ARUBA: A Risk-Utility-Based Algorithm for Data Disclosure

Mohamed R. Fouad, Guy Lebanon, and Elisa Bertino

Abstract

Dealing with sensitive data has been the focus of much of recent research. On one hand data disclosure may incur some risk due to security breaches, but on the other hand data sharing has many advantages. For example, revealing customer transactions at a grocery store may be beneficial when studying purchasing patterns and market demand. However, a potential misuse of the revealed information may be harmful due to privacy violations. In this paper we study the tradeoff between data disclosure and data retention. Specifically, we address the problem of minimizing the risk of data disclosure while maintaining its utility above a certain acceptable threshold. We formulate the problem as a discrete optimization problem and leverage the special monotonicity characteristics for both risk and utility to construct an efficient algorithm to solve it. Such an algorithm determines the optimal transformations that need to be performed on the microdata before it gets released. These optimal transformations take into account both the risk associated with data disclosure and the benefit of it (referred to as utility). Through extensive experimental studies we compare the performance of our proposed algorithm with other date disclosure algorithms in the literature in terms of risk, utility, and time. We show that our proposed framework outperforms other techniques for sensitive data disclosure.

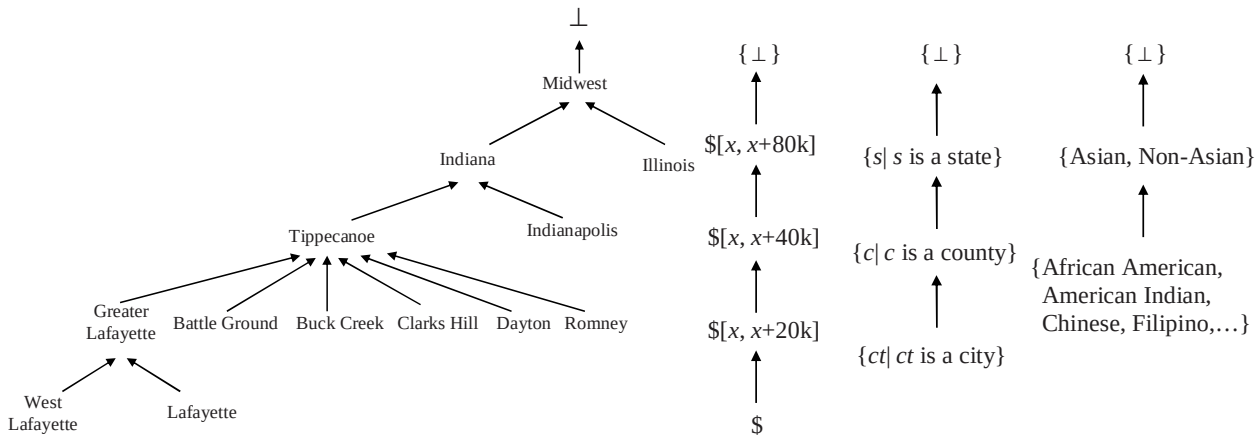
Index Terms

rivacy, Security, Risk Management, Data Sharing, Data Utility, Anonymity.rivacy, Security, Risk Management, Data Sharing, Data Utility, Anonymity.P

I. Introduction

Maximizing data usage and minimizing privacy risk are two conflicting goals. Disclosing the minimum amount of information (or no information at all) is compelling specially when organizations try to protect the privacy of individuals. To achieve such goal, the organizations typically try to (1) hide the identity of individual to whom data pertains, and (2) apply a set of transformations to the microdata before releasing it. These transformations include data suppression, data generalization, and data perturbation. Data suppression refers to suppressing certain attribute values (or equivalently disclosing the value $\perp$). Data generalization [15] refers to releasing a less specific variation of the original data; for example, releasing 479** for the zip code instead of 47906. In data generalization a value generalization hierarchy (VGH) for each attribute is constructed and consulted whenever a generalization is to take place (see Fig. 1(a) for an example of the VGH for the *city* attribute). Data perturbation [9] adds noise directly to the original data values; for example, perturbing a numeric value such as a salary by a Gaussian noise. In this paper, we focus on the technique of data generalization which includes data suppression as a special case.

M. Fouad and E. Bertino are with the Department of Computer Science, Purdue University, USA, E-mail: {mrf,bertino}@cs.purdue.edu
G. Lebanon is with the Department of Statistics, Purdue University, USA, E-mail: lebanon@stat.purdue.edu



(a) A partial VGH for *city* (b) DGHs for *salary*, *city*, and *race*

Fig. 1. Value generalization hierarchy (VGH) and domain generalization hierarchy (DGH)

We measure the harmful effect due to the disclosure of private data using the notion of an expected loss or a risk. This loss could be incurred, for example, as a result of privacy violations, financial loss due to identity theft, and security breaches. On the other hand, releasing data has its own merits. Released data could be useful for data mining and research purposes, data sharing, and improved service provisioning. Examples of risk-utility conflicts include, but not limited to, (i) medical research benefits vs. fear of patients' privacy violation, (ii) detecting purchasing patterns of customers vs. privacy of customers transactions, and (iii) benefits of disclosing sensitive geospatial data (for example, maps) vs. threats to national security.

Releasing more general information seems to have a diminishing effect on both risk and utility. However, the fact that we have opposite goals for risk and utility (minimizing the risk and maximizing the utility) raises the following crucial question: "Up to what level of generalization can we tolerate?". Indeed, without the help of powerful models that asses the risk and utility of a given information item, answering the above question is impossible. Many models have been proposed to quantify data utility all of which show that data generalization has negative impact on how useful data is. Xiao et al. [17] define the information loss of a more general attribute value v^* in terms of the number of values that it represents. Under the approach by Bayardo and Agrawal [1] and Xu et al. [18], a penalty cost is assigned to a generalized or suppressed tuple to reflect the information loss in such transformations. Fung et al. [3] define a tuple information in terms of the number of records that could be generalized to this tuple. An entropy-based model to assess information gain/loss is adopted in the approach by Wang et al. [16]. From the proposed models it is evident that when the released records are generalized to a greater extent, a larger information loss is incurred.

Assessing the risk of releasing a given information item has also been the subject of recent research. Assessing the risk is a more challenging task than quantifying the utility and there exist only very few models for assessing risk. Intuitively, releasing more specific information will incur a higher risk than releasing general information. Cheng et al. [2] model the risk of a tuple in terms of the value of information contained in it. A privacy risk model has been proposed by Lebanon et al. [6] that takes into account both the entity identification and the sensitivity of the disclosed information.

In this paper we propose an efficient algorithm (ARUBA) to address the tradeoff between data utility

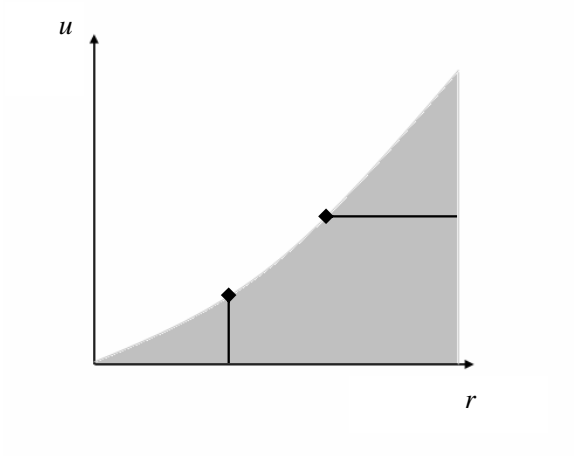


Fig. 2. Space of disclosure rules and their risk and expected utility. The shaded region correspond to all achievable disclosure policies

and data privacy. ARUBA operates on the microdata to identify the optimal set of transformations that need to be applied in order to minimize the risk and in the meantime maintain the utility above a certain threshold.

