On the Identification of Property Based Generalizations in Microdata Anonymization

Rinku Dewri, Indrajit Ray, Indrakshi Ray, Darrell Whitley

On the Identification of Property Based Generalizations in Microdata Anonymization

Rinku Dewri, Indrajit Ray, Indrakshi Ray, and Darrell Whitley

Colorado State University, Fort Collins, CO, USA
{rinku, indrajit, iray, whitley}@cs.colostate.edu

Abstract. Majority of the search algorithms in microdata anonymization restrict themselves to a single privacy property and a single criteria to optimize. The solutions obtained are therefore of limited application since adherence to multiple privacy models is required to impede different forms of privacy attacks. Towards this end, we propose the concept of a *property based generalization* (PBG) to capture the non-dominance relationships that appear when multiple objectives are to be met in an anonymization process. We propose an evolutionary algorithm that can identify a representative subset of the set of PBGs for the purpose of decision making.

1 Introduction

Anonymizing data is challenging because re-identifying the values in sanitized attributes is not impossible when other publicly available information or an adversary's background knowledge can be linked with the shared data. Matching shared attributes between different data sources can be made ambiguous by altering the released information to map to more number of individuals represented in the data set. Samarati and Sweeney enforce such mappings in the k*–anonymity* model using *generalization* and *suppression* schemes [1–3].

An unavoidable consequence of performing data anonymization is the loss in information content of the data set. Researchers have therefore looked at different methods to obtain an optimal generalization [1, 3–6] that maximizes the utility of the anonymized data while satisfying a pre-specified privacy property. The adoption of such an optimization framework brings forth pertinent practical issues that have been ignored for long.

First, data utility and respondent privacy are two equally important facets of data publishing. Proper anonymization thus involves weighing the risk of publicly disseminated information against the statistical utility of the content. In such a situation, it is imperative that the data publisher understands the implications of setting a parameter in a privacy model to a particular value. Second, the k–anonymity model is prone to other forms of attacks on privacy. As a result, a multitude of other privacy models have been proposed over time [7–9], quite often followed by newer forms of privacy attacks. The inclusion of multiple models in the anonymization process is desirable since a single comprehensive

model is yet to be developed. The third issue centers around the notion of biased privacy [10]. Consider the $k-$anonymity model where the measure of privacy (the value of k) is given by the minimum size of an equivalence class. Thus, two anonymizations inducing the same value of k will be considered equally good with respect to privacy protection. However, it is quite possible that for one of the anonymizations, a majority of the individual tuples have lesser probabilities of privacy breaches than their counterparts in the other anonymization. Individual privacy levels as depicted by such a model can therefore be misleading – higher for some, minimalistic for others.

In this paper, we propose resolutions to these issues using the notion of *property based generalizations*. First, inclusion of multiple objectives in the anonymization process is captured using *properties* as anonymization objectives. Second, evaluation of a generalization with respect to a privacy property is performed using both worst case and vector based measurements. The overall effectiveness of a generalization is then measured in terms of its achievement and trade-offs in the different properties. The concept of a single optimal solution is therefore discarded and a representative subset of the minimal solution set is sought. Towards this end, our third contribution is in terms of an evolutionary algorithm that can be used to efficiently search the domain generalization lattice to identify such representative solutions.

The remainder of the paper is organized as follows. Section 2 describes some of the related work in k-anonymization. Section 3 presents the preliminary concepts. Property based generalizations are introduced in section 4, followed by a description of the modified dominance operator in section 5. The evolutionary algorithm is presented in section 6. Section 7 discusses some empirical results. Finally, section 8 concludes the paper.

2 Related Work

Several algorithms have been proposed to find effective k-anonymization. The *μ-argus* algorithm is based on the greedy generalization of infrequently occurring combination of quasi-identifiers and suppresses outliers to meet the k-anonymity requirement [5]. The *Datafly* approach uses a heuristic method to first generalize the quasi-identifier containing the most number of distinct values [3]. Sequences of quasi-identifier values occurring less than k times are suppressed.

On the more theoretical side, Sweeney proposes the *MinGen* algorithm [3] that exhaustively examines all potential generalizations to identify the optimal generalization that minimally satisfies the anonymity requirement. However, the approach is impractical even on modest sized data sets. Meyerson and Williams have proposed an approximation algorithm that achieves an anonymization with $O(k \log k)$ of the optimal solution [11].

Samarati proposes an algorithm [1] that identifies all generalizations satisfying k-anonymity. The approach in *Incognito* [12] is also aimed towards finding all generalizations that satisfy k-anonymity for a given value of k.

A genetic algorithm based formulation is proposed by Iyengar to perform k-anonymization [6]. Bayardo and Agrawal propose a complete search method that iteratively constructs less generalized solutions starting from a completely generalized data set [4]. The idea of a *solution cut* is presented by Fung et al. in their approach to top down specialization [13]. LeFevre et al. extend the notion of generalization on attributes to generalization on tuples in the data set [14]. Dewri et al. [15] explore privacy and utility trade-offs using multi-objective optimization formulations involving an average case privacy measure. Huang and Du also explore multi-objective optimization in the problem of optimizing randomized response schemes for privacy protection [16].

3 Data Anonymization

A data set of size $\mathcal{N}$ is conceptually arranged as a table of rows (or *tuples*) and columns (or *attributes*). Each attribute denotes a semantic category of information that is a set of possible values. Attributes are unique within a table. Each row is a tuple of s values $\langle v_1, \ldots, v_s \rangle$, s being the number of attributes in the data set, such that the value v_j is in the domain of the j^{th} attribute A_j, for $j = 1, \ldots, s$. The domain of attribute A_j is denoted by the singleton sets $A_j = \{a_{j1}\}, \ldots, \{a_{j|A_j|}\}$ where $|A_j|$ is the size of the domain of the attribute.

A *generalization* of attribute A_j is a union of its domain into supersets. Hence the generalized domain of A_j can be written as $H_j^1 = A_{j1}, \ldots, A_{jm}$ such that $\bigcup_i A_{ji} = \cup A_j$ and $A_{jp} \cap A_{jq} = \phi$ for $p \neq q$. We then say H_j^1 is a generalized domain of A_j, denoted as $H_j^1 <_G A_j$. The domain H_j^1 can be further generalized in a similar manner to the domain H_j^2. Generalization of an attribute's domain in this manner gives rise to a *domain generalization hierarchy* (DGH) $H_j^{N_j} <_G \ldots <_G H_J^1 <_G H_j^0$, where $H_j^0 = A_j$. N_j is called the *length* of the attribute's DGH. The DGH is a specification of how an attribute's values can be combined progressively to bigger sets. H_j^0 is a full specialization of attribute A_j, meaning that no two values belong to a single set. The other extreme of this is a full generalization $H_j^{N_j}$ where all values of the attribute belong to a single set. The *generalization level* of the attribute is signified by an integer between 0 and N_j. A generalization level of 0 signifies that all values are distinguishable from each other, while a level of N_j signifies that no two values can be distinguished from each other.

