Generalizing Data to Provide Anonymity when Disclosing Information*

(Extended Abstract)

Pierangela Samarati
Computer Science Laboratory
SRI International
Menlo Park, CA 94025, USA
samarati@csl.sri.com

Latanya Sweeney
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
sweeney@ai.mit.edu

Contact Author:
Latanya Sweeney, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA Telephone: 617–253–3539, Fax: 617–253–3539, Internet: sweeney@ai.mit.edu

*The work of Pierangela Samarati was supported in part by National Science Foundation and by DARPA. The work of Latanya Sweeney was supported in part by the Medical Informatics Training Grant IT15 LM07092 from the National Library of Medicine.

www.manaraa.com

Abstract

The proliferation of information on the Internet and access to fast computers with large storage capacities has increased the volume of information collected and disseminated about individuals. The existence os these other data sources makes it much easier to re-identify individuals whose private information is released in data believed to be anonymous. At the same time, increasing demands are made on organizations to release individualized data rather than aggregate statistical information. Even when explicit identifiers, such as name and phone number, are removed or encrypted when releasing individualized data, other characteristic data, which we term *quasi-identifiers*, can exist which allow the data recipient to re-identify individuals to whom the data refer.

In this paper, we provide a computational disclosure technique for releasing information from a private table such that the identity of any individual to whom the released data refer cannot be definitively recognized. Our approach protects against linking to other data. It is based on the concepts of generalization, by which stored values can be replaced with semantically consistent and truthful but less precise alternatives, and of *k-anonymity*. A table is said to provide k-anonymity when the contained data do not allow the recipient to associate the released information to a set of individuals smaller than k. We introduce the notions of generalized table and of minimal generalization of a table with respect to a k-anonymity requirement. As an optimization problem, the objective is to minimally distort the data while providing adequate protection. We describe an algorithm that, given a table, efficiently computes a preferred minimal generalization to provide anonymity.

www.manaraa.com

1 Introduction

In a globally-network society, there is greater demand by society for individual-specific data, yet the widespread availability of information makes it extremely difficult to release any information about individuals without breaching privacy. The existence of extensive registers of business establishments in the hands of government agencies, trade associations and firms like Dunn and Bradstreet has virtually ruled out the possibility of releasing database information about businesses [6]. Even when released information has no explicit identifiers, such as name and phone number, other characteristic data, such as birth date and ZIP code, often combine uniquely and can be linked to publicly available information to re-identify individuals [11].

Denning showed that no release of data can be proven secure from any arbitrary inference attack [5]. In this paper we provide an effective guard against the specific problem of linking to other data and show that as an optimization problem, we must provide as much information as possible while still protecting privacy and confidentiality by thwarting linking attempts. Recent computational disclosure systems, namely μ-Argus [7] and Datafly [10], use techniques that suppress information and generalize data to prevent linking, but Datafly often distorts the data more than is needed and μ-Argus often fails to provide adequate protection.

In this paper we focus our attention on generalizing information to provide anonymity in released data. This paper has three main contributions: the formal definition of anonymity in the context of protecting disclosed information against linking; the formal definition of minimally generalizing tables to satisfy given anonymity constraints; and an efficient algorithm for determining minimal generalizations that retain preferred specificity in the released data.

We introduce the definition of *quasi-identifiers*, which are attributes that can be exploited for linking, and of *k-anonymity*, which characterizes the degree of protection of data with respect to inference by linking. We show how k-anonymity can be ensured in information release by generalizing data to be disclosed and we introduce the concepts of *generalized* table, *minimal generalization*, and *preferred minimal generalization*. Intuitively, a generalization is minimal if data are not generalized more than necessary to provide k-anonymity. The definition of preferred generalization allows the user to select among possible minimal generalizations those that satisfy particular conditions, such as favoring certain attributes in the generalization process. We provide an algorithm that produces a preferred minimal generalization of a given table. Although the number of possible generalizations of a table is exponential in the number N of tuples to be generalized, the algorithm proves to operate linearly in its best and general cases, degrading to N^2 only in the worst case.

Related work in the area of information protection deals with access control systems [8] and statistical databases [1]. Our work, however, conceptually differs from these proposals by providing a different, yet complementary, approaches to protection. Access control systems address the problem of controlling specific access to data with respect to rules that state whether a piece of data can or cannot be released. This point of view differs from that under consideration in this work because it is not the disclosure of a specific piece of data that must be protected, but rather the ability to link that data to a particular entity (e.g., a patient in a hospital database, a tax-payer in the IRS database, a customer organization in a generic business database). The closest work can be found in the area of statistical databases. However, statistical database techniques [1, 9] address the problem of producing tabular data based on queried information, ensuring that it is not possible for users to infer private data from the produced summary. In our approach instead, we release generalized individual-specific data on which users can produce summaries according to their own needs. Emerging applications and technology such as data mining and the Internet are making tremendous demands for the release of detailed information about individuals such that the identity of the individuals cannot be recognized.

The remainder of this paper is organized as follows. We begin in Section 2 by illustrating the linking problem and providing definitions of quasi-identifier and k-anonymity. Section 3 introduces generalization to provide anonymity and the concept of generalized tables and minimal generalization. Section 4 discusses the problem of efficiently computing a minimal generalization for a table and presents an algorithm for that. Section 5 concludes the paper. The paper also contains an Appendix reporting two supplementary algorithms discussed in the paper, and an example illustrating an application of our approach.

2 The anonymity problem

Data holders often release personal information with all explicit identifiers, such as name, address, phone number, and Social Security number, removed or encrypted in the incorrect belief that privacy is maintained because the

www.manaraa.com

Medical Data released as anonymous

	SSN	Name	Ethnicity	Date Of Birth	Sex	ZIP	Marital Status	Problem
1			black	09/27/64	male	02139	divorced	obesity
2			black	09/30/64	male	02139	divorced	hypertension
3			black	04/18/64	male	02139	married	chest pain
4			black	04/15/64	male	02139	married	chest pain
• 5			black	09/15/64	male	02138	married	shortness of breath
6			caucasian	03/13/63	male	02141	married	hypertension
7			caucasian	03/18/63	male	02141	married	shortness of breath
8			caucasian	09/13/64	female	02138	married	shortness of breath
9			caucasian	09/07/64	female	02138	married	obesity
10			caucasian	05/14/61	female	02138	single	chest pain
11			caucasian	05/08/61	female	02138	single	obesity

Voter List

	Name	Address	City	ZIP	DOB	Sex	Race	Party	.....
	.....	.....	.....	.....	.....	.....	.....	.....	.....
	.....	.....	.....	.....	.....	.....	.....	.....	.....
•	Jim A. Cosby	570, Laurel St.	Cambridge	*02138*	*9/15/64*	*male*	*black*	democrat	.....
	.....	.....	.....	.....	.....	.....	.....	.....	.....