The rest of the paper is organized as follows. The problem statement is presented in Section II. Section III introduces the basic definitions and terminology used throughout the paper. Different risk and utility models are discussed in Section IV. In section V, we develop an efficient scalable algorithm for data disclosure. Experimental results that show the superiority of our proposed algorithm over existing algorithms are reported in Section VI. Section VII surveys related work. Finally, Section VIII presents concluding remarks and outlines future work.

II. Problem Statement

In this paper we consider the problem of identifying the optimal set of transformations which, when carried out on a given table, generate a resulting table that satisfies a set of optimality constraints. The optimality constraints are defined in terms of a preset objective function as well as risk and utility conditions.

The relationship between the risk and expected utility is schematically depicted in Fig. 2 which displays different instances of a disclosed table by their 2-D coordinates (r, u) representing their risk and expected utility, respectively. In other words, different generalization procedures poss different utility and risk which lead to different locations in the (r, u)-plane. The shaded region in the figure corresponds to the set of feasible points (r, u) (i.e., the risk and utility are achievable by a certain disclosure policy) whereas the unshaded region corresponds to the infeasible points. The vertical line corresponds to all instances whose risk is fixed at a certain level. Similarly, the horizontal line corresponds to all instances whose expected utility is fixed at a certain level. Since the disclosure goal is to obtain both low risk and high expected utility, we are naturally most interested in disclosure policies occupying the boundary of the shaded region. Policies in the interior of the shaded region can be improved upon by projecting them to the boundary.

The vertical and horizontal lines suggest the following two ways of resolving the risk-utility tradeoff. Assuming that it is imperative that the risk remains below a certain level, we can define the problem as

$$\text{maximize } u \qquad \text{subject to} \qquad r \leq c. \tag{1}$$

Alternatively, insisting on having the expected utility to be no less than a certain level we can define the problem as

$$\text{minimize } r \qquad \text{subject to} \qquad u \geq c. \tag{2}$$

A more symmetric definition of optimality is given by

$$\text{minimize } (r - \lambda u) \tag{3}$$

where $\lambda \in \mathbb{R}_+$ is a parameter controlling the relative importance of minimizing risk and maximizing utility.

In this paper, without loss of generality, we model our problem as in (2). Specifically, we address the problem of identifying the optimal transformations that produce the minimum risk and lower bound the utility above a given threshold. Given a specific tuples $\boldsymbol{a} = \langle a_1, a_2, \cdots, a_i, \cdots, a_k \rangle$, the following problem has to be solved:

$$\boldsymbol{t}^* = \arg\min_{\boldsymbol{t}} \ r(\boldsymbol{t}) \text{ subject to } u(\boldsymbol{t}) \geq c \tag{4}$$

where $\boldsymbol{t}$ is a generalization of $\boldsymbol{a}$.

III. Notations and Definitions

Throughout the paper, we will usually refer to an arbitrary record as $\boldsymbol{a}$ or $\boldsymbol{b}$ and to a specific record in a particular database using a subscript $\boldsymbol{a}_i$. Attributes are denoted by A_i (or simply A). Attribute values of A are represented using the notation $[\boldsymbol{a}]_j$ (or $[\boldsymbol{a}_i]_j$) or just a_j (or a_{ij}). Note the "**bold**" typesetting representing vector notation and the "non-bold" typesetting representing attribute values. A collection of n records such as a database is denoted by ($\boldsymbol{a}_1$,...,$\boldsymbol{a}_n$).

Definition 1: The *depth* of an attribute value a_i corresponding to attribute A, denoted by $depth(a_i)$, is the length of the path from a_i to $\perp$ in the VGH corresponding to A, that is, the maximum possible number of generalization steps applicable to this value.

Example 1: In the VGH shown in Fig. 1(a), $depth(\text{Greater Lafayette}) = 4$.

Definition 2: The *generalization set* of an attribute value a_i corresponding to attribute A, $GE(a_i)$, is the set of all ancestors of a in the VGH corresponding to A. We denote any element in $GE(a_i)$ as $\widehat{a}$. The *parent* of a_i is the immediate ancestor and is denoted by $parent(a_i)$. On the other hand, the *specialization set* of an attribute value a_i, $SP(a_i)$, is the set of all descendants of a_i in the VGH corresponding to A. That is, $\forall_{a_i \in SP(\widehat{a_i})} \ \widehat{a_i} \in GE(a_i)$. The *child* of a_i is the immediate descendent and is denoted by $child(a_i)$.

Example 2: In the VGH shown in Fig. 1(a), $GE(\text{Lafayette}) = \{\text{Greater Lafayette, Tippecanoe, Indiana, Midwest, } \perp\}$, and $SP(\text{Greater Lafayette}) = \{\text{West Lafayette, Lafayette}\}$.

Definition 3: An *immediate generalization* of a record $\boldsymbol{a} = \langle a_1, a_2, \cdots, a_i, \cdots, a_k \rangle$ with respect to an attribute a_i is a transformation on this record in which the value a_i is replaced by $parent(a_i)$ from the corresponding VGH. It is denoted by $ig_{a_i}(\mathbf{a})$, that is, $ig_{a_i}(\mathbf{a}) = \langle a_1, a_2, \cdots, parent(a_i), \cdots a_k \rangle$. The set of all immediate generalizations of a record $\mathbf{a}$ is denoted by $IG(\mathbf{a}) = \bigcup_{i=1}^{k} \ ig_{a_i}(\mathbf{a})$. The set of all generalizations of a record $\mathbf{a}$ is denoted by $G(\mathbf{a})$.

Lemma 1: The risk and utility associated with a record $\boldsymbol{a}$ ($r(\boldsymbol{a})$ and $u(\boldsymbol{a})$, respectively) have the following property:

$$r(\boldsymbol{a}) \geq r(ig_{a_i}(\boldsymbol{a})) \text{ and } u(\boldsymbol{a}) \geq u(ig_{a_i}(\boldsymbol{a})), \ \forall i : 1, 2, \ldots, k.$$

This property, which we refer to as the *monotonicity property*, can be easily verified for most standard definitions of utility and risk.

Definition 4: An *immediate specialization* of a record $\boldsymbol{a} = \langle a_1, a_2, \cdots, a_i, \cdots, a_k \rangle$ with respect to an attribute a_i is a transformation on this record in which the value a_i is replaced by $child(a_i)$ from the corresponding VGH. It is denoted by $is_{a_i}(\mathbf{a})$, that is, $is_{a_i}(\mathbf{a}) = \langle a_1, a_2, \cdots, child(a_i), \cdots a_k \rangle$. The set of all immediate specializations of a record **a** is denoted by $IS(\mathbf{a}) = \bigcup_{i=1}^{k} is_{a_i}(\mathbf{a})$. The set of all specialization of a record **a** is denoted by $S(\mathbf{a})$. Note that $|IG(\mathbf{a})| \leq k$ and $|IS(\mathbf{a})| \leq k$.