A *domain generalization lattice* is a graph with $\prod_i (N_i + 1)$ nodes. Every node $(n_1, \ldots, n_s); 0 \leq n_i \leq N_i$ is a vector of s dimensions where the i^{th} element n_i specifies the generalization level for attribute A_i. An edge exists between two nodes $(n_1, \ldots, n_s)$ and $(m_1, \ldots, m_s)$ if and only if $\sum_i |n_i - m_i| = 1$.

Given a DGH for each quasi-identifier in the data set, a tuple is said to be in an *anonymized* form when a generalization is applied on the attribute values. The anonymized form is represented as follows. Let us assume a tuple $\langle v_1, \ldots, v_s \rangle$ in the data set. Let $(n_1, \ldots, n_s); 0 \leq n_i \leq N_i$ be the vector representing the generalization level for each attribute; n_i is the level to use in the DGH for

attribute A_i. To map the value v_1 to its generalized form we replace it by the index of the set to which it belongs in the generalized domain at level n_1. For example, if $H_1^{n_1} = A_{11}, \ldots, A_{1m}$ and $v_1 \in A_{1p_1}$, then v_1 is replaced by p_1. After performing similar operations for the other attribute values, the tuple is anonymized to the form $\langle p_1, \ldots, p_s \rangle$, p_i being the set index for value v_i in $H_i^{n_i}$. Transforming all tuples in the data set in this manner results in an anonymized data set.

The anonymized tuples of a data set can then be grouped together into equivalence classes. Two anonymized tuples $\langle p_1, \ldots, p_s \rangle$ and $\langle q_1, \ldots, q_s \rangle$ belong to the same equivalence class if $p_i = q_i; 1 \leq i \leq s$. The k-anonymity property requires that every such equivalence class should be of size at least k.

Attributes can be further divided into *sensitive* and *non-sensitive* ones. For example, values in the "Disease" attribute of a medical history data set is not sensitive in itself, but is considered so if a certain disease is linked to a certain patient. The *ℓ-diversity* property requires that every equivalence class resulting from anonymizing the quasi-identifiers should contain at least ℓ "well-represented" values for a sensitive attribute [8]. The property can be instantiated in different forms depending on the meaning of "well-represented". The instantiation we use here is called *distinct* ℓ-diversity. Distinct ℓ-diversity states that the number of distinct values for a sensitive attribute is at least ℓ in every equivalence class.

4 Property Based Generalization

Multiple objectives to meet during data anonymization are captured in the form of properties [10]. Formally, a property is defined as follows.

Definition 1. Property. *A property is a function $\mathcal{P}$ that maps a table T to a vector of size equal to the number of tuples in the table. The vector is called a property vector and denoted by $\mathcal{P}(\mathsf{T})$.*

A property refers to a privacy, utility or any other measurable feature of a tuple. It signifies the grounds under which a comparison is made between two nodes in the lattice. For example, applying the generalization levels corresponding to a node results in multiple equivalence classes. If we pick our property to be the *"size of the equivalence class to which a tuple belongs,"* then each tuple will have an associated integer. This results in a property vector $\mathcal{P}_{equiv}(\mathsf{T}) = (k_1, k_2, \ldots, k_{\mathcal{N}})$ for a data set of size $\mathcal{N}$, where k_i is the equivalence class size of the i^{th} tuple. A property is therefore a vector based measurement. The motivation behind using such vector based measurements is two fold. First, it fits the conventional "worst case" method of measuring privacy. Second, it allows us to determine the efficiency of a node with respect to the distribution of privacy levels across the data set. These two methods of assessing a node are jointly represented through the use of *quality index functions*.

4.1 Quality Index functions

Comparison between generalizations with respect to a single property can be done by defining an ordering operation on the co-domain of the property. The ordering operator is a user-defined method of evaluating the superiority of a property vector. Typically, such operators are functions defined on the values of the property vectors.

Definition 2. Quality Index. *Let $\mathcal{T}$ be the collection of all possible generalized versions of a table T. Given a property $\mathcal{P}$, a quality index $\mathcal{I}_{\mathcal{P}}$ is a function $\mathcal{I}_{\mathcal{P}} : \mathcal{T} \times \mathcal{T} \rightarrow \mathbb{R}$ which assigns an ordered pair of two tables $\mathsf{T}_l, \mathsf{T}_m \in \mathcal{T}$ a real value $\mathcal{I}_{\mathcal{P}}(\mathsf{T}_l, \mathsf{T}_m)$.*

Quality index functions map a pair of nodes to the set of real numbers. The underlying idea is to quantify quality differences between generalizations by applying common metrics. The value $\mathcal{I}_{\mathcal{P}}(\mathsf{T}_l, \mathsf{T}_m)$ signifies the quality of table T_l relative to table T_m and with respect to the property $\mathcal{P}$. We would therefore say that T_l is preferable over T_m with respect to $\mathcal{P}$ if $\mathcal{I}_{\mathcal{P}}(\mathsf{T}_l, \mathsf{T}_m) > \mathcal{I}_{\mathcal{P}}(\mathsf{T}_m, \mathsf{T}_l)$, assuming that a higher value signifies better achievement of the property. Otherwise, the relationship is $\mathcal{I}_{\mathcal{P}}(\mathsf{T}_l, \mathsf{T}_m) < \mathcal{I}_{\mathcal{P}}(\mathsf{T}_m, \mathsf{T}_l)$.

Worst case measurements A quality index function in the definition requires two tables as input. However, a commonly used method of evaluating a generalization is through *unary* quality index functions. Unary quality indices are functions applied independently on generalizations, i.e. they have a single table as input. For example, the k-anonymity property is a unary quality index based on the equivalence class size property $\mathcal{P}_{equiv}$, given as $\mathcal{I}_{\mathcal{P}_{equiv}}(\mathsf{T}) = \min_i(\mathcal{P}_{equiv}(\mathsf{T}))$. Unary indices only allow the measurement of an aggregate property of a generalization. This prohibits any kind of comparison of individual property values maintained by tuples in a generalization with that maintained in another. Having said so, we do not specify any restriction on the formulation of a quality index function. This is because data utility functions are typically unary in nature, i.e. they are absolute estimates of the information content of the anonymized data. We keep the generic binary formulation since unary functions are a special case of binary functions. In other words, when using worst case privacy models or information loss measurements, we shall assume $\mathcal{I}_{\mathcal{P}}(\mathsf{T}_l, \mathsf{T}_m) \equiv \mathcal{I}_{\mathcal{P}}(\mathsf{T}_l)$.