Figure 1: Re-identifying anonymous data by linking to outside data

resulting data look anonymous. Simultaneously, municipalities sell and distribute population registers that include information specific to individuals, such as local census data, voter registration lists, city directories, as well as information from motor vehicle agencies, tax assessors and real estate agencies. In most of these cases, population registers can be used to re-identify individuals by linking or matching it to specific attributes in released "anonymous" data. As an example, the 1997 voting list for Cambridge, Massachusetts contains demographics on 54,805 voters. Birth date alone can uniquely identify the name and address of 12% of the voters; birth date and gender, 29%; birth date and a 5-digit ZIP code, 69%; and 97% (53,033 voters) can be identified when the full postal code and birth date are used [11].

A data holder can often identify attributes in their data that also appear in outside sources, and these attributes are candidates for linking. We term them, *quasi-identifiers*, and it is essentially the combinations of these quasi-identifiers that must be protected.

Definition 2.1 (Quasi-identifier) *Let $T(A_1, \ldots, A_n)$ be a table. A quasi-identifier of T is a set of attributes $\{A_i, \ldots, A_j\} \subseteq \{A_1, \ldots, A_n\}$ whose release must be controlled.*

Figure 1 exemplifies a table of released medical data that has been de-identified by suppressing names and SSNs so as not to disclose the identities of individuals to whom the data refer. An example of a quasi-identifier for this table is represented by attributes {ZIP, date of birth, ethnicity, sex, marital status}. As illustrated in the figure, the first four attributes can be retrieved by accessing a list of voters, which is publically available. other externally available, but are not included in the quasi-identifier since they are assumed to be suppressed (or encrypted) in the release. Quasi-identifiers combine with external knowledge to retrieve the identities of individuals. For instance, in Figure 1, the voter list my have only one `black male` born on `9/15/64` and leaving in the `02138` area. This identifies the corresponding bulleted tuple in the released data as pertaining to `Jim Cosby`. As stated earlier, birthdate alone can re-identify 12% of the voters in Cambridge, MA.

Given a table $T(A_1, \ldots, A_n)$, a subset $\{A_i, \ldots, A_j\}$ of the attributes in T, and a tuple $t \in T$, we use $t[A_i, \ldots, A_j]$ to denote the sequence of the values of the $A_i, \ldots, A_j$ in t. We use $T[A_i, \ldots, A_j]$ to denote the projection, maintaining duplicate tuples, of attributes $A_i, \ldots, A_j$ in T. In this paper we assume the existence of a private table PT, already de-identified and we assume sets of attributes are specified which represent quasi-identifiers. We denote with $\mathsf{QI}_{\mathsf{PT}}$ the set of quasi-identifiers in PT.

One way to state an anonymity constraint involves making sure characteristics and combinations of characteristics found in the data combine to match at least k individuals. To determine how many individuals each released tuple matches requires combining the released data with externally available data. This is obviously an impossible task for the data holder who releases information. Although we can assume the data holder knows what data are part of external knowledge, and therefore what constitutes quasi-identifiers, the specific values of data in external knowledge cannot be assumed. The way to satisfy the k-anonymity requirement is therefore to force such a constraint on the released data.

www.manaraa.com

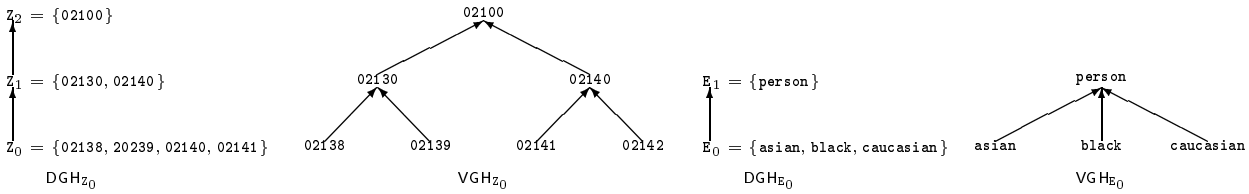

Figure 2: Examples of domain and value generalization hierarchies

Definition 2.2 (k-anonymity) *Let $T(A_1, \ldots, A_n)$ be a table and QI_T be the quasi-identifiers associated with it. T is said to satisfy k-anonymity iff for each quasi-identifier $QI \in \mathsf{QI}_T$,: each sequence of values in $T[QI]$ appears at least with k occurrences in $T[QI]$.*

It can be proved that if each quasi-identifier in the released data satisfies k-anonymity, then the combination of released data to external sources cannot match lower than k individuals. This property holds provided that all attributes in the released table PT which are externally available in combination (i.e., appearing together in an external table or in a possible join of external tables) are defined in a quasi-identifier associated with PT.

3 Generalization to combat linking inference

There are many techniques to achieve k-anonymity, such as changing unusual information to typical values, inserting complementary records, swapping entries, scrambling records and suppressing information [1, 3, 9]. We propose an approach that produces a version of PT such that the k-anonymity requirement is satisfied by re-coding values to make them more general.

In a classical relational database system, domains are used to describe the set of values that attributes assume. For example, there might be a ZIP code domain, a *number* domain and a *string* domain. We extend this notion of a domain to make it easier to describe how to *generalize* the values of an attribute. In the original database, where every value is as specific as possible, every attribute is in the *ground* domain. For example, 02139 is in the *ground* ZIP code domain, $\mathtt{Z}_0$. In order to achieve k-anonymity we can make the ZIP code less informative. We do this by saying there is a more general, less specific, domain that can be used to describe ZIP codes, Z_1, in which the last digit has been replaced by a 0. There is also a mapping from $\mathtt{Z}_0$ to $\mathtt{Z}_1$, such as $\mathtt{02139} \rightarrow \mathtt{02130}$. This mapping between domains is stated by means of a generalization relationship, which represents a *partial order* on the set Dom of domains, and which is required to satisfy the following conditions: *1)* each domain D_i has at most one *direct* generalized domain; and *2)* all maximal elements of Dom are singleton.[1] The definition of this generalization implies the existence, for each domain $D \in \mathsf{Dom}$, of a hierarchy, which we term the **domain generalization hierarchy** DGH_D. Since generalized values can be used in place of more specific ones, it is important that all domains in the hierarchy be compatible. Compatibility can be ensured by using the same storage representation form for all domains in a generalization hierarchy. A *value generalization relationship* is also defined which associates with each value v_i in a domain D_i a *unique* value in domain D_j direct generalization of D_i. Such a relationship implies the existence, for each domain D of a **value generalization hierarchy** VGH_D. Figure 2 illustrates an example of domain and value generalization hierarchies for domain $\mathtt{Z}_0$, representing ZIP codes for Cambridge, MA, and $\mathtt{E}_0$ representing ethnicity.

In the remainder of this paper we will often need to refer to a domain or value generalization hierarchy in terms of the graph representing all and only the *direct* generalization relationships between the elements in it (i.e., implied generalization relationships do not appear as arcs in the graph). We will use the term hierarchy indistinguishably to denote either a partially ordered set or the graph representing the set and all the direct generalization relationships between its elements. We will explicitly refer to the ordered set or to the graph when it is not otherwise clear from context.