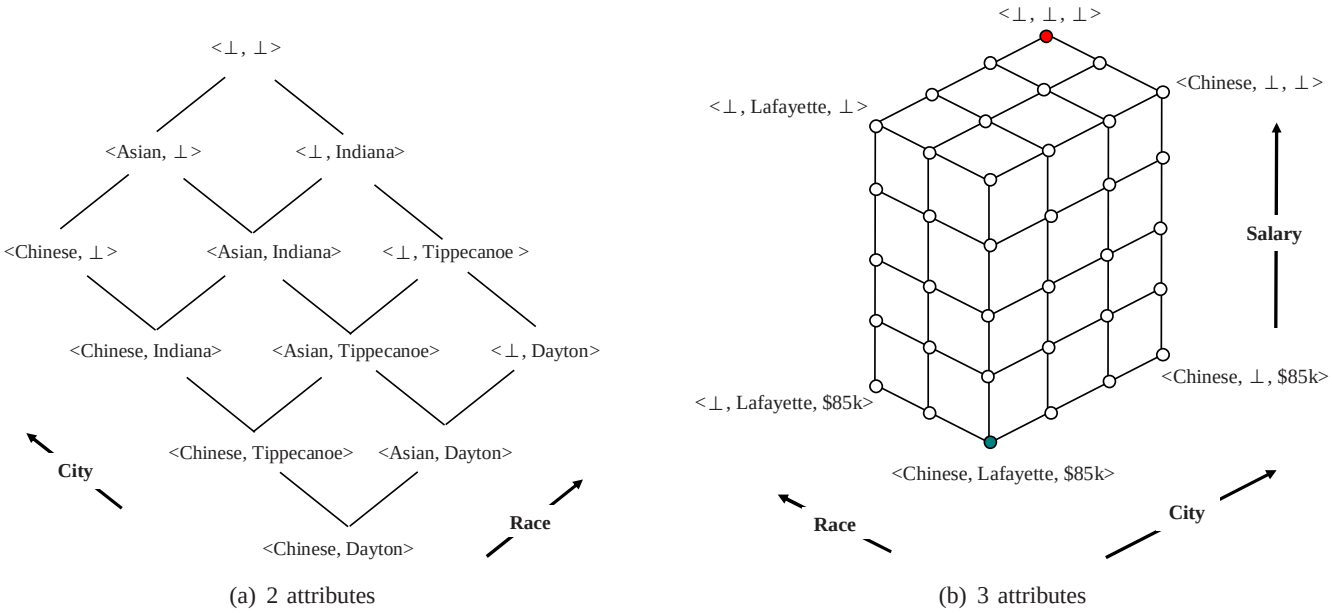
Example 3: In Fig. 3(a), $IG(\langle$Chinese, Tippecanoe$\rangle) = \{\langle$Asian, Tippecanoe$\rangle$, $\langle$Chinese,Indiana$\rangle\}$ and $IS(\langle$Chinese, Tippecanoe$\rangle) = \{\langle$Chinese,Dayton$\rangle\}$.

Definition 5: A *generalization lattice* for a given record $\mathbf{a} = \langle a_1, a_2, \cdots, a_i, \cdots, a_k \rangle$ is the lattice formed by the immediate generalization relation on the set $\big(\{a_1\} \cup GE(a_1)\big) \times \big(\{a_2\} \cup GE(a_2)\big) \cdots \times \big(\{a_k\} \cup GE(a_k)\big)$. It is a graph (V, E) where $V = \big(\{a_1\} \cup GE(a_1)\big) \times \big(\{a_2\} \cup GE(a_2)\big) \cdots \times \big(\{a_k\} \cup GE(a_k)\big)$ and $E = \{(v_1, v_2) |\ v_1, v_2 \in V\ \wedge\ v_1 \in IG(v_2) \cup IS(v_2)\}$. The *dimension* of the lattice is the number of attributes of the initial record, that is, k.

Lemma 2: The generalization lattice for a given record $\mathbf{a} = \langle a_1, a_2, \cdots, a_i, \cdots, a_k \rangle$ has $\Pi_{i=1}^{k} \big(depth(a_i) + 1\big)$ nodes.

Definition 6: A *border node* **a** is a lattice vertex that satisfies the following condition: $|IG(\mathbf{a})| < k$ or $|IS(\mathbf{a})| < k$. It is the node in which at least one of the attributes cannot be further generalized or cannot be further specialized. Otherwise, if $|IG(\mathbf{a})| = |IS(\mathbf{a})| = k$, **a** is called an *inner node*.

Example 4: In Fig. 3(a), $\langle$Chinese, Tippecanoe$\rangle$ is a border node whereas $\langle$Asian, Indiana$\rangle$ is an inner node.



(a) 2 attributes (b) 3 attributes

Fig. 3. Example of 2D and 3D lattices

Fig. 1(b) shows examples of domain generalization hierarchies for the *race*, *city*, and *salary* attributes. Using these hierarchies, two lattices representing specific records with different number of attributes are depicted in Fig. 3. Notice that moving in one dimension is equivalent to generalizing the attribute that corresponds to this dimension. Moreover, the dimension of the lattice is the number of attributes and the

size of each dimension is the number of generalization steps for the corresponding attribute.

Definition 7: A *feasible node* is the lattice vertex that satisfies all the given constraints that are mentioned in equations (1) and (2). Otherwise, it is called *infeasible node*. The best feasible node is called the *optimal node*.

Note that all the children of a feasible node are also feasible and all the parents of an infeasible node are also infeasible.

IV. Risk and Utility Computation

Our proposed algorithms make use of existing tools to quantify the utility and risk of a given tuple. In order to determine whether a tuple $\boldsymbol{a}$ is feasible, one needs to compute $u(\boldsymbol{a})$. On the other hand, the proposed algorithms consider the objective function of minimizing the risk. Therefore, it is imperative that, given a tuple $\boldsymbol{a}$, a tool for quantifying risk $r(\boldsymbol{a})$ exists. In this section, we describe some models that have been proposed in the literature for utility and risk assessment. It is worth to note that all these models intuitively adhere to the fact that both risk and utility increase as the disclosed data becomes more specific and decrease as the disclosed data becomes more general.

A. Utility Assessment Models

Utility assessment models are often specified in terms of the number of leaves of the VGH subtree rooted at each attribute value. Specifically, one way to assess the utility of a record $\mathbf{a} = \langle a_1, a_2, \cdots, a_i, \cdots, a_k \rangle$ is

$$u(\boldsymbol{a}) = \sum_{i=1}^{k} 1/n_i, \tag{5}$$

where n_i is the number of leaf nodes of the VGH rooted at a_i. Note that, this model has a few disadvantages. According to this model, a non-zero (although minimum) value is assigned to the most general node and the utility of the leaf nodes is k. A variation of (5) is to use a logarithmic function as in

$$u(\boldsymbol{a}) = \sum_{i=1}^{k} \ln(m_i/n_i), \tag{6}$$

where m_i and n_i are the total number of leaf nodes of the VGH and the number of leaf nodes of the VGH subtree rooted at a_i, respectively. In agreement with our intuition, equation (6) assigns zero utility for the most general node.

Instead of taking into account the number of leaf nodes as a metric for utility assessment, one may consider attribute depths as defined in Definition 1, for example $\sum_{i=1}^{k} depth(a_i)$ (the sum of the heights of all VGHs *minus* the number of lattice generalization steps that are performed to obtain the record $\boldsymbol{a}$). As data gets more specific, its depth increases and, accordingly, so does the utility. As in the previous case, the utility of the most general node $(\langle \perp, \perp, \cdots, \perp \rangle)$ is zero.

In some cases, information loss, denote by $\triangle u$, can be used in lieu of utility. Maximizing the utility u is analogous to minimizing the information loss $\triangle u$ and, therefore, it is straightforward to transfer the optimization problem from one of these utility measures to the other. Xiai and Tao [17] defined the information loss as follows: $\triangle u(\boldsymbol{a}) = \sum_{i=1}^{k} (n_i - 1)/m_i$, where m_i and n_i are defined as above. Likewise, Iyengar [4] proposes the LM loss metric which is based on summing up normalized information losses for each attribute i.e. LM $= \triangle u(\boldsymbol{a}) = \sum_{i=1}^{k} (n_i - 1)/(m_i - 1)$.