Measuring quality with spread Privacy of an anonymized table can also be quantified in terms of the differences in individual privacy levels when compared with another anonymized table. Characterizing privacy in this manner captures the changes brought forth in individual privacy levels when moving from one node to another in the generalization lattice. This helps distinguish the privacy preserving efficiency of the two nodes even when both generate the same worst case privacy. We use the *spread* based quality index function in this context. The function is based on the total amount of variation (or spread) present between tuples with respect to a property, given as

$$\mathcal{I}^{spr}_{\mathcal{P}}(\mathsf{T}_l, \mathsf{T}_m) = \sum_{i=1}^{\mathcal{N}} \max(p^l_i - p^m_i, 0)$$

where $(p_1^x, \ldots, p_{\mathcal{N}}^x) = \mathcal{P}(\mathsf{T}_x)$. Thus, T_l better preserves privacy than T_m if $\mathcal{I}_{\mathcal{P}}^{spr}(\mathsf{T}_l, \mathsf{T}_m) > \mathcal{I}_{\mathcal{P}}^{spr}(\mathsf{T}_m, \mathsf{T}_l)$. This characterization follows from the intuition that a generalization better than another should be able to retain higher values of the measured property for more individuals represented in the data set.

The spread quality index function provides a relative characterization of privacy. The function value is only representative of the quality of a node relative to another. However, absolute estimates are more preferable since a node then does not have to be evaluated repeatedly for the same property. Hence, a unary function that can provide the same information as the binary spread function is desired. Formulating such a function is not difficult as highlighted in the following observation.

Observation: *Let $S_{\mathcal{P}}(\mathsf{T}_x)$ denote the sum of the property values in the property vector $\mathcal{P}(\mathsf{T}_x)$. Then $\mathcal{I}_{\mathcal{P}}^{spr}(\mathsf{T}_l, \mathsf{T}_m) > \mathcal{I}_{\mathcal{P}}^{spr}(\mathsf{T}_m, \mathsf{T}_l)$ if and only if $S_{\mathcal{P}}(\mathsf{T}_l) > S_{\mathcal{P}}(\mathsf{T}_m)$.*

Comparing nodes under the light of the spread function can therefore be performed using the sum of the property values, i.e. $\mathcal{I}_{\mathcal{P}}^{spr}(\mathsf{T}_l, \mathsf{T}_m) \equiv \mathcal{I}_{\mathcal{P}}(\mathsf{T}_l) = S_{\mathcal{P}}(\mathsf{T}_l)$. Hence, in the subsequent sections, we shall use the notation $\mathcal{I}_{\mathcal{P}}(\mathsf{T}_l)$ to denote the quality of T_l with respect to $\mathcal{P}$, keeping in mind that $\mathcal{I}_{\mathcal{P}}$ is either a unary function (as used in worst case measurements and loss assessments) or the sum function $S_{\mathcal{P}}$ (sufficient to infer the quality according to the binary spread function).

4.2 Anonymizing with multiple properties

Ideally, any number of properties can be studied on a generalized table. Let us consider an anonymization with respect to the set of properties $\mathbf{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_r\}$. Assessing the quality of a generalization T_l with respect to the properties $\mathbf{P}$ will result in a vector of values $\mathbf{I}_{\mathbf{P}}(\mathsf{T}_l) = [\mathcal{I}_{\mathcal{P}_1}(\mathsf{T}_l), \ldots, \mathcal{I}_{\mathcal{P}_r}(\mathsf{T}_l)]$ where the i^{th} element represents the quality of T_l with respect to the property $\mathcal{P}_i$. A dominance relation $\succeq$ is then specified over the set of such vectors to characterize the efficiency of a generalization, such that $\mathbf{I}_{\mathbf{P}}(\mathsf{T}_l) \succeq \mathbf{I}_{\mathbf{P}}(\mathsf{T}_m)$ if

1. $\forall i = 1 \ldots r : \mathcal{I}_{\mathcal{P}_i}(\mathsf{T}_l) \geq \mathcal{I}_{\mathcal{P}_i}(\mathsf{T}_m)$, and
2. $\exists j \in \{1, \ldots, r\} : \mathcal{I}_{\mathcal{P}_j}(\mathsf{T}_l) > \mathcal{I}_{\mathcal{P}_j}(\mathsf{T}_m)$.

This relation states that for a table to be better than another, it must not have worse quality across all the properties while maintaining better quality with respect to at least one property. Note that the dominance relation is transitive in nature, i.e if $\mathbf{I}_{\mathbf{P}}(\mathsf{T}_1) \succeq \mathbf{I}_{\mathbf{P}}(\mathsf{T}_2)$ and $\mathbf{I}_{\mathbf{P}}(\mathsf{T}_2) \succeq \mathbf{I}_{\mathbf{P}}(\mathsf{T}_3)$, then $\mathbf{I}_{\mathbf{P}}(\mathsf{T}_1) \succeq \mathbf{I}_{\mathbf{P}}(\mathsf{T}_3)$. Using dominance to evaluate a generalization introduces the concept of a *property based generalization* (PBG).

Definition 3. Property Based Generalization. *Let $\mathcal{T}$ be the collection of all possible generalized versions of a table T of size $\mathcal{N}$. Given the properties $\mathbf{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_r\}$ and quality index functions $\mathbf{I} : \mathcal{I}_{\mathcal{P}_1}, \ldots, \mathcal{I}_{\mathcal{P}_r}$ (not necessarily unique), $\mathsf{T}_l \in \mathcal{T}$ is a property based generalization of $\mathsf{T}_m \in \mathcal{T}$ with respect to $\mathbf{P}$, denoted as $\mathsf{T}_l \vdash_{\mathbf{P}} \mathsf{T}_m$, if and only if $\mathbf{I}_{\mathbf{P}}(\mathsf{T}_l) \succeq \mathbf{I}_{\mathbf{P}}(\mathsf{T}_m)$.*

The following observations summarize the literary meaning of property based generalizations.