[1] The motivation behind condition 2 is to ensure that all values in each domain can be eventually generalized to a single value.

www.manaraa.com

Eth:E_0	ZIP:Z_0
a	38
a	39
a	41
a	42
b	38
b	39
b	41
b	42
c	38
c	39
c	41
c	42

PT

Eth:E_1	ZIP:Z_0
p	38
p	39
p	41
p	42
p	38
p	39
p	41
p	42
p	38
p	39
p	41
p	42

$GT_{[1,0]}$

Eth:E_1	ZIP:Z_1
p	30
p	30
p	40
p	40
p	30
p	30
p	40
p	40
p	30
p	30
p	40
p	40

$GT_{[1,1]}$

Eth:E_0	ZIP:Z_2
a	00
a	00
a	00
a	00
b	00
b	00
b	00
b	00
c	00
c	00
c	00
c	00

$GT_{[0,2]}$

Eth:E_0	ZIP:Z_1
a	30
a	30
a	40
a	40
b	30
b	30
b	40
b	40
c	30
c	30
c	40
c	40

$GT_{[0,1]}$

Figure 3: Examples of generalized tables for PT

3.1 Generalized table and minimal generalization

Given a private table PT, our approach is to provide a table which respects k-anonymity by generalizing the values stored in it. Generalization is effective because substituting attribute values with generalized values maps more values to the same generalized result. This typically decreases the number of distinct tuples and thus increases the number of tuples having the same values. We enforce generalization at the attribute level. Generalizing an attribute means substituting its values with corresponding values from a more general domain. Since attributes can change domain in the generalization process, the simple classical notation D_i for the domain of attribute A_i does not suffice. Instead, we need to explicitly refer to the domain an attribute assumes in a specific table. In the following, $dom(A_z, T)$ denotes the domain of attribute A_z in table T. $D_i = dom(A_z, \mathsf{PT})$ denotes the domain associated with attribute A_i in the definition of the private table PT.

A table $T_j(A_1, \ldots, A_n)$ is said to be a **generalization of a table** $T_i(A_1, \ldots, A_n)$, written $T_i \leq T_j$ iff: *(1)* T_i and T_j have the same number of tuples; *(2)* the domain of each attribute A_z in T_j is equal to or a generalization of the domain of A_z in T_i; and *(3)* each tuple t_i in T_i has a corresponding tuple t_j in T_j (and vice-versa) such that for each attribute A_z, $t_j[A_z]$ is equal to or a generalization of $t_i[A_z]$.

Example 3.1 *Consider the table* PT *illustrated in Figure 3 and the domain and value generalization hierarchies for* E_0 *and* Z_0 *illustrated in Figure 2. The remaining four tables in the figure are examples of generalized tables for* PT. *For the clarity of the example, every table reports, together with each attribute, the domain for the attribute in the table. With respect to k-anonymity:* $GT_{[1,0]}$ *and* $GT_{[0,1]}$ *satisfies k-anonymity for* $k = 1, 2$*;* $GT_{[0,2]}$ *satisfies k-anonymity for* $k = 1, \ldots, 4$*, and* $GT_{[1,1]}$ *satisfies k-anonymity for* $k = 1, \ldots, 6$.

It is easy to see that the number of different generalizations of a table is equal to the number of different combinations of domains that the attributes in the tables can assume. Clearly, not all generalizations are equally satisfactory. A trivial possible generalization, for instance, is the one that generalizes each attribute to the highest possible level of generalization, thus collapsing all tuples in the table to the same list of values. This provides k-anonymity at the price of a strong generalization of the data. Such an extreme generalization is not needed if a less generalized table (i.e., containing more specific values) exists which satisfies k-anonymity. This concept is captured by the following definitions of k-minimal generalization.

Definition 3.1 (k-minimal generalization − wrt a quasi-identifier) *Let T_i and T_j be two tables such that $T_i \leq T_j$. T_j is said to be a* k-minimal *generalization of a table T_i* wrt to a quasi-identifier QI *iff:*

1. *T_j satisfies k-anonymity wrt QI*
2. *$\forall T_z : T_i \leq T_z, T_z \leq T_j$, T_z satisfies k-anonymity wrt $QI \Rightarrow T_z[QI] = T_j[QI]$.*

Intuitively, a table T_j generalization of T_i is k-minimal if it satisfies k-anonymity and there does not exist any generalization of T_i which satisfies k-anonymity and of which T_j is a generalization.

Example 3.2 *Consider table* PT *and its generalizations illustrated in Figure 3. Assume QI=(Eth,ZIP) to be a quasi-identifier. It is easy to see that for $k = 2$ there exist two k-minimal generalizations, which are* $GT_{[1,0]}$ *and*

www.manaraa.com

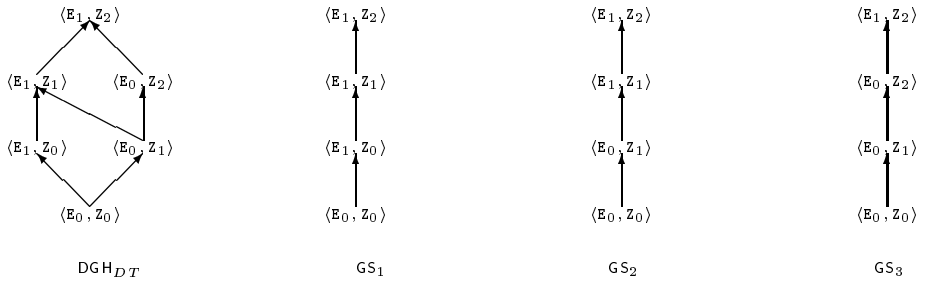

Figure 4: Domain generalization hierarchy DGH_{DT} and strategies for $DT = \langle \mathtt{E}_0, \mathtt{Z}_0 \rangle$

$\mathsf{GT}_{[0,1]}$. *Table* $\mathsf{GT}_{[0,2]}$*, which satisfies the anonymity requirement is not minimal since it is a generalization of* $\mathsf{GT}_{[0,1]}$*. Analogously,* $\mathsf{GT}_{[1,1]}$ *cannot be minimal, being a generalization of both* $\mathsf{GT}_{[1,0]}$ *and* $\mathsf{GT}_{[0,1]}$*. There are also only two k-minimal generalized tables for k=3, which are* $\mathsf{GT}_{[1,0]}$ *and* $\mathsf{GT}_{[0,2]}$*.*

Definition 3.2 (k-minimal generalization) *Let $T_i(A_1, \ldots, A_n)$ and $T_j(A_1, \ldots, A_n)$ be two tables such that $T_i \leq T_j$. T_j is said to be a* k-minimal *generalization of T_i iff:*

1. *T_j is k-minimal wrt every quasi-identifier QI in QI_{T_i}.*
2. *$\forall k = 1, \ldots, n : \nexists QI \in \mathsf{QI}_{T_i}, A_z \in QI \Rightarrow dom(A_z, T_j) = dom(A_z, T_i)$.*

According to Definition 3.2 a table T_j is a k-minimal generalization of table T_i iff it is k-minimal with respect to every quasi-identifier of T_i and attributes not belonging to any quasi-identifier are not generalized. It is trivial to see that a table that satisfies k-anonimity has a *unique* k-minimal generalization, which is itself. It is also easy to see that the necessary and sufficient condition for a table T to satisfy k-anonimity is that the cardinality of the table must be at least k, as stated by the following theorem. The requirement that the maximal elements of Dom be singleton ensures the sufficiency of the condition.