B. Risk Assessment Models

Lebanon et al. [6] have proposed an analytical model to quantify the privacy risk. The risk of disclosing a record $\boldsymbol{a}$ is decomposed into two parts: (i) the user-specified data sensitivity $\Phi(\boldsymbol{a})$, and (ii) the attacker's probability of identifying the data owner based on $\boldsymbol{a}$ and side information θ. Data sensitivity is a subjective and personalized measure, for example $\Phi(\boldsymbol{a}) = \sum_{i\,:\,a_i \neq \perp} w_i$, where w_i represents the sensitivity of the attribute value a_i to the user who owns this data. The second component of the risk corresponding to the attacker's probability of identifying the data owner is given by $1/|\rho(\boldsymbol{a}, \theta)|$ where $|\rho(\boldsymbol{a}, \theta)|$ is the number of entries in the database θ consistent with the disclosed data $\boldsymbol{a}$ (anonymity number). Multiplying the two components we obtain

$$r(\boldsymbol{a}, \theta) = \frac{\Phi(\boldsymbol{a})}{|\rho(\boldsymbol{a}, \theta)|},$$

The database θ is assumed to be the side information available to the attacker but, assuming it is unknown, replacing it with the original database of pre-disclosed records provides an upper bound of the risk.

In this paper we consider as risk a more general combination of the data sensitivity Φ and anonymity number $|\rho|$ given by an arbitrary function

$$r(\boldsymbol{a}, \theta) = f(\Phi(\boldsymbol{a}), |\rho(\boldsymbol{a}, \theta)|).$$

Three examples which we concentrate on are:

- **Model I:** $f_1(x, y) = x/y$ which leads to the risk proposed by Lebanon et al. [6].
- **Model II:** $f_2(x, y) = 1/y$ which leads to non-personalized and constant data sensitivity.
- **Model III:** $f_3(x, y) = x \log(1/y)$ corresponding to an entropic measure emphasizing small values of $1/|\rho|$.

By means of each of the above risk models, in the next section we compute the risk associated with data disclosure to compare between our proposed algorithms, discrete optimization algorithm, and k-anonymity.

V. Algorithms For Optimal Data Disclosure

Taking into account the special the nature of the optimization problem at hand as well as the monotonicity property of both risk and utility, the discrete optimization problem (4) reduces to the following problem: Given a record $\boldsymbol{a}$, it is required to

$$\text{minimize } r(\boldsymbol{a}^{(x_1, x_2, \ldots, x_i, \ldots, x_k)})$$

subject to

$$u(\boldsymbol{a}^{(x_1, x_2, \ldots, x_i, \ldots, x_k)}) \geq c, \quad 0 \leq x_i \leq h_i, \quad \forall i : 1, 2, \ldots, k$$

where: $h_i = depth(a_i)$, x_i represents the number of generalization steps applied on the i^{th} attribute value of the record $\boldsymbol{a}$, and $\boldsymbol{a}^{(x_1, x_2, \ldots, x_i, \ldots, x_k)}$ is the resulting record after applying these generalization steps. Moreover, the risk and utility satisfy the following:

$$r(\boldsymbol{a}^{(x_1, x_2, \ldots, x_i, \ldots, x_k)}) \leq r(\boldsymbol{a}^{(x_1, x_2, \ldots, x_i+1, \ldots, x_k)}),$$
$$u(\boldsymbol{a}^{(x_1, x_2, \ldots, x_i, \ldots, x_k)}) \leq u(\boldsymbol{a}^{(x_1, x_2, \ldots, x_i+1, \ldots, x_k)}), \ \forall i : 1, 2, \ldots, k.$$

A brute-force method for obtaining the optimal transformations is to try all possible combinations of attribute values and their generalizations and select the transformation that produces a feasible anonymized table which poses the minimum risk. Note that

- a crucial difference between our algorithm and most of the other anonymization algorithms is that we apply the transformations on a record-by-record basis instead of dealing with sets of equivalent records and we capture record similarities by means of the number of consistent records, $|\rho(\boldsymbol{a}, \theta)|$, that is embedded in the risk models;
- the proposed algorithms do not require the construction of the lattice beforehand;
- the risk and utility functions are called as needed;
- checking whether a node v has been visited (i.e., $v \in V$) can be implemented by inserting the nodes in V in a hash table and checking if v, when hashed using the same hashing function, collides with any existing node; and
- the proposed algorithms can be easily extended to handle the dual problem of maximizing the utility subject to a risk constraint.

A. Basic Top-Down Algorithm (BTDA)

In this section we propose a modification of the brute-force algorithm that uses the *priority queue* data structure to navigate through lattice nodes until it reaches the optimal point.

Definition 8: A *priority queue* is a linked list of lattice nodes sorted by risk in ascending order.

Algorithm 1: BTDA Algorithm

Input: A record $\boldsymbol{a} = \langle a_1, a_2, \cdots, a_i, \cdots, a_k \rangle$, a utility threshold c, and risk and utility functions $r(\boldsymbol{a}), u(\boldsymbol{a})$, respectively.
Output: The optimal node $\boldsymbol{a}^*$.

BTDA()

(1) initialize Q, V
`/* Q is priority queue where r() is used to insert a node. V is the set of visited nodes.*/`

(2) insert $\langle \perp, \perp, \cdots, \perp \rangle$ in both Q and V
(3) **while** (The front node, call it ***v***, of Q is infeasible, i.e. $u(\boldsymbol{v}) < c$)
(4) delete ***v*** from Q
(5) insert $IS(\mathbf{v}) - V$ in Q and V

(6) `/* v is the first feasible node with min risk */`

return ***v***

Theorem 1: Algorithm 1 generates the optimal node.

Proof: We prove the theorem by contradiction. Assume that the front node of Q, say ***v***, is feasible but not optimal. This implies that the optimal node is one of the nodes already inserted in Q after ***v*** or

one of their children yet to be inserted. Since children nodes have higher risk than their parents and the parents have higher risk than $\boldsymbol{v}$ (because they are inserted after $\boldsymbol{v}$ in the priority queue), the optimal node $\boldsymbol{a}^*$ has higher risk than $\boldsymbol{v}$ which contradicts with the optimality definition. ■

B. ARUBA

In this section we propose an efficient algorithm, referred to as A Risk-Utility Based Algorithm (ARUBA), to identify the optimal node for data disclosure. The algorithm scans a significantly smaller subset of nodes (the so called *frontier nodes*) that is guaranteed to include the optimal node.

Definition 9: A *frontier node* is a lattice vertex that is feasible and that has at least one infeasible immediate generalization.

Theorem 2: The optimal node is a frontier node.

Proof: First, it is evident that the optimal node, say $\boldsymbol{a}^*$, is feasible. Second, we prove that all its immediate generalizations are infeasible by contradiction. Assume that at least one of its parents, say $\boldsymbol{b} \in IG(\boldsymbol{a}^*)$, is feasible. Since $r(\boldsymbol{b}) \leq r(\boldsymbol{a}^*)$ and $\boldsymbol{b}$ is feasible, then $\boldsymbol{b}$ is better than $\boldsymbol{a}^*$ which contradicts the fact that $\boldsymbol{a}^*$ is the optimal node. Therefore, all immediate generalizations of $\boldsymbol{a}^*$ are infeasible and $\boldsymbol{a}^*$ is thus a frontier node. ■

Definition 10: An *adjacency cube* associated with a lattice vertex $\boldsymbol{v} = \langle v_1, v_2, \cdots, v_i, \cdots, v_k \rangle$ is the set of all nodes $\Big\{ \langle u_1, u_2, \cdots, u_i, \cdots, u_k \rangle | u_i \in \{v_i, parent(v_i), child(v_i)\}$ $\forall i : 1, 2, \cdots, k \Big\} \setminus \{\langle v_1, v_2, \cdots, v_i, \cdots, v_k \rangle\}$. The number of nodes in the adjacency cube $\leq 3^k - 1$.

Example 5: Fig. 4 displays the adjacency cube associated with $\boldsymbol{f}$ is $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{g}, \boldsymbol{h}, \boldsymbol{i}\}$.