- We consider T_l to be *better* than T_m if and only if $\mathsf{T}_l \vdash_{\mathbf{P}} \mathsf{T}_m$. Equivalently, T_m is *worse* than T_l.
- T_l and T_m are considered *incomparable* (or mutually non-dominated) if and only if $\mathsf{T}_l \nvdash_{\mathbf{P}} \mathsf{T}_m$, $\mathsf{T}_m \nvdash_{\mathbf{P}} \mathsf{T}_l$ and $\mathsf{T}_l \neq \mathsf{T}_m$.

Incomparable generalizations signify trade-offs across certain properties. Therefore, it is our objective to identify such generalizations for reporting. In addition, the chosen generalizations must also be minimal. Minimal property based generalizations are analogous to Pareto-optimal solutions in a multi-objective optimization problem.

Definition 4. Minimal Property Based Generalization. *Given a collection $\mathcal{T}$ of generalized versions of a table T and the properties $\mathbf{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_r\}$, $\mathsf{T}_w \in \mathcal{T}$ is a minimal property based generalization of $\mathcal{T}$ if $\nexists \mathsf{T}_m \in \mathcal{T} : \mathsf{T}_m \vdash_{\mathbf{P}} \mathsf{T}_w$.*

5 Representative PBGs

One drawback of using the dominance relation $\succeq$ is the inability to control the number of minimal PBGs to report during the search process. We assume here a search process with a finite memory, called the *archive*, to store minimal PBGs. The search process iteratively tries to converge to the set of minimal PBGs. A *generator* component is responsible for creating a new candidate generalization, preferably using the current set of generalizations in the archive. An *updator* component performs a comparison of the candidate generalization with those maintained in the archive and removes all generalizations which cannot be minimal PBGs. The purpose behind maintaining such an archive is to guide the search process towards better regions of the search space, and at the same time maintain a list of the best solutions found so far.

The issue to address is the size of the archive. With no limitation on the size, it may become impossible to store additional prospective generalizations owing to restrictions on physical memory. The primary criteria to fulfill is that the archive maintain generalizations that are minimal PBGs and at the same time have enough diversity to represent the trade-off behavior across the multiple properties.

Let $\mathcal{M}$ denote the set of all minimal PBGs corresponding to a given data set. The objective is to obtain a polynomially bounded sized subset of $\mathcal{M}$. Let $(\epsilon_1, \ldots, \epsilon_r); \epsilon_i > 0$ denote a *discretization vector*, r being the number of properties considered. The quality index space $\mathbb{R}^r$ is then discretized by placing a hypergrid with the co-ordinates $0, \epsilon_i, 2\epsilon_i, \ldots$ along each of the r dimensions. This divides the space into boxes with side lengths same as the discretization vector. Assuming the quality index functions are bounded on both side, i.e. $0 < \mathcal{I}_{\mathcal{P}_i}(\mathsf{T}) \leq K_i$, the box of a generalization T_l is given by the vector

$$\mathcal{B}(\mathsf{T}_l) = \left[\left\lfloor \frac{\mathcal{I}_{\mathcal{P}_1}(\mathsf{T}_l)}{\epsilon_1} \right\rfloor, \ldots, \left\lfloor \frac{\mathcal{I}_{\mathcal{P}_r}(\mathsf{T}_l)}{\epsilon_r} \right\rfloor \right].$$

Algorithm 1 Updator using $\succeq_{box}$

Input: Archive $\mathcal{A}$, candidate generalization T
Output: Updated archive $\mathcal{A}$

1. If ($\mathcal{A} = \phi$) then $\mathcal{A} \leftarrow \{\mathsf{T}\}$; goto step 7
2. Let $\mathcal{S}_{dominate} = \{\mathsf{T}' \in \mathcal{A} | \mathbf{I_P}(\mathsf{T}) \succeq_{box} \mathbf{I_P}(\mathsf{T}')\}$
3. $\mathcal{A} \leftarrow \mathcal{A} - \mathcal{S}_{dominate}$
4. Let $\mathcal{S}_{dominated} = \{\mathsf{T}' \in \mathcal{A} | \mathbf{I_P}(\mathsf{T}') \succeq_{box} \mathbf{I_P}(\mathsf{T})\}$
5. Let $\mathcal{S}_{box} = \{\mathsf{T}' \in \mathcal{A} | \mathcal{B}(\mathsf{T}) = \mathcal{B}(\mathsf{T}')\}$
6. If ($\mathcal{S}_{dominated} = \phi$ and $\mathcal{S}_{box} = \phi$) then $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathsf{T}\}$
7. Return $\mathcal{A}$

A modified dominance relation, called *box-dominance* and denoted by $\succeq_{box}$, is then formulated as

$$\mathbf{I_P}(\mathsf{T}_l) \succeq_{box} \mathbf{I_P}(\mathsf{T}_m) \iff \begin{cases} \mathcal{B}(\mathsf{T}_l) \succeq \mathcal{B}(\mathsf{T}_m) & \text{, if } \mathcal{B}(\mathsf{T}_l) \neq \mathcal{B}(\mathsf{T}_m) \\ \mathbf{I_P}(\mathsf{T}_l) \succeq \mathbf{I_P}(\mathsf{T}_m) & \text{, otherwise} \end{cases}.$$

The box-dominance relation first places the quality index value vectors ($\mathbf{I_P}(\mathsf{T}_l)$ and $\mathbf{I_P}(\mathsf{T}_m)$) in their boxes. If the vectors are on different boxes, then T_l cannot be a PBG of T_m if the box of T_l does not dominate the box of T_m. Otherwise, for the case when the boxes are same, the dominance is checked on the quality index values. Further, every box is allowed to hold only one generalization. Choice between two incomparable generalizations belonging to the same box is made arbitrarily.

Non-dominated boxes signify regions where a minimal PBG exists. By allowing the existence of a single generalization per non-dominated box, the modified dominance relationship maintains a representative subset of the minimal PBGs. The discretization vector determines the size of the boxes and hence impacts the size of the representative subset. If quality index values are in the integer domain, then using a discretization vector of all ones implies using the un-modified dominance relation.