Theorem 3.1 *Let T be a table and k be a natural number. If $|T| \geq k$, then there exists at least a k-minimal generalization for T. If $|T| < k$ there are no k-minimal generalization for T.*

When different minimal generalizations exists preference criteria can be applied to choose a *minimal preferred* generalization. For instance, tables which generalize or do not generalize specific attributes or return the highest number of distinct tuples can be preferred.For instance, for a k-anonymity requirement with k=2, $\mathsf{GT}_{[1,0]}$ and $\mathsf{GT}_{[0,1]}$ are both minimal but $\mathsf{GT}_{[0,1]}$ maybe preferred because it contains a larger number of distinct tuples.

3.2 Generalization hierarchies and strategies

Since quasi-identifiers may be composed of different attributes, it is useful to talk about generalization relationship and hierarchies in terms of tuples composed of elements of Dom or of their values. Given a tuple $DT = \langle D_1, \ldots, D_n \rangle$ such that $D_i \in \mathsf{Dom}$, $i = 1, \ldots, n$. we define the **domain generalization hierarchy** of DT as $\mathsf{DGH}_{DT} = \mathsf{DGH}_{D_1} \times \ldots \times \mathsf{DGH}_{D_n}$, assuming the cartesian product is ordered by imposing coordinatewise order [4]. DGH_{DT} defines a lattice whose minimal element is DT. For instance, Figure 4 illustrates the domain generalization hierarchy $\mathsf{DGH}_{\mathtt{E}_0,\mathtt{Z}_0}$ where the domain generalization hierarchies of $\mathtt{E}_0$ and $\mathtt{Z}_0$ are as illustrated in Figure 2.

The generalization hierarchy of a domain tuple DT defines different ways in which DT can be generalized. In particular each path from DT to the unique maximal element of DGH_{DT} in the graph describing DGH_{DT} defines a possible alternative path that can be followed in the generalization process. We refer to the set of nodes in each of such paths together with the generalization relationships between them as a **generalization strategy** for DGH_{DT}. The different domain generalization strategies for $\mathsf{DGH}_{\mathtt{E}_0,\mathtt{Z}_0}$ above are illustrated in Figure 4. The number of different possible strategies for a domain hierarchy is stated by the following theorem.

www.manaraa.com

Theorem 3.2 *Let $DT = \langle D_1, \ldots, D_n \rangle$ be a tuple such that $D_i \in \mathsf{Dom}, i = 1, \ldots, n$. The number of different strategies for DT is: $\frac{h_{DT}!}{h_1! \ldots h_n!}$, where each h_i is the length of the path from D_i to the top domain in DGH_{D_i} and $h_{DT} = \sum_{i=1}^{n} h_i$.* □

For each strategy a minimal local generalization can be defined as the table satisfying k-anonymity, whose sequence of domains DT' belongs to the strategy such that there are no other tables satisfying k-anonymity whose sequence of domains DT'' is in the strategy and such that $DT'' \leq DT'$. Since a strategy is a total order, the minimal local generalization is always unique. The following theorem states the correspondence between k-minimal generalization and the local minimal generalization with respect to a strategy.

Theorem 3.3 *Let $T(A_1, \ldots, A_n)$ and let $DT = \langle D_1, \ldots, D_n \rangle$ be the tuple where $D_z = dom(A_z, T)$, $z = 1, \ldots, n$, be a table to be generalized. Every k-minimal generalization of T is a local minimal generalization for some strategy of DGH_{DT}.*

The converse is not true, i.e., a local minimal generalization with respect to a strategy may not correspond to a k-minimal generalization. For instance, consider table PT and its generalized tables illustrated in Figure 3, whose minimality has been discussed in Example 3.1. For $k = 3$ the minimal local generalizations are: $\mathsf{GT}_{[1,0]}$ for strategy 1, $\mathsf{GT}_{[1,1]}$ for strategy 2, and $\mathsf{GT}_{[0,2]}$ for strategy 3. However, as we have shown in Example 3.1, $\mathsf{GT}_{[1,1]}$ is not k-minimal for k=3. For $k = 2$ the minimal local generalizations are: $\mathsf{GT}_{[1,0]}$ for strategy 1, and $\mathsf{GT}_{[0,1]}$ for strategies 2 and 3. From Directly from Theorem 3.3, a table has at most as many generalizations as the number of generalization strategies of its domain sequence. The number of k-minimal generalizations can be smaller if the generalized table, locally minimal with respect to a strategy, is a generalization of a table locally minimal to another strategy ($\mathsf{GT}_{[1,1]}$ for $k = 3$ in the example above), or if different strategies have the same local minimal generalization ($\mathsf{GT}_{[0,1]}$ for k=2 in the example above).

4 An optimal algorithm for determining a minimal generalization

In this section we present an efficient way to determine a minimal generalization of a table. Since the k-anonymity property requires the existence of k occurrences for each sequence of values present in the table, though only for those of quasi-identifiers, we can concentrate on the generalization of specific quasi-identifiers within table PT. The generalized table PT is obtained by enforcing generalization on each quasi identifier $QI \in \mathsf{QI}_{\mathsf{PT}}$. The correctness of the combination of the generalizations independently produced for each quasi-identifier is ensured by the fact that the definition of generalized table requires the correspondence of values across whole tuples and by the fact that the quasi-identifiers of a table are disjoint. (This last constraint can be removed if generalization of non disjoint quasi-identifiers be executed serially.) Our generalization algorithm is based on the definition and use of distance vectors, defined in the following subsection.

4.1 Distance vectors

We introduce a distance vector metric with Euclidean properties that measures distances between tuples and tables based on the number of generalization levels required to have the tuples or tables share the same generalized values. We further show that these distance vectors represent generalization strategies.

Definition 4.1 (Distance vector) *Let $T_i(A_1, \ldots, A_n)$ and $T_j(A_1, \ldots, A_n)$ be two tables such that $T_i \leq T_j$. The distance vector of T_i to T_j is the vector $\mathsf{DV}_{i,j} = [d_1, \ldots, d_n]$ where each d_z is the length of the* unique *path between $D = dom(A_z, T_i)$ and $dom(A_z, T_j)$ in the domain generalization hierarchy DGH_D.*

Intuitively the distance vector captures how many generalization steps table T_j is from table T_i for each attribute. To illustrate, consider table PT and its generalized tables illustrated in Figure 3. The distance vectors between PT and its different generalizations are the vectors appearing as subscripts of each table.