Theorem 3: Let $\mathcal{L}$ be a generalization lattice of dimension k. Except for border nodes, a frontier node $\mathbf{f} \in \mathcal{L}$ has at least k frontier neighbors in the adjacency cube associated with it.

Proof: We prove the theorem for the case of 2D lattice. This proof generalizes to more than 2D but the details are omitted due to lack of space. Fig. 4 shows a general section of a 2D lattice. Assume that the node $\boldsymbol{f}$ is a frontier node. There are 2 cases:

- Both $\boldsymbol{c}$ and $\boldsymbol{e}$ are infeasible. If $\boldsymbol{b}$ is feasible, then it is a frontier node (since $\boldsymbol{c}$ is infeasible). Otherwise, $\boldsymbol{c}$ is a frontier node. The same argument applies to nodes $\boldsymbol{e}$, $\boldsymbol{g}$, and $\boldsymbol{h}$.
- One of $\boldsymbol{c}$ and $\boldsymbol{e}$ is infeasible. Assume, without loss of generality, that $\boldsymbol{c}$ is infeasible and $\boldsymbol{e}$ is feasible. Since $\boldsymbol{c}$ is infeasible, then $\boldsymbol{d}$ is infeasible and, therefore, $\boldsymbol{e}$ is a frontier node. Moreover, if $\boldsymbol{b}$ is feasible, then it is a frontier node (since $\boldsymbol{c}$ is infeasible). Otherwise, $\boldsymbol{a}$ is a frontier node.

In both cases, the frontier node $\boldsymbol{f}$ has two frontier neighbors in its adjacency cube. ■

Algorithm 2: ARUBA Algorithm

Input: A record $\boldsymbol{a} = \langle a_1, a_2, \cdots, a_i, \cdots, a_k \rangle$, a utility threshold c, and risk and utility functions $r(\boldsymbol{a}), u(\boldsymbol{a})$, respectively.

Output: The optimal node $\boldsymbol{a}^*$.

ARUBA()

(1) initialize S, V

`/* S is the set of uninvestigated frontier nodes, V is the set of visited nodes. */`

(2) locate an initial frontier node $\boldsymbol{f}$, update V

(3) set $r^* = r(\boldsymbol{f})$

(4) set $\boldsymbol{a}^* = \boldsymbol{f}$

(5) $S = S \cup \boldsymbol{f}$

(6) **while** $(S \neq \Phi)$

(7) extract $\boldsymbol{v}$ from S

(8) **if** $r(\boldsymbol{v}) \leq r^*$

(9) set $r^* = r(\boldsymbol{v})$

(10) set $\boldsymbol{a}^* = \boldsymbol{v}$

(11) locate the set of uninvestigated neighboring frontier nodes in the adjacency cube associated with $\boldsymbol{v}$, call it NF

(12) update V

(13) $S = S \cup NF$

(14) `/* All frontier nodes are scanned and` $\boldsymbol{a}^*$ `is the node with min risk */`

return $\boldsymbol{a}^*$

Theorem 4: Algorithm 2 generates the optimal node.

Proof: The proof follows directly from Theorem 3 in that all frontier nodes will have been visited when Algorithm 2 terminates. Since the optimal node is a frontier node (from Theorem 2), Algorithm 2 will generate the optimal node. ■

The initial frontier node may be obtained by (i) using binary search to locate the node with a utility closest to c given the maximum utility (utility for the most specific node), or (ii) navigating through a random path.

C. Example

For the sake of illustration, consider the simple 2D lattice in Fig. 5. The subscripts assigned to each node are hypothetical risks and utilities satisfying the monotonicity property. The figure show the feasible nodes

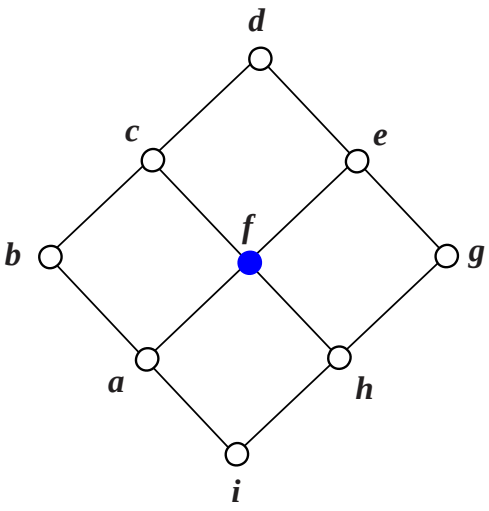


Fig. 4. Neighboring frontier nodes

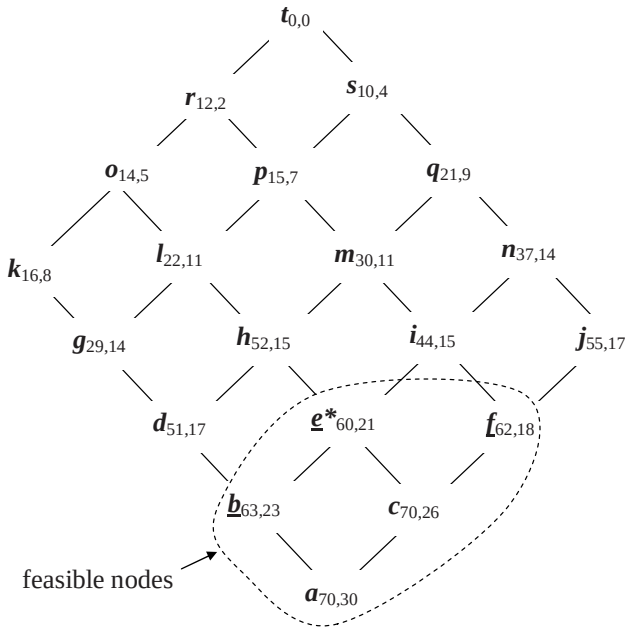


Fig. 5. Illustrative example for $\min r$ s.t. $u \geq 18$ (the feasible nodes are shown, the frontier nodes are underlined, the subscripts of each node give the hypothetical risk and utility, respectively)

Iter. #	Front ↓	of Q			Visited nodes V
1	$t_{0,0}$				*t*
2	$s_{10,4}$	$r_{12,2}$			*t s r*
3	$r_{12,2}$	$p_{15,7}$	$q_{21,9}$		*t s r p q*
4	$o_{14,5}$	$p_{15,7}$	$q_{21,9}$		*t s r p q o*
5	$p_{15,7}$	$k_{16,8}$	$q_{21,9}$	$l_{22,11}$	*t s r p q o k l*
⋮					
15	$j_{55,17}$	$e_{60,21}$	$f_{62,18}$	$b_{63,23}$	*t s r p q o k l m g n h d i j e f b*
16	$e_{60,21}$	$f_{62,18}$	$b_{63,23}$		*t s r p q o k l m g n h d i j e f b*

Iter. #	Unvisited frontier nodes S	Visited nodes V
0	*f*	*a c f j*
1	*e*	*a c f j e i n*
2	*b*	*a c f j e i n b d h m*
3		*a c f j e i n b d h m*

Fig. 6. For the lattice shown in Fig. 5, a list of visited visited nodes at different iterations of Algorithm 1 (left) and Algorithm 2 (right)

with the frontier nodes underlined and the optimal node identified as $\boldsymbol{e}^*$. We assume a risk minimization problem subject to $u \geq 18$.

First, we apply Algorithm 1 on the displayed lattice. Fig. 6 shows the status of the priority queue Q and the set of visited nodes V after the execution of each iteration of the algorithm (steps 3,4,5). The algorithm starts off by inserting the most general node ***t*** in Q and V. Due to the fact that it is infeasible, ***t*** is removed from Q and its unvisited immediate specializations are inserted in Q in ascending order of risk (***s*** then ***r***). The algorithm goes on until the node at the front of Q is feasible (node ***e*** in iteration #16). At the end of the execution the queue contains the frontier nodes and the number of visited nodes is 18.