An updator using $\succeq_{box}$: Algorithm 1 outlines an updator algorithm using box-dominance. The algorithm starts with an empty archive $\mathcal{A}$. The first candidate generalization from the generator is therefore automatically inserted into the archive. For subsequent candidates, use of $\succeq_{box}$ effectuates a two level dominance check as explained earlier. First, all generalizations for which the candidate T is a PBG are removed from the archive (Steps 2 and 3). Next, two sets are computed – (i) $\mathcal{S}_{dominated}$ as the set of all generalizations which are PBGs of T, and (ii) $\mathcal{S}_{box}$ as the set of all generalizations whose boxes are same as that of T. The candidate T should not be inserted into the archive if the set $\mathcal{S}_{dominated}$ is non-empty, i.e. there exists a generalization in the archive which is a PBG of T. Further, if $\mathcal{S}_{box}$ is not empty then inclusion of T in the archive will result in the presence of two different generalizations that are positioned in the same box. Step 6 checks for these two conditions, thereby guaranteeing that only non-dominated boxes contain a solution and only one solution is contained in a non-dominated box.

Theorem 1. *Let $\mathcal{M}_g$ denote the set of all generalizations produced by a generator until iteration t and $\mathcal{M}_g^*$ denote the set of minimal PBGs of $\mathcal{M}_g$. Then the archive $\mathcal{A}$ as maintained by Algorithm 1 contains only minimal PBGs of $\mathcal{M}_g$, i.e. $\mathcal{A} \subseteq \mathcal{M}_g^*$.*

Proof. We assume that Algorithm 1 is incorrect, implying $\mathcal{A} \nsubseteq \mathcal{M}_g^*$. Therefore there exists $\mathsf{T}_s \in \mathcal{A}$ generated at iteration s such that $\mathsf{T}_s \notin \mathcal{M}_g^*$.

If $\mathsf{T}_s \notin \mathcal{M}_g^*$ then there exists $\mathsf{T}_q \in \mathcal{M}_g$ discovered at iteration $q \neq s$ such that $\mathbf{I_P}(\mathsf{T}_q) \succeq \mathbf{I_P}(\mathsf{T}_s)$. Also, either $\mathcal{B}(\mathsf{T}_q) = \mathcal{B}(\mathsf{T}_s)$ or $\mathcal{B}(\mathsf{T}_q) \succeq \mathcal{B}(\mathsf{T}_s)$. We can merge these cases and say $\mathbf{I_P}(\mathsf{T}_q) \succeq_{box} \mathbf{I_P}(\mathsf{T}_s)$.

Case (i) $q < s$: If T_q is present in $\mathcal{A}$ at iteration s then T_s will not be included in the archive since $\mathcal{S}_{dominated}$ for T_s contains at least T_q. If T_q is not present in $\mathcal{A}$ at iteration s then it must have been removed by a generalization T_r in $\mathcal{A}$ such that $\mathbf{I_P}(\mathsf{T}_r) \succeq_{box} \mathbf{I_P}(\mathsf{T}_q)$. We therefore have $\mathbf{I_P}(\mathsf{T}_r) \succeq \mathbf{I_P}(\mathsf{T}_q)$ or $\mathcal{B}(\mathsf{T}_r) \succeq \mathcal{B}(\mathsf{T}_q)$. Using the transitivity of the $\succeq$ relation, we have $\mathbf{I_P}(\mathsf{T}_r) \succeq \mathbf{I_P}(\mathsf{T}_s)$ or $\mathcal{B}(\mathsf{T}_r) \succeq \mathcal{B}(\mathsf{T}_s)$, which implies $\mathbf{I_P}(\mathsf{T}_r) \succeq_{box} \mathbf{I_P}(\mathsf{T}_s)$. Hence in this case as well $\mathcal{S}_{dominated} \neq \phi$ for T_s. Note that T_r itself might have got removed from the archive between iteration r and iteration s. However, owing to the transitivity, the generalization which removes it will instead appear in $\mathcal{S}_{dominated}$ for T_s. Hence T_s will never appear in $\mathcal{A}$, i.e. $\mathsf{T}_s \notin \mathcal{A}$, which is a contradiction.

Case (ii) $q > s$: In this case, if T_s exists in $\mathcal{A}$ at iteration q then it would be removed from the archive as it belongs to the set $\mathcal{S}_{dominate}$ of T_q. Further, if T_q gets removed and T_s gets re-generated at a later iteration, the transitivity property would assure that T_s does not get re-inserted into the archive. Thus, $\mathsf{T}_s \notin \mathcal{A}$ which is again a contradiction.

Therefore, T_s can never be a member of the archive at iteration t if it is not a minimal PBG. We can therefore say Algorithm 1 is correct and the archive $\mathcal{A}$ contains only minimal PBGs of $\mathcal{M}_g$. □

Theorem 2. *The archive $\mathcal{A}$ as maintained by Algorithm 1 is of bounded size, given as $|\mathcal{A}| \leq \prod_{i=1}^{r-1} b_i$ where b_i is the i^{th} largest element of the vector $(\frac{K_1}{\epsilon_1}, \ldots, \frac{K_r}{\epsilon_r})$.*

Proof. Recall that $K_1, \ldots, K_r$ are the upper bounds of the quality index functions for r properties. These values can very well be equal. By using box coordinates at $0, \epsilon_i, 2\epsilon_i, \ldots$ along each dimension i, we have divided the quality index value space into $\prod_{i=1}^{r} \frac{K_i}{\epsilon_i}$ boxes and only one node in each box can be included in $\mathcal{A}$. We now cluster these boxes into groups of b_r boxes, giving us a total of $\prod_{i=1}^{r-1} b_i$ clusters. A cluster is formed by grouping together boxes that have the same co-ordinates in all but one dimension. Note that choosing b_r as the parameter to decide the number of boxes in a cluster gives us the smallest possible cluster size and hence the largest number of clusters. This is required if an upper bound on the archive size is to be computed. Next, in a cluster, the box having the maximum co-ordinate value in the differing dimension will dominate all other boxes in the cluster. Therefore, only such a box will contain a minimal PBG. Each cluster can therefore contribute only one minimal PBG, bounding the archive size to the number of such clusters, i.e. $|\mathcal{A}| \leq \prod_{i=1}^{r-1} b_i$. □

Algorithm 2 PBG-EA

Output: Archive $\mathcal{A}$ of representative minimal PBGs

1. $\mathcal{A} \leftarrow \phi$; $t \leftarrow 0$
2. Initialize population P_t
3. Evaluate P_t
4. Update $\mathcal{A}$ with nodes in P_t
5. Assign fitness to nodes in P_t and $\mathcal{A}$
6. Perform selection in $P_t \cup \mathcal{A}$
7. Generate P_{t+1} by performing recombination on selected nodes
8. Update $\mathcal{A}$ with nodes in P_{t+1}
9. $t \leftarrow t + 1$; Repeat from Step 5 unless t=maximum number of iterations allowed
10. Return $\mathcal{A}$

6 An Evolutionary Generator

An efficient generator is required not only to find new candidate PBGs, but also to minimize the number of node evaluations performed during the search process. The generator evaluates each node that it explores and provides it to the updator. We propose here an evolutionary algorithm for this purpose, henceforth called *PBG-EA*. The algorithm follows the structure described in Algorithm 2. The update method from Algorithm 1 is used iteratively in steps 4 and 8. Specifics of the other steps are described next.

