publishing*. In *VLDB*, 2007.

[21] P.Samarati. *Protecting respondents' identities in microdata release*. In *TKDE*, 13(6): 1010-1027, 2001.

[22] P. Samarati and L. Sweeney. *Generalizing data to provide anonymity when disclosing information*. In *PODS*, 1998.

[23] L. Sweeney. *Achieving k-anonymity privacy protection using generalization and suppression*. Int'l Journal on Uncertainty, Fuzziness, and Knowledge-Base Systems 10(5): 571-588, 2002.

[24] Y. Tao, X. Xiao, J. Li, D. Zhang. *On Anti-Corruption Privacy Preserving Publication*. In *ICDE*, 2008.

[25] K. Wang, B.C.M. Fung, and P.S. Yu. *Template-based privacy preservation in classification problems*. In *ICDM*, 2005.

[26] R.C. Wong, A.W. Fu, K. Wang, J. Pei. *Minimality attack in privacy preserving data publishing*. In *VLDB*, 2007.

[27] X. Xiao and Y. Tao. *Anatomy: Simple and effective privacy preservation*. In *VLDB*, pages 139-150, 2006.

[28] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. *Aggregate Query Answering on Anonymized Tables*. In *ICDE*, 2007.