We also apply Algorithm 2 on the same lattice. The algorithm starts from node ***a*** and assume that the first frontier node to be visited is ***f***. Along the path to ***f***, the nodes ***a, c, f, j*** are visited before determining that ***f*** is a frontier node. Node ***f*** is inserted in S. In the next iteration, the uninvestigated nodes in the adjacency cube of ***f*** are visited (nodes ***e, i, n***) where it is determined that ***e*** is a frontier node and needs to be inserted in S. The algorithm continues until S is empty. Fig. 6 shows the status of the set of uninvestigated frontier nodes S and the set of visited nodes V after the execution of each iteration of the algorithm (steps 6 through 13). At the end of execution, the algorithm has visited all frontier nodes and determined that ***f*** is the optimal node. The number of visited nodes in this case is 11 which is, considering the small scale of the lattice, still a good improvement over Algorithm 1.

VI. Experiments

We conducted our experiments on a real Wal-Mart database. An `item description` table of more than 400,000 records each with more than 70 attributes is used in the experiments. Part of the table is used to represent the disclosed data whereas the whole table is used to generate the attacker's dictionary. Throughout all our experiments, the risk components are computed as follows. First, the identification risk is computed by using the Jaro distance functionto identify the dictionary items consistent with a released record to a certain extent (we used 80% similarity threshold to imply consistency.) Second, the sensitivity of the disclosed data is assessed by means of an additive function and random weights that are generated using a uniform random number generator. The heights of the generalization taxonomies VGHs are chosen to be in the range from 1 to 5.

We use a modified harmonic mean to compute the sensitivity of a parent node w_p with l immediate children given the sensitivities of these children w_i: $w_p = \frac{1}{\sum_{1 \leq i \leq l} \frac{1}{w_i}}$ with the exception that the root node (corresponding to suppressed data) has a sensitivity weight of 0. Clearly, the modified harmonic mean satisfies the following properties: (i) the sensitivity of any node is greater than or equal to zero provided that the sensitivity of all leaves are greater than or equal to zero, (ii) the sensitivity of a parent node is always less than or equal (in case of 1 child) the sensitivity of any of its descendent nodes, and (iii) the higher the number of children a node has the lower the sensitivity of this node is. For example, given a constant city weight w_c, the weight of the `County` node j in the VGH for the `City` is $\frac{1}{\sum_{1 \leq i \leq l_j} \frac{1}{w_c}} = \frac{w_c}{l_j}$, where l_j is the number of cities in the county j. Moreover, the sensitivity of the `State` node in the same VGH is $\frac{1}{\sum_{1 \leq j \leq m} \frac{1}{w_c / l_j}} = \frac{w_c}{\sum_{1 \leq j \leq m} l_j} = \frac{w_c}{n}$, where m is the number of counties in the state and $n = \sum_{1 \leq j \leq m} l_j$ is the number of cities in the state. Due to the randomness nature of the sensitivity weights, each of the obtained result points is averaged over 5 runs.

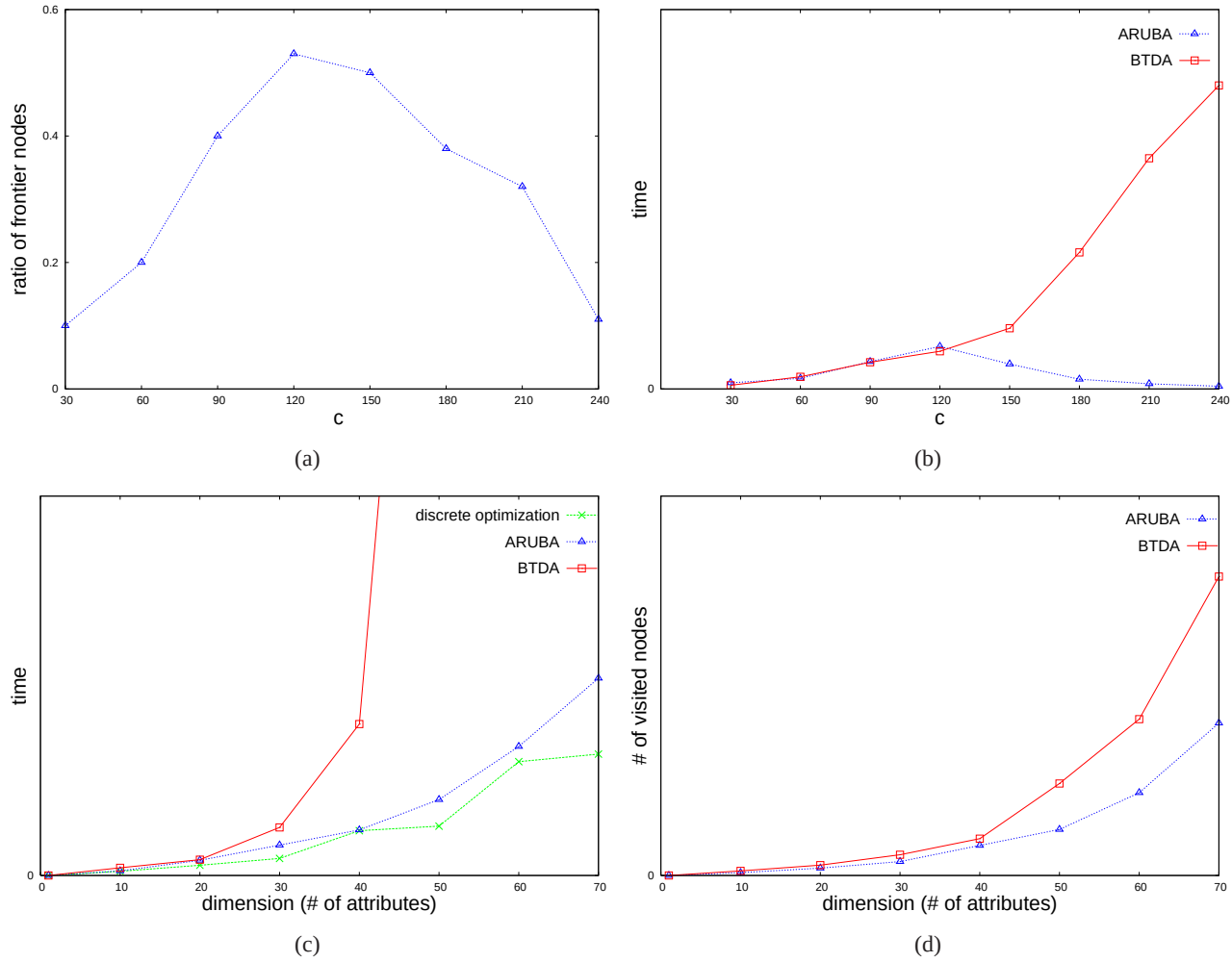


Fig. 7. Algorithms behavior with increasing utility threshold c (subfigures (a) and (b)) and with increasing dimension (subfigures (c) and (d))

database	Wal-Mart
table	item description
# records	400,000
# attributes	[1,70]
heights h_i^s of VGHs	[1,5]
$r_1(\boldsymbol{a})$	$\frac{\Phi(\boldsymbol{a})}{\lvert\rho(\boldsymbol{a},\theta)\rvert}$
$r_2(\boldsymbol{a})$	$\frac{1}{\lvert\rho(\boldsymbol{a},\theta)\rvert}$
$r_3(\boldsymbol{a})$	$-\Phi(\boldsymbol{a}) \log \lvert\rho(\boldsymbol{a},\theta)\rvert$
$\Phi(\boldsymbol{a})$	$\sum_{i\,:\,a_i \neq \perp} w_i$
w_i for leaves	randomly generated weights
w_i for non-leaves	$\frac{1}{\sum_{1\leq i\leq l} \frac{1}{w_i}}$
$u(\boldsymbol{a})$	$\sum_{i=1}^{k} depth(a_i)$
c	[1,250], $\frac{1}{2}\sum_{i=1}^{k} h_i$ (when fixed)

Fig. 8. A summary of experiment parameters and functions

We use a simplified utility function $u(\boldsymbol{a})$ to capture the information benefit of releasing a record $\boldsymbol{a}$: $u(\boldsymbol{a}) = \sum_{i=1}^{k} depth(a_i)$. For each record $\boldsymbol{a}$, the minimum risk is obtained subject to the constraint $u(\boldsymbol{a}) \geq c$. Fig. 8 summarizes the parameters and function settings used throughout the experimental evaluation.