Population initialization A population P_t is a collection of N_{pop} nodes in the lattice and undergoes changes as the algorithm progresses. Recall that every node is a vector of s dimensions where s is the number of quasi-identifiers. The population P_0 is created by randomly selecting nodes in the lattice. The fully generalized and fully specialized nodes are always inserted into this initial population as they are trivially minimal PBGs.

Node evaluation Evaluation of a population means computing the quality index values for each node in the population. We focus on the strategy to handle outliers at this point. Outliers in a data set are uncommon combination of attribute values in a tuple. Enforcing a k-anonymity property in the presence of outliers may lead to excessive generalization in the attributes. The approach applied here is to use an upper bound on the number of suppressed tuples. Let η be the maximum number of tuples that is allowed for suppression and $\mathcal{N}$ be the total number of tuples in the data set. Consider the sets $E_1, \ldots, E_{\mathcal{N}}$ where E_i contains anonymized tuples that are indistinguishable from $i-1$ other tuples. In other words, all tuples in the set E_i are i-anonymous. Note that some E_is may be empty sets. If the anonymized data set is to be made k-anonymous, then all tuples in the sets $E_1, \ldots, E_{k-1}$ must be suppressed. Given the hard limit on suppression, this will be possible only if the number of tuples in the union of these sets is less than or equal to η. The same strategy can be applied in a reverse manner. Tuples in all sets $E_1, \ldots, E_j$ are suppressed such that j is the smallest integer satisfying $\sum_{i=1}^{j+1} |E_i| > \eta$. The data set is then k-anonymous with $k = j + 1$. The number of tuples suppressed is $|E_1 \cup \ldots \cup E_j|$ and can be accounted for in the loss measurement.

Fitness assignment Fitness signifies the potential of a node to be a minimal PBG relative to the current population and archive. The fitness assignment we use is adapted from the one used in the SPEA2 algorithm [17]. Let dom_P be the number of nodes in $P_t \cup \mathcal{A}$ dominated by $P \in P_t \cup \mathcal{A}$. The fitness of a node P is then computed as the sum of the dominance counts of the nodes which dominate P, or $Fitness_P = \sum_{P' \in P_t \cup \mathcal{A} \text{ and } P' \succeq P} dom_{P'}$. All non-dominated generalizations will therefore have a fitness of zero. Hence, lower fitness implies better generalizations.

Selection Nodes are selected for recombination by using a binary tournament strategy in $P_t \cup \mathcal{A}$. Under this strategy, two nodes are randomly chosen from $P_t \cup \mathcal{A}$ and the one with the lower fitness is selected. The process is repeated for N_{pop} times, giving a selected population of size N_{pop}.

Recombination The process of recombination involves the crossover and mutation operators, the resulting nodes from which are used as the next population P_{t+1}. A single point crossover is started by first choosing two nodes (without replacement) from the selected population. Parts of the vectors representing the two nodes are then swapped at a randomly chosen crossover point. The swapping procedure is performed with a probability of p_{cross}; otherwise chosen nodes move unchanged into the next population. Each crossover operation results in two nodes for the next population. Performing the operation on the entire selected population creates N_{pop} nodes for inclusion in P_{t+1}. An intermediate single-step mutation is performed on these nodes — with a probability p_{mut}, each attribute's generalization level is either increased or decreased by one using appropriate rounding so that generalization levels are between zero and the DGH lengths.

7 Performance Analysis

We applied our methodology to the "adult.data" benchmark data set available from the UCI machine learning database. The attributes used in this study along with their DGH lengths are listed in Table 1(a). The total number of nodes in the lattice is 17920. The suppression limit η is set at 1% of the data set size, i.e. $\eta = 301$.

k-anonymity and ℓ-diversity are used as the privacy objectives for experiments using worst case privacy. For experiments with spread based measurements, we consider the two properties $\mathcal{P}_1$: *size of equivalence class of a tuple* and $\mathcal{P}_2$: *count of sensitive attribute value of a tuple in its equivalence class.* Sum of the property values in the respective property vectors are denoted by S_k and S_ℓ respectively in the plots. We use the "Occupation" attribute as the sensitive attribute wherever required.

Information loss estimates are obtained using the general loss metric (GLM) and classification error (CM) [6]. The attribute "Salary Class" is used as the class label while performing experiments with the CM metric. The lattice size in this case is 8960. Solutions reported by PBG-EA are compared with those obtained by an exhaustive search of the entire generalization lattice. Note that

Table 1. (a) Attributes and DGH lengths used from the *adult census* data set. (b) CE and RR values in PBG-EA anonymization with different sets of properties. Values are shown as $\frac{\text{mean}}{\text{variance}}$ from the 20 runs.

(a)

Attribute	No. of values	DGH length
Age	74	6
Work Class	7	3
Education	16	3
Marital Status	7	3
Race	5	1
Gender	2	1
Native Country	41	4
Salary Class	2	1
Occupation	14	*sensitive*

(b)

Objectives	CE	RR
k, GLM	$\frac{3.7\times10^{-4}}{6.5\times10^{-9}}$	$\frac{0.94}{8\times10^{-4}}$
k, ℓ, GLM	$\frac{3.3\times10^{-4}}{1.1\times10^{-7}}$	$\frac{0.93}{1.1\times10^{-3}}$
S_k, GLM	$\frac{5.7\times10^{-4}}{2.5\times10^{-7}}$	$\frac{0.84}{1.7\times10^{-3}}$
S_k, S_ℓ, GLM	$\frac{6.6\times10^{-4}}{2.0\times10^{-7}}$	$\frac{0.83}{1.4\times10^{-3}}$

the number of nodes evaluated in the exhaustive search is equal to the size of the lattice, while that used by PBG-EA is much less.

An instance of PBG-EA is run with a population size $N_{pop} = 25$ and for 100 iterations. Probability of crossover is set at $p_{cross} = 0.8$ and probability of mutation at $p_{mut} = 1/\text{number of quasi-identifiers} = 0.125$. Each experiment is run 20 times to compute the mean and variance of the performance metrics (discussed below). The discretization vector is set to all ones, unless otherwise indicated.