The relationship between distance vectors and minimal generalizations, which is the basis of the correctness of our approach, is stated by the following theorem.

Theorem 4.1 *Let T_i and T_j be two tables such that $T_i \leq T_j$ and T_j satisfies k-anonymity. T_j is k-minimal $\Leftrightarrow \nexists T_z, T_z \neq T_j$, such that $T_i \leq T_z$, T_z satisfies k-anonymity, and $\mathsf{DV}_{i,z} \leq \mathsf{DV}_{i,j}$.*

www.manaraa.com

VectorCover(*Outliers*, *All*) /* It returns the set of sorted vectors representing all possible distances between tuples in *Outliers* and tuples in *All*. Uses priority queue *queue* with elements $\langle V$,*tpl-set*$\rangle$. Priority is based on $\mathsf{dist}(v) \star |$*tpl-set*$|$.*/
1. *queue* $\leftarrow \langle [d_1, \ldots, d_n]$, *All*$\rangle$, with $d_1 = \ldots, d_n = 0$
2. *vects* $= \langle [d_1, \ldots, d_n] \rangle$, with $d_1 = \ldots, d_n = 0$ /* sorted list of distance vectors, based on $<$ */
3. **while** *queue*$\neq \emptyset$ **do**
   3.1 Select and remove $\langle v,$ *Neighbors*$\rangle$ from *queue*
   3.2 **if** *Neighbors* $\cap$ *Outliers* $\neq \emptyset$ **then**
      3.2.1 select a tuple s from *Neighbors* $\cap$ *Outliers*
      3.2.2 *New-pairs* $\leftarrow \emptyset$ /* keep tracks of new pairs to be added to *queue* */
      3.2.3 **for each** tuple $t \in$ *Neighbors* **do**
         3.2.3.1 $V_{s,t} \leftarrow$ compute distance between s and t
         3.2.3.2 **if** $\exists \langle V,$ *tuples*$\rangle \in$ *New-pairs* such that $V_{s,t} \leq V, \mathsf{dist}(V_{s,t}) > 1$ **then** *tuples* $\leftarrow$ *tuples* $\cup \{t\}$.
         3.2.3.3 **else if** $\exists \langle V,$ *tuples*$\rangle \in$ *New-pairs* such that $V < V_{s,t}, \mathsf{dist}(V_{s,t}) > 1$ **then** $V \leftarrow V_{s,t}$, *tuples* $\leftarrow$ *tuples* $\cup \{t\}$.
         3.2.3.4 **else if** $\mathsf{dist}(V_{s,t}) > 1$ **then** *New-pairs* $\leftarrow$ *New-pairs* $\cup \{\langle V_{s,t}, \{t\}\rangle\}$
      3.2.4 *queue* $\leftarrow$ *queue* $\cup \{\langle V,$ *tuples*$\rangle | \langle V,$ *tuples*$\rangle \in$ *New-pairs* and *tuples* not singleton$\}$
4. **return** *vects*

Figure 5: Procedure VectorCover

Intuitively, the the minimal generalizations of table T_i are exactly those tables T_j satisfying k-anonymity with minimal distance vectors $\mathsf{DV}_{i,j}$. For instance, with reference to the generalized tables illustrated in Figure 3 we have already noticed how, for $k = 3$, $\mathsf{GT}_{[1,1]}$ cannot be minimal because $\mathsf{GT}_{[0,1]}$ and $\mathsf{GT}_{[1,0]}$ also satisfy k-anonimity. (Remember that the subscript indicates the distance vector of the generalized table from PT).

Our minimal generalization algorithm also makes use of another form of the distance vector that captures how far apart two tuples belonging to the same table are. Let T be a table and $x, y \in T$ two tuples such that $x = \langle v'_1, \ldots, v'_n \rangle$ and $y = \langle v''_1, \ldots, v''_n \rangle$ be two tuples where each v_i, v'_i is a value in domain D_i. The **distance vector** between x and y is the vector $\mathsf{V}_{x,y} = [d_1, \ldots, d_n]$ where d_i is the length of the paths from v'_i and v''_i to their closest common ancestor in the value generalization hierarchy VGH_{D_i}. For instance, with reference to the PT illustrated in Figure 3, the distance between $\langle$`a,39`$\rangle$ and $\langle$`b,39`$\rangle$ is [1,0].

The relationship between distance vectors for tables and distance vectors for tuples is important. Intuitively, the distance between two tuples x and y in table T is the distance vector between T and the table T', with $T \leq T'$ where the domains of the attribute in T' are the most specific domains for which x and y generalize to the same tuple t. This observation brings us to the following theorem.

Theorem 4.2 *Let T_i and T_j be two tables such that $T_i \leq T_j$. If T_j is k-minimal then $\mathsf{DV}_{j,j} \leq V_\oplus$, where $V_\oplus$ is the least upper bound of all the distance vectors $V_{x,y}$ where x, y are tuples in T_i and either x or y has a number of occurrences smaller than k.*

According to Theorem 4.2 the distance vector of a minimal generalization of a table cannot be greater than the smallest vector dominating all possible distance vectors between the tuples in T_i. This property is exploited by the minimal generalization algorithm to reduce the number of possible strategies to be evaluated, as illustrated in the following subsection.

4.2 Reducing the number of strategies: VectorCover

As discussed in Section 3 each k-minimal generalization is locally minimal to a specific generalization strategy. Performing all possible generalizations in the domain generalization hierarchy would eventually reveal a minimal generalization strategy in which an instance of data achieves a k-anonymity requirement. However, the large number of possible strategies (see Theorem 3.2) makes this approach largely impractical. Theorem 4.2 above helps reduce the number of generalizations (strategies) to be considered. Though reduced, this approach bears the cost of determining all the possible vectors between outliers (tuples which have less than k occurrences) and every other tuple in the table. In the worst case, where all tuples are outliers, the cost of this operation is $O(N^2)$ where N is the total number of tuples in the table. Given this observation, instead of constructing all possible vectors, we present an efficient algorithm named, VectorCover, that given the set of outliers and the set of all the tuples in the table, without computing all pairwise combinations, determines the different distance vectors between the outliers and the other