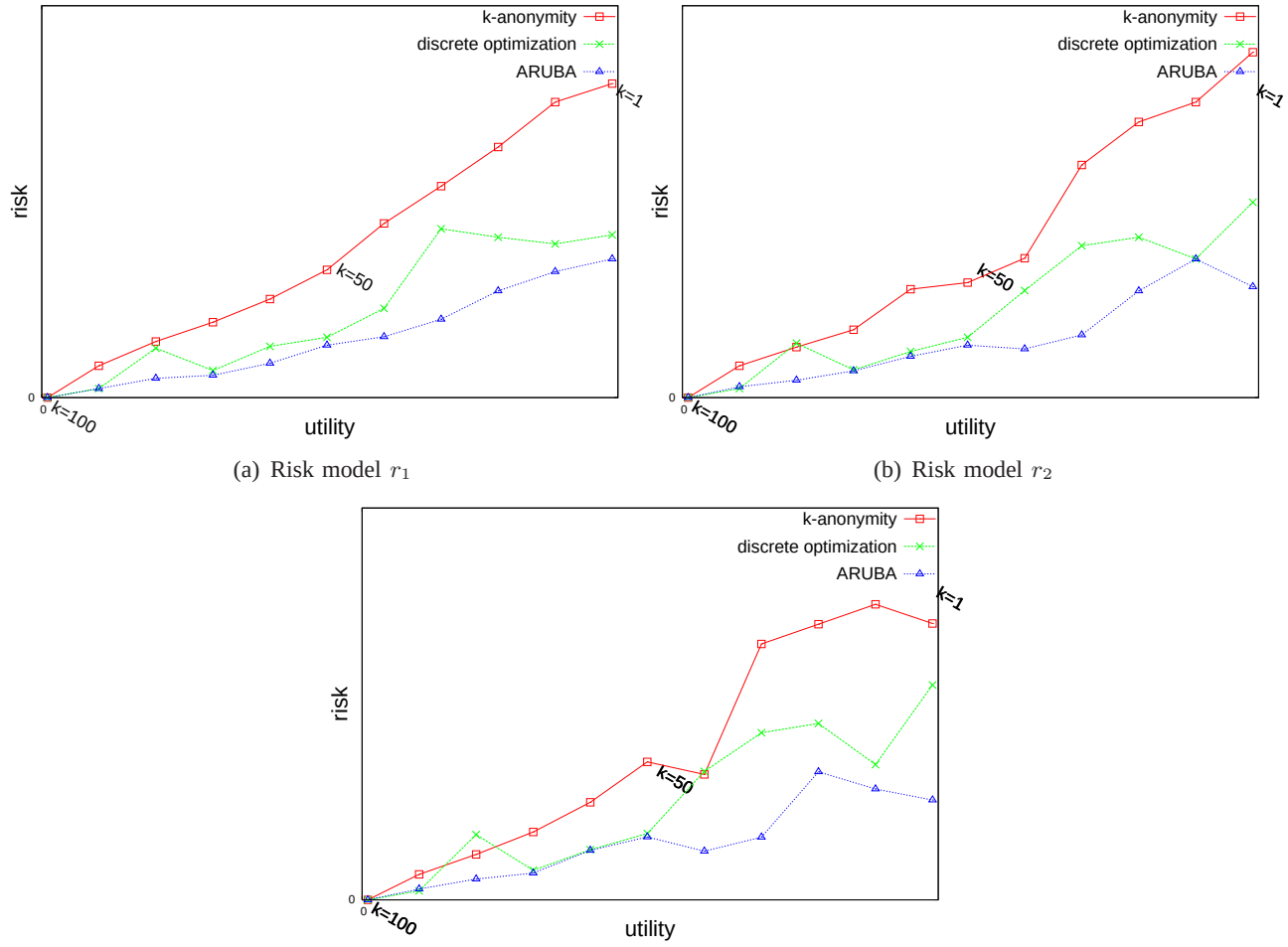
The impact of varying the utility threshold c while maintaining a full set of attributes is shown in Fig. 7(a) and Fig. 7(b). The percentage of frontier nodes is plotted as c varies from 30 to 240 in Fig. 7(a). It is evident that the number of frontier nodes is not directly proportional to c. When c is large, all lattice nodes tend to be infeasible leading to zero or a small number of frontier nodes. Likewise, when c is too small, all lattice nodes tend to be feasible leading to zero or small number of frontier nodes (refer to the definition of frontier nodes in Section V). In Fig. 7(b), the running time for both algorithms is measured at various values of c. It shows that ARUBA almost always outperforms BTDA especially for large values of c. Intuitively, as c increases towards the high extreme, the number of frontier nodes rapidly decreases (as shown in Fig. 7(a)) and, consequently, ARUBA converges very quickly. On the other hand, for large values for c more lattice nodes will be visited by BTDA before the optimum is reached. Therefore, the performance of BTDA deteriorates as c increases. Interestingly, for small values of c, there is no significant difference between ARUBA and BTDA. The reason is that the number of frontier nodes decreases rapidly as c approaches the lower extreme as well and ARUBA tends to perform well.

Throughout the following set of experiments, we fix the utility threshold c at a certain level which is intentionally chosen to be midway through the lattice (i.e., $c = \frac{1}{2}\sum_{i=1}^{k} h_i$) where ARUBA tends to perform the worst. We implement a heuristic discrete optimization algorithm, Branch and Bound [5], to obtain the heuristic optimum disclosure rule. Fig. 7(c) and Fig. 7(d) show that ARUBA outperforms BTDA in terms of both execution time and number of lattice visited nodes. Moreover, ARUBA exhibits a comparable performance with the discrete optimization algorithm in terms of time as shown in Fig. 7(c) but with a lower risk as shown in Fig. 9.

We compare the risk and utility associated with a disclosed table based on our proposed algorithm and arbitrary k-anonymity rules for k from 1 to 100. At each value of k, we generate a set of 10 k-anonymous tables and then compute the average utility associated with these tables using the simplified utility measure mentioned earlier. For each specific utility value c, we run both our proposed algorithm and the discrete optimization algorithm to identify the table that has not only the minimum risk but also a utility greater than or equal to c. We use each of the three risk models when solving these optimization problems. In Fig. 9 we plot the utility and risk of ARUBA (optimally selected disclosure policies), discrete optimization algorithm, and standard k-anonymity rules for different risk models. It is clear that ARUBA consistently outperforms both of the discrete optimization algorithm and standard k-anonymity rules regardless the nature of the model used to compute the risk. It is worth mentioning that a crucial difference between our algorithm and most of the other anonymization algorithms is that we apply the transformations on a record-by-record basis instead of dealing with sets of equivalent records and we capture record similarities by means of the number of consistent records, $|\rho(\boldsymbol{a}, \theta)|$, that is embedded in the risk models.