We use two metrics to quantify the efficiency of PBG-EA in terms of its ability to converge to the true minimal PBGs (as found by the exhaustive search) and how well the solutions represent the set of all minimal PBGs.

Let $\mathcal{M}$ be the set of all minimal PBGs for a data set and $\mathcal{M}'$ be the solutions in the archive at the end of the final iteration. Quality index values of all nodes in $\mathcal{M}$ and $\mathcal{M}'$ are normalized by dividing the values by the corresponding maximum in $\mathcal{M}$. The *convergence error* (CE) is then given as $CE = \sum_{M' \in \mathcal{M}'} \min_{M \in \mathcal{M}} [dist\,(\mathbf{I_P}(M), \mathbf{I_P}(M'))]$ where *dist* is the euclidean distance between two vectors. A CE value of zero means all solutions in the archive have converged to some minimal PBG.

The *representation ratio* (RR) is the fraction of non-dominated boxes in $\mathcal{M}$ that are occupied by a solution in $\mathcal{M}'$. Given a discretization vector, solutions in $\mathcal{M}$ are assigned their respective boxes and the non-dominated boxes are marked. RR signifies how many of these marked boxes are occupied by a solution in the archive. A value of one signifies that a solution in each non-dominated box exists in the archive.

Figure 1 compares the PBG-EA solutions with those obtained from an exhaustive search for worst case privacy measurements as in k-anonymity and ℓ-diversity. Trade-offs between the k value and the loss are evident from the two property (k-GLM) solutions. The convergence efficiency of PBG-EA is worth mentioning as 94% of all minimal PBGs are discovered by the algorithm (Table

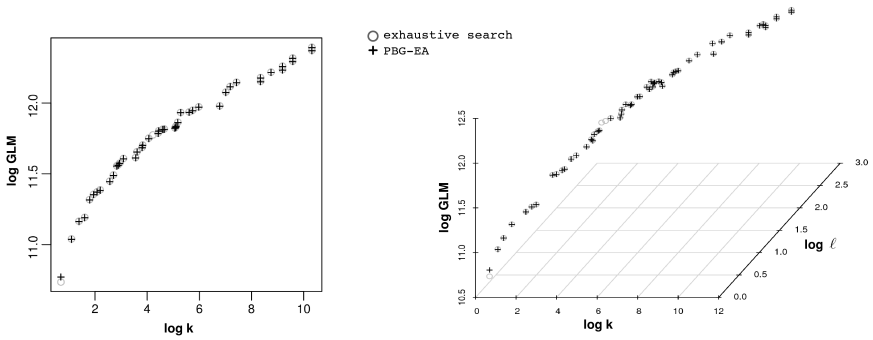

Fig. 1. PBG-EA solutions for the two property (k-GLM) and three property (k-ℓ-GLM) problems.

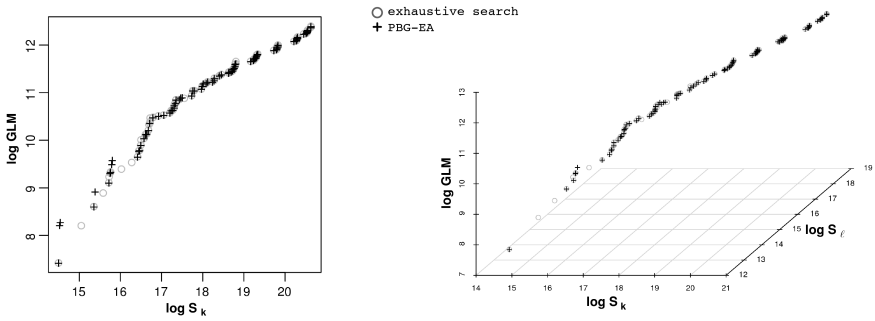

Fig. 2. PBG-EA solutions for the two property (S_k-GLM) and three property (S_k-S_ℓ-GLM) problems using spread based quality.

1(b)). Although the algorithm utilizes random numbers in a number of places, this performance of the algorithm is more or less persistent (low variance across the 20 runs). The trade-offs in the three property (k-ℓ-GLM) seem to be more in terms of loss, rather than between k and ℓ.

Figure 2 shows the PBG-EA solutions when the spread function is used to measure privacy. The RR is slightly lower in this case. Nonetheless, the convergence error is still low. Using the spread function induces a higher number of minimal PBGs whose discovery typically requires more number of iterations. We observe that increasing the number of properties from two to three has very little influence on the RR. A good fitness assignment strategy is required for early determination of solution efficiency. The dominance count based fitness assignment is ideally suited here. Typically, a solution is not worth exploring if it is dominated by a large fraction of the nodes in the lattice. The fitness scheme takes this a step further to also consider the quality of the solutions that dominate it.

PBGs can also be used to find generalizations that are acceptable in terms of more than one loss metric. Figure 3 shows the minimal PBGs obtained when

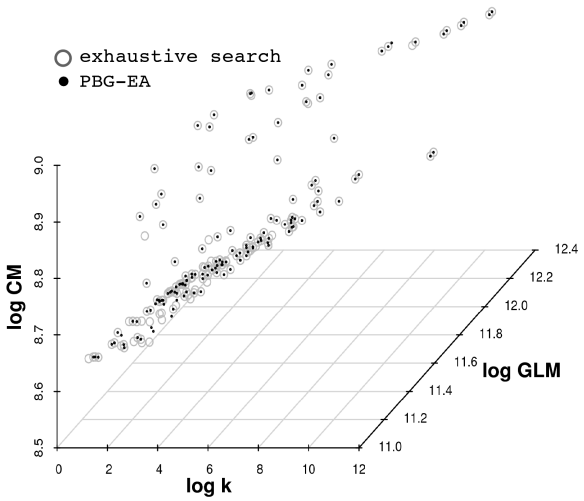

Fig. 3. PBG-EA solutions with a single privacy property (k-anonymity) and two loss metrics (GLM-CM).

the k-anonymity property is evaluated against two different loss metrics, namely GLM and CM. The heavily scattered points across the entire space signify the existence of reciprocal relationships between GLM and CM. Ideally, for a given k value, the convention is to choose the generalization with the lowest GLM. However, the multi-loss metric experiment indicates that choosing a generalization with a comparatively higher GLM can serve the dual purpose of making the anonymized data set also suitable for classification tasks.