www.manaraa.com

Preferred Minimal Generalization (MinGen) Algorithm
Input: Private table PT; quasi-identifier $QI = (A_1, \ldots, A_n)$ and k-anonymity constraint; preference specifications.
Output: A minimal generalization MGT of PT$[QI]$ wrt k-anonymity chosen according to the preference specifications
Method
1. MGT $\leftarrow$ PT$[QI]$ /* create a copy of PT$[QI]$ */
2. *history* $\leftarrow [d_1, \ldots, d_n]$, where $d_i = 0, i = 1, \ldots, n$.
3. **while** there exists an attribute $A_i \in QI$ and value $v \in$ MGT$[A_i]$ such that freq$(v,$ MGT$[A_i]) < k$ **do**
   3.1 let $V_i = [d_1, \ldots, d_n]$ be the vector such that $d_j = 1$ if $j = i, d_j = 0$ otherwise
   3.2 MGT $\leftarrow$ generalize(MGT, V_i, *history*)
   3.3 *history* $\leftarrow$ *history* + V_i /* where $[x_1, \ldots, x_n] + [y_1, \ldots, y_n] = [x_1 + y_1, \ldots, x_n + y_n]$*/
4. *All* $\leftarrow \{t \mid t$ is a distinct tuple in MGT$\}$
5. *Outliers* $\leftarrow \{t \mid t \in All,$ freq$(t,$ MGT$[QI]) < k\}$
6. **if** *Outliers* $\neq \emptyset$ **do**
   6.1 *Distances* $\leftarrow$ VectorCover(*Outliers*,*All*) /* sorted list of vectors to be considered */
   6.2 *Strategies* $\leftarrow$ Levelsets(*Distances*)
/* Orders sets of vectors in levels */
   6.3 *lmin* $\leftarrow$ binsearch(*Strategies*,MGT,*history*,k)
   6.4 *Allmins* $\leftarrow \{V \mid \langle lmin, Vset_{lmin}\rangle \in Strategies, V \in Vset_{lmin},$ and $\forall t \in G_V,$ freq$(t, G_V) \geq k$ where $G_V =$ generalize(MGT, V, *history*)$\}$
   6.5 *pref-vec* $\leftarrow$ preferred(MGT,*AllMins*,*history*) /* select a vector corresponding to preference specificaton */
   6.6 MGT $\leftarrow$ generalize(MGT, *pref-vec*, *history*).
7. **return** MGT.

Figure 6: The Preferred Minimal Generalization (MinGen) Algorithm

tuples. Techniques that interpret distance vectors use a distance function dist based on the sum of the elements in the vector. More formally, given a vector $V = [d_1, \ldots, d_n]$, dist$(V) = \sum_{i=1}^{n} d_i$. It is easy to see that the distance function satisfies the properties of Euclidean geometry, i.e., for all possible tuples x, y, and z: *1)* dist$(V_{x,y}) \geq 0$; *2)* dist$(V_{x,y}) = 0$ iff $x = y$; *3)* dist$(V_{x,y}) =$ dist$(V_{y,x})$; and *4)* dist$(V_{x,y}) \leq$ dist$(V_{x,z}) +$ dist$(V_{z,y})$.

VectorCover analogizes the problem of finding generalization strategies stored as vectors, to that of finding a minimum spanning tree on a clique. The nodes are the tuples and the weights on the edges are the distance vectors between the tuples. A technique for finding an optimum cover was presented by Agrawal et al. in[2]. However, the algorithm proposed in [2] does not apply efficiently to cliques (the performance is $O(N^2)$ where N is the total number of tuples). Moreover, with respect to [2] we have an additional requirement for the spanning tree, which is that the set of edges contained in the spanning tree must include at least one occurrence of each distinct distance vector found in the clique. Despite that, we can do better than $O(N^2)$ by exploiting the Euclidean properties of the distance function defined on vectors. In particular, VectorCover does not automatically visit all incident unvisited neighbors, but instead uses the distances to cluster tuples together so that tuples that are far away from a given tuple may be close to each other. Like in [2], VectorCover removes old edges and adds new ones to reduce the overall sum of the weights on all the edges. An algorithm implementing VectorCover is illustrated in Figure 5. For space requirements, we refer the reader to Figure 5 for more details on the algorithm. Here we only note that the key to implementing VectorCover efficiently is to make it easy to select an unvisited edge that is likely to have a distance vector less than known distances. To this purpose we exploit the Euclidean nature of the distance function which ensures that whenever two tuples, x and y, incident to a third tuple, z, the vector representing the least upper bound between $V_{z,x}$ and $V_{z,y}$ places an upper bound on $V_{x,y}$. When known distances place an upper bound on an unvisited edge, the edge is visited and distances to neighboring tuples that have similar distances from a common tuple are computed. The correctness of the procedure is stated by the following theorem.

Theorem 4.3 *Let Outliers and All be two sets of tuples. VectorCover(Outliers,All) finds all and only the distinct vectors $V_{x,y}$ such that $x \in$ Outliers and $y \in$ All.*

The performance of VectorCover depends on the nature of the data. Let N be the number of tuples in PT. It is N in the best case, $O(N)$ in the general case, deteriorating to $O(N^2)$ only in the worst case where the only minimal generalization is the one to the most general domains.

www.manaraa.com

4.3 The Preferred Minimal Generalization Algorithm (MinGen)

Figure 6 illustrated an algorithm, called *MinGen*, which, given a table PT, a quasi-identifier $QI \in \mathsf{PT}_{QI}$, and a k-anonymity constraint, produces a table MGT which is a k-minimal generalization of PT$[QI]$. It assumes that $k < |\mathsf{PT}|$, which is a necessary and sufficient condition for the existence of a minimal generalized table (see Theorem 3.1). The algorithm makes use of two basic functions: preferred, which selects a vector that correspond to a preferred generalization according to the given specifications; and generalize, which enforces generalization. The generalize function requires as parameters a table T_i and two distance vectors v and h, and returns the table T_j, generalization of T_i, with $\mathsf{DV}_{i,j} = s$. The third parameter h, constructed by the algorithm, represent the distance of T_i from the private table PT provided as input, and is used to ensure that the generalization is absolute with respect to the domains in PT. The algorithm uses also function VectorCover discussed in the previous subsection, and functions Levelsets and binsearch, reported in the appendix, which we explain here. Given a set of vectors ordered in increasing distance, Levelsets defines an assignment of levels to vectors as follows. Vectors are considered in the order in which they appear in the list. Each new vector V considered is assigned to the lowest level at which there does not exists any vector $V' < V$. Once all the vectors have been considered, vector V_u representing the lowest appear bound of the vectors in the considered set is determined. If V_u does not belong to the set a new higher level is created to which V_u is assigned. Moreover, the levels are examined and possibly updated as follows. Each vector V appearing at a level l is replicated at all the levels between the lowest the highest level (this excluded) at which there exists a vector dominated by V and the lowest level (this excluded) at which there is a vector dominating V. This "padding" process ensures that, if we represent the order between vectors as a graph, all paths between the zero vector and all the vectors at a given level have the same distance.

For the sake of space we omit the complete explanation of the algorithm and refer the reader to Figure 6 for details. Here, we outline only the major steps. Step 3 generalizes each single attribute to satisfy k-anonymity since k-anonymity of each attribute is a necessary condition for the k-anonymity of the quasi-identifier. Step 6 is the core of the algorithm. Substep 6.1 calls procedure VectorCover to retrieve the distance vectors corresponding to generalization strategies to be evaluated. Distance vectors are returned in a list sorted according to increasing distance. Step 6.2 calls Levelsets to retrieve set *Strategies* representing the assignment of levels to vectors returned by VectorCover. Step 6.3 calls binsearch which performs a binary search on the levels and returns the lowest level at which there exists a vector corresponding to a minimal generalization.