VII. Related Work

Much of the research carried out on data transformations focused on anonymizing a disclosed table so that every record that belongs to it is made indistinguishable from as many other released records as



(a) Risk model r_1

(b) Risk model r_2

(c) Risk model r_3

Fig. 9. A comparison between our proposed algorithms and k-anonymity

possible [13], [14], [8], [10], [1], [7]. This approach, although may sometimes achieve privacy, does not address the privacy-utility tradeoff.

Samarati et al. [12] introduced the concept of minimal generalization in which k-anonymized tables are generated without distorting data more than needed to achieve k-anonymity. Such approach, although it tries to minimize suppressions and generalizations, does not take into account sensitivity and utility of different attribute values at various levels of the generalization hierarchies.

The tradeoff between privacy and utility is investigated by Rastogi et al. [11]. A data-perturbation-based algorithm is proposed to satisfy both privacy and utility goals. However, they define privacy based on a posterior probability that the released record existed in the original table. This kind of privacy measure does not account for sensitive data nor does it make any attempt to hide the identity of the user to whom data pertains. Moreover, they define the utility as how accurate the results of the *count*() query are. Indeed, this definition does not capture many aspects concerning the usefulness of data.

A top-down specialization algorithm is developed by Fung et al. [3] that iteratively specializes the data by taking into account both data utility and privacy constraints. A genetic algorithm solution for the same problem is proposed by Iyengar [4]. Both approaches consider classification quality as a metric for data utility. However, to preserve classification quality, they measure privacy as how uniquely an individual can

be identified by collapsing every subset of records into one record. The per-record customization nature of our algorithm makes it much more practical than other algorithms in terms of both privacy and utility.

A personalized generalization technique is proposed by Xiao and Tao [17]. Under such approach users define maximum allowable specialization levels for their different attributes. That is, sensitivity of different attribute values are binary (either released or not released). In contrast, our proposed scheme provides users with the ability to specify sensitivity weights for their attribute values.

VIII. Conclusions

In this paper we propose an efficient algorithm to address the tradeoff between data utility and data privacy. Maximizing data usage and minimizing privacy risk are two conflicting goals. Our proposed algorithm (ARUBA) deals with the microdata on a record-by-record basis and identifies the optimal set of transformations that need to be applied in order to minimize the risk and in the meantime keep the utility above a certain acceptable threshold. We use predefined models for data utility and privacy risk throughout different stages of the algorithm. We show that the proposed algorithm is consistently superior in terms of risk when compared with k-anonymity and discrete optimization algorithm without a significant sacrifice in the execution time.

As future work, we plan to elaborate more on the impact of different risk and utility models on the performance of our algorithm. Estimating the dictionary of the attacker and the required set of transformations based on incremental disclosure of information is also a subject of future research. Finally, as an ongoing work, we are working on improving the scalability of the proposed algorithm.

References

[1] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 217–228, Washington, DC, USA, 2005. IEEE Computer Society.

[2] P.-C. Cheng, P. Rohatgi, C. Keser, P. A. Karger, G. M. Wagner, and A. S. Reninger. Fuzzy multi-level security: An experiment on quantified risk-adaptive access control. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 222–230, Washington, DC, USA, 2007. IEEE Computer Society.

[3] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *Proc. of the 21st IEEE International Conference on Data Engineering (ICDE 2005)*, pages 205–216, Tokyo, Japan, April 2005. IEEE Computer Society.

[4] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 279–288, 2002.

[5] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4), 1966.

[6] G. Lebanon, M., Scannapieco, M. R. Fouad, and E. Bertino. Beyond k-anonymity: A decision theoretic framework for assessing privacy risk. In *Privacy in statistical databases, Springer Lecture Notes in Computer Science, volume 4302.*

[7] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *SIGMOD Conference*, pages 49–60, 2005.

[8] T. Li and N. Li. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proc. of ICDE*, 2007.

[9] L. Liu, M. Kantarcioglu, and B. Thuraisingham. The applicability of the perturbation based privacy preserving data mining for real-world data. *Data Knowl. Eng.*, 65(1):5–21, 2008.

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

[11] V. Rastogi, D. Suciu, and S. Hong. The boundary between privacy and utility in data publishing. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 531–542, 2007.

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

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

[14] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International, 1998.

[15] L. Sweeney. Privacy-enhanced linking. *ACM SIGKDD Explorations*, 7(2), 2005.

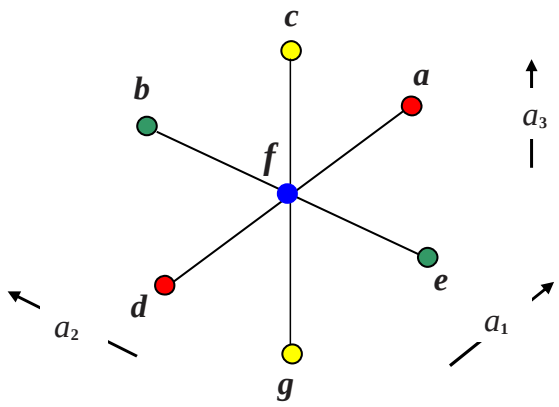


Fig. 10. Proof for Theorem 3 for 3D lattice

[16] K. Wang, P. S. Yu, and S. Chakraborty. Bottom-up generalization: A data mining solution to privacy protection. In *ICDM '04: Proceedings of the Fourth IEEE International Conference on Data Mining*, pages 249–256, Washington, DC, USA, 2004. IEEE Computer Society.

[17] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proc. of SIGMOD*, 2006.

[18] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 785–790, New York, NY, USA, 2006. ACM.

APPENDIX
PROOF OF THEOREM 3 FOR THE CASE OF 3D LATTICE

Proof: Consider the section of the 3D lattice shown in Fig. 10 and assume that $\boldsymbol{f}$ is a frontier node. There are 3 cases:

- All nodes in $IG(\boldsymbol{f}) = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$ are infeasible. Consider the node $\boldsymbol{e}$. If $ig_{a_1}(\boldsymbol{e})$ is infeasible, then $\boldsymbol{e}$ is a frontier node. Otherwise, $ig_{a_1}(\boldsymbol{e})$ a frontier node. The exact same argument applies to nodes $\boldsymbol{d}$ and $\boldsymbol{g}$.
- Exactly two nodes in $IG(\boldsymbol{f})$ are infeasible. Assume, without loss of generality, that these two nodes are $\boldsymbol{a}$ and $\boldsymbol{b}$. Since $\boldsymbol{c}$ is feasible and $ig_{a_1}(\boldsymbol{c}) = ig_{a_3}(\boldsymbol{a})$ is infeasible (since $\boldsymbol{a}$ is infeasible), then $\boldsymbol{c}$ is a frontier node. Now, consider the node $\boldsymbol{e}$. If $ig_{a_1}(\boldsymbol{e})$ is infeasible, then $\boldsymbol{e}$ is a frontier node. Otherwise, $ig_{a_1}(\boldsymbol{e})$ a frontier node. The exact same argument applies to node $\boldsymbol{d}$.
- Exactly one node in $IG(\boldsymbol{f})$ are infeasible. Assume, without loss of generality, that this node is $\boldsymbol{c}$. Since $\boldsymbol{a}$ is feasible and $ig_{a_1}(\boldsymbol{c}) = ig_{a_3}(\boldsymbol{a})$ is infeasible (since $\boldsymbol{c}$ is infeasible), then $\boldsymbol{a}$ is a frontier node. Likewise, it can be proved that $\boldsymbol{b}$ is a frontier node. Now, if $ig_{a_3}(\boldsymbol{e})$ is infeasible, then $\boldsymbol{e}$ is a frontier node. Otherwise, $ig_{a_3}(\boldsymbol{e})$ is a frontier node since $\boldsymbol{c} = ig_{a_2}\big(ig_{a_3}(\boldsymbol{e})\big)$ is infeasible.

In all of the above cases, $\boldsymbol{f}$ has at least 3 neighboring frontier nodes. ■