Although the set of minimal PBGs for the data set used in the study is not unbounded in size, we experimented with several discretization vectors to demonstrate the efficiency of PBG-EA in finding a representative subset. Table 2(a) shows the performance measures for some vectors. The high representation ratio is indicative of the fact that PBG-EA solutions cover most of the non-dominated boxes generated by the use of the discretization vectors. For two property (k-GLM) anonymization, as higher ϵ values are used for the objectives, the efficiency of PBG-EA improves in terms of RR. However, using more number of properties tend to slightly affect the performance owing mostly to the limited number of iterations.

Efficiency of PBG-EA in converging quickly to a minimal PBG is evaluated by counting the number of unique nodes that are evaluated by it during the search process. Although the evolutionary algorithm can potentially explore 2500 (25×100) distinct nodes in the lattice, a much smaller number is actually evaluated. Table 2(b) lists the average (out of the 20 runs) number of unique node evaluations performed for different problem instances. We consider this low percentage of node evaluations to be a positive indication of the convergence efficiency of PBG-EA. This is particularly promising since the entire set of minimal

Table 2. (a) CE and RR values in PBG-EA anonymization with different discretization vectors. (b) Average number of nodes evaluated in PBG-EA for different sets of properties. Total number of nodes in the first four sets is 17920 and that in the last set is 8960.

(a)

ϵ_k	ϵ_ℓ	ϵ_{GLM}	**CE**	**RR**
5	-	100	$\frac{4.3\times10^{-4}}{2.6\times10^{-7}}$	$\frac{0.95}{1.6\times10^{-3}}$
10	-	1000	$\frac{1.6\times10^{-4}}{8.0\times10^{-8}}$	$\frac{0.98}{8.2\times10^{-4}}$
50	-	10000	$\frac{1.7\times10^{-4}}{1.1\times10^{-8}}$	$\frac{1.0}{0.0}$
5	2	100	$\frac{4.9\times10^{-3}}{2.5\times10^{-7}}$	$\frac{0.92}{1.2\times10^{-3}}$
10	4	1000	$\frac{7.4\times10^{-3}}{9.3\times10^{-7}}$	$\frac{0.92}{7.0\times10^{-4}}$
50	6	10000	$\frac{1.8\times10^{-2}}{8.2\times10^{-6}}$	$\frac{0.88}{1.7\times10^{-3}}$

(b)

Objectives	**Avg. node evaluations**
k, GLM	916 (5.1%)
k, ℓ, GLM	946 (5.3%)
S_k, GLM	1136 (6.3%)
S_k, S_ℓ, GLM	1197 (6.7%)
k, GLM, CM	1073 (11.9%)

PBGs (all one discretization vector) is found by exploring a small 5% of nodes in the lattice. The nodes evaluated is slightly higher for three property problems. This is not surprising since the number of minimal PBGs is also comparatively higher in such cases.

8 Conclusions

In this paper, we propose identifying the basic properties that provide the requisite protection from a privacy breach, and then measuring them for each underlying individual. This generates a property vector for every generalization. Comparison of generalizations with respect to a single property is performed using quality index functions that measure privacy using the variations in individual privacy levels. Optimality in such generalizations is signified by non-dominated generalizations (minimal PBGs) under a dominance relation. A representative subset of solutions is maintained by using a box-dominance operator in an evolutionary algorithm. Application on a benchmark data set shows that the algorithm can quickly discover a diverse set of minimal PBGs with a small number of node evaluations. Observations from our multi-loss experiment suggest that the problem of microdata anonymization to serve multiple usages needs to be explored in more details.

Acknowledgment

This work has been supported in part by a grant from the U.S. Air Force Office of Scientific Research (AFOSR) under contract FA9550-07-1-0042.

References

1. Samarati, P.: Protecting Respondents' Identities in Microdata Release. IEEE Transactions on Knowledge and Data Engineering **13**(6) (2001) 1010–1027

2. Samarati, P., Sweeney, L.: Generalizing Data to Provide Anonymity when Disclosing Information. In: Proceedings of the 17th ACM Symposium on Principles of Database Systems. (1998) 188
3. Sweeney, L.: Achieving k–Anonymity Privacy Protection Using Generalization and Suppression. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems **10**(5) (2002) 571–588
4. Bayardo, R.J., Agrawal, R.: Data Privacy Through Optimal k-Anonymization. In: Proceedings of the 21st International Conference on Data Engineering. (2005) 217–228
5. Hundepool, A., Willenborg, L.: Mu and Tau Argus: Software for Statistical Disclosure Control. In: Proceedings of the Third International Seminar on Statistical Confidentiality. (1996)
6. Iyengar, V.S.: Transforming Data to Satisfy Privacy Constraints. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2002) 279–288
7. Li, N., Li, T., Venkatasubramaniam, S.: t–Closeness: Privacy Beyond k–Anonymity and ℓ–Diversity. In: Proceedings of the 23rd International Conference on Data Engineering. (2007) 106–115
8. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: ℓ–Diversity: Privacy Beyond k–Anonymity. In: Proceedings of the 22nd International Conference on Data Engineering. (2006) 24
9. Truta, T.M., Vinay, B.: Privacy Protection: p-Sensitive k-Anonymity Property. In: Proceedings of the 22nd International Conference on Data Engineering Workshops. (2006) 94
10. Dewri, R., Ray, I., Ray, I., Whitley, D.: On the Comparison of Microdata Disclosure Control Algorithms. In: 12th International Conference on Extending Database Technology. (2009) 240–251
11. Meyerson, A., Williams, R.: On the Complexity of Optimal k-Anonymity. In: Proceedings of the 23rd ACM Symposium on the Principles of Database Systems. (2004) 223–228
12. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient Full-Domain k-Anonymity. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. (2005) 49–60
13. Fung, B.C.M., Wang, K., Yu, P.S.: Top-Down Specialization for Information and Privacy Preservation. In: Proceedings of the 21st International Conference in Data Engineering. (2005) 205–216
14. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian Multidimensional K-Anonymity. In: Proceedings of the 22nd International Conference in Data Engineering. (2006) 25
15. Dewri, R., Ray, I., Ray, I., Whitley, D.: On the Optimal Selection of k in the k-Anonymity Problem. In: Proceedings of the 24th International Conference on Data Engineering. (2008) 1364–1366
16. Huang, Z., Du, W.: OptRR: Optimizing Randomized Response Schemes for Privacy-Preserving Data Mining. In: Proceedings of the 24th International Conference on Data Engineering. (2008) 705–714
17. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems. (2002) 95–100