The correctness of the algorithm relies on the correctness of function VectorCover and on the particular way levels are constructed in Levelsets. In particular, it can be proved that, for how levels are defined, if a vector corresponding to a minimal generalization appears at level l than no other vectors corresponding to a minimal generalization can exists at level lower than l. This property allows us to efficiently retrieve the interested level by performing the binary search.

With respect to the complexity we make the following observations. Let N be the number of tuples in PT. The loop to achieve k-anonymity at each attribute (step 3) is at most $O(N * |QI|)$. VectorCover has a best case of N and a worst case of $O(N^2)$. The construction of *strategies* can be done in logarithmic time. The binary search is performed in $O(\log(\sum_{i=1}^{n} h_i))$ where h_i is the height of the hierarchy of the domain of attribute A_i. The complexity of the test in binsearch to determine whether a level achieves k-anonymity is $O(|distances| * |\mathsf{MGT}|)$. In the worst case $|distances| = \prod_{i=1}^{n} h_i$; however VectorCover guarantees $|distances|$ to be minimal. These observations imply that the number n of attributes in QI is an important factor affecting the computational cost. To consider this computational cost with respect to general cases of application, we surveyed real-word data [11]. The number of attributes in quasi-identifiers found in many releases of census data, voter lists, public health data, and medical records ranged from 3 to 5 attributes and the heights of the hierarchies ranged from 1 to 8. Because *MinGen* searches only over the generalizations specific to the data, in the general case the number of candidate generalizations dramatically reduced to a small fraction of all possible generalizations. In 300 pediatric medical records with 7617 visits and 285 fields stored in over 12 relational database tables. We found that generalizations in *trygen* were 3-7% of the total possible number of generalizations when we considered 4 independent sets of quasi-identifiers known to link to public and semi-public information.

5 Conclusions

The practical significance of releasing individualized-data such that linking the data to other sources to re-identify individuals cannot be done, offers many benefits to our electronic society. This work provides an effective and optimal

www.manaraa.com

solution to this problem. Open questions remain. The size of and conditions for k necessary to ensure k-anonymity must be further investigated. The quality of generalized data is best when the attributes most important to the recipient do not belong to any quasi-identifier. For public-use files this is acceptable, but determining the quality and usefulness in other settings must be further researched. Finally, data are dynamic. Tuples are added, changed, and removed constantly. As a result, releases of generalized data over time can be subject to a temporal inference attack, which need to be carefully investigated.

Acknowledgements

The authors thank Steve Dawson, at SRI, for discussions and support; Sylvia Barrett, at Harvard, for support; and, Eric Jordan, Patrick Thompson and Mojdeh Mohtashemi, at MIT, for editorial suggestions.

References

[1] N.R. Adam and J.C. Wortman. Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21:515–556, 1989.

[2] Rakesh Agrawal, Alex Borgida, and H.V. Jagadish. Efficient management of transitive relationships in large data and knowledge bases. In Bruce Linsday James Clifford and David Maier, editors, *Proc. of the 1989 ACM SIGMOD Int'l Conf. on Management of Data*, pages 253–262, Portland, Oregon, June 1989.

[3] P.C. Chu. Cell suppression methodology: The importance of suppressing marginal totals. *IEEE Trans. on Knowledge Data Systems*, 4(9):513–523, July/August 1997.

[4] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[5] D. Denning, P. Denning, and M.D. Schwartz. The tracker: A threat to statistical database security. *ACM Trans. on Database Systems*, 4(1):76–96, March 1979.

[6] N. Kirkendall et. al. Report on statistical disclosure limitation methodology. Statistical policy working paper, no. 22, Washington: Office of Management and Budget, 1994.

[7] A. Hundepool and L. Willenborg. μ- and τ-Argus: Software for statistical disclosure control. In *Third International Seminar on Statistical Confidentiality*, Bled., 1996.

[8] Sushil Jajodia, Pierangela Samarati, V.S. Subrahmanian, and Elisa Bertino. A unified framework for enforcing multiple access control policies. In *Proc. of the 1997 ACM SIGMOD Int'l Conf. on Management of Data*, Tucson, AZ, U.S.A, May 1997.

[9] Malvestuto, F. Moscarini, and M. Rafanelli. Suppressing marginal cells to protect sensitive information in a two-dimensional statistical table. In *Proceedings of the Tenth ACM Symposium on Principles of Database Systems*, Denver, CO, May 1991.

[10] Latanya Sweeney. Guaranteeing anonymity when sharing medical data, the Datafly system. In *Bureau of the Census, Record Linkage Bulletin.*, Washington,DC: Bureau of the Census,, 1997.

[11] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. *Journal of Law, Medicine, & Ethics*, 25(2–3):98–110, 1997.

[12] Tian Zhang, Raghu Ramakrishan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In H.V. Jagadish and I.S. Mumick, editors, *Proc. of the 1996 ACM SIGMOD Int'l Conf. on Management of Data*, pages 103–114, Montreal, Canada, June 1996.

www.manaraa.com

A Additional Algorithms

In this section we report the functions Levelsets and binsearch used by algorithm *MinGen*, whose behavior has been illustaretd in the paper. The semantics of Levelsets has been illustrated in the paper. The algorithm below enforces an optimized version of the level construction based on the use of a graph.

LevelSets(*Distances*)
2. $D \leftarrow \{(dists,\ E)\}$ /* build DAG where *dists* = *distances*, and $e_{x,y}$ element of E iff d_x, d_y element of distances and $d_x < d_y$ }
   /* make all path lengths same by padding nodes as follows */
3. **for all** d_1, d_2, d_3 element of *dists* and $e_{13}, e_{23} \in E$
   3.1 $l1 \leftarrow$ length of path from $[0, ..., 0]$ to d_1 in D
   3.2 $l2 \leftarrow$ length of path from $[0, ..., 0]$ to d_2 in D
   3.3 **if** $l_1 > l_2$ **then** $dists \leftarrow dists \cup d2'$; $E \leftarrow E \cup \{e_2, 2'\}$
   3.4 **else if** $l_2 > l_1$ **then** $dists \leftarrow dists \cup \{d_1'\}$; $E \leftarrow E \cup \{e1, 1'\}$ /* d' and d have the same distance vectors but $d \neq d'$ */
4. $lsets \leftarrow \emptyset$
5. $maxpath \leftarrow max$ path length in D
6. **for** i=1,...,*maxpath* **do**
   6.1 $lsets \leftarrow lsets \cup \{\langle i, S_i\rangle | d \in S_i$ iff $d \in dists$ and length of path from [0,...,0] to d in D is i }
7. $lsets \leftarrow lsets \cup \{\langle l_{mx}, S_{mx}\rangle | l_{mx} = maxpath{+}1$, and $v_m = d_i \oplus d_j\ \forall d_i, d_j \in S_m$ where $\langle maxpath, S_m\rangle \in lsets\}$
   /* $[x_1, \ldots, x_n] \oplus [y_1, \ldots, y_n] = [\max(x_1, y_1), \ldots, \max(x_n, y_n)]$ */
8. **return** *lsets*

binsearch(*Strategies*,MGT,*history*,*k*) /* determines the lowest level with a minimal generalization satisfying k-anonymity*/
1. $min \leftarrow 1$; $max \leftarrow \max\{i | \langle i, G_i\rangle \in Strategies\}$
2. **while** $min < max$ **do**
   2.1 $try \leftarrow \lfloor \frac{min+max}{2} \rfloor$
   2.2 $trygens \leftarrow \{tryvecs | \langle try, tryvecs\rangle \in Strategies\}$
   2.3 $reach\_k \leftarrow$ `false`
   2.4 **while** $trygens \neq \emptyset \wedge reach\_k \neq$ `true` **do**
      2.4.1 Select a distance vector V from *trygens*
      2.4.2 $trygens \leftarrow trygens \Leftrightarrow \{V\}$
      2.4.3 $Gen\_try \leftarrow$ generalize(MGT, V, *history*)
      2.4.4 **if** all tuples $t \in Gen\_try$ have freq(t, Gen_try) $\geq k$ **then** $reach\_k \leftarrow$ `true`
   2.5 **if** $reach\_k =$ `true` **then** $max \leftarrow try$ **else** $min \leftarrow try + 1$
3. **return** *min*

www.manaraa.com

B An Example

In this section we illustrate the behavior of the algorithm in the generalization of the private table illustrated in Figure 1. Let QI={Ethnicity, Date Of Birth, Sex, ZIP, Marital Status}. The resulting table PT[QI] to be k-anonymized is reported in Figure 7. For the sake of space, in Figure 7 names and values of attributes are abbreviated. We assume the hierarchies for ZIP and Ethnicity to be as illustrated in Figure 2. Domain hierarchies for Date_of_Birth (assuming values in the sixties), Sex, and Marital Status are illustrated in Figure 8. The hierarchy for dates maps specific dates the form `mm/dd/yy` in domain D_0, into months `mm/yy` in domain D_1, years (`yy`) in domain D_2, 5-year intervals in D_3 and the complete 10-year interval in D_4. (Note that although the hierarchy can seem to change the format of the data, compatibility of specific and generalized values can be ensured by using the same storage representation form. For instance, generalization of the month can be represented by `mm/1/yy`, generalization to the year by `1/1/yy` and so on.) The value hierarchy for Sex maps each possible value of S_0 into value `not_released` of S_1. The hierarchy for Marital Status M_0 is illustrated in Figure 8.

The ordered set of vectors returned by VectorCover and the levels constructed by Levelsets are reported below. Vectors in italic represent vectors replicated in the "padding" process. The vectors in bold face represents the least upper bound of all the vectors in *Distances*. Note that these generalization vectors are relative with respect to vector *history* $[0, 1, 0, 0, 0]$ representing the generalization performed on attribute DOB in step 3 of the algorithm.

Distances	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**	**Level 6**
00000	00000	00011	02002	12102	12112	**12122**
00011		01001	12020	02122	*12122*	
01001		01010	02120	12021	*12021*	
01010		01010	10111	*10111*		
02002		10100	11110	*11110*		
02120						
02122						
10100						
10111						
11110						
12020						
12021						
12102						
12112						

Figure 9 illustrates the k-minimal generalizations which can be returned by the algorithm for k=2 in dependence of the preference specified.

www.manaraa.com

	Eth	DOB	Sex	ZIP	Mar
1	b	09/27/64	m	02139	div
2	b	09/30/64	m	02139	div
3	b	04/18/64	m	02139	mar
4	b	04/15/64	m	02139	mar
5	b	09/15/64	m	02138	mar
6	c	03/23/63	m	02141	mar
7	c	03/18/63	m	02141	mar
8	c	09/13/64	f	02138	mar
9	c	09/07/64	f	02138	mar
10	c	05/14/61	f	02138	sin
11	c	05/08/61	f	02138	sin

Figure 7: Stored Values for the quasi-identifier of PT of Figure 1

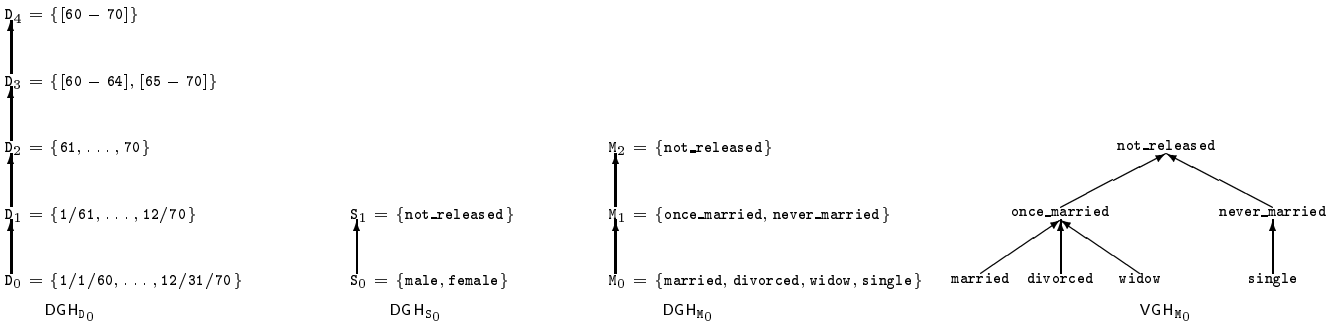

Figure 8: Examples of Domain and Value Generalization Hierarchies

	Eth	DOB	Sex	ZIP	Mar
1	b	64	m	02130	div
2	b	64	m	02130	div
3	b	64	m	02130	mar
4	b	64	m	02130	mar
5	b	64	m	02130	mar
6	c	63	m	02140	mar
7	c	63	m	02140	mar
8	c	64	f	02130	mar
9	c	64	f	02130	mar
10	c	61	f	02130	sin
11	c	61	f	02130	sin

$GT_{[0,2,0,1,0]}$

	Eth	DOB	Sex	ZIP	Mar
1	b	09/64	m	02130	once
2	b	09/64	m	02130	once
3	b	04/64	m	02130	once
4	b	04/64	m	02130	once
5	b	09/64	m	02130	once
6	c	03/63	m	02140	once
7	c	03/63	m	02140	once
8	c	09/64	f	02130	once
9	c	09/64	f	02130	once
10	c	05/61	f	02130	never
11	c	05/61	f	02130	never

$GT_{[0,1,0,1,1]}$

	Eth	DOB	Sex	ZIP	Mar
1	p	09/64	n_r	02139	div
2	p	09/64	n_r	02139	div
3	p	04/64	n_r	02139	mar
4	p	04/64	n_r	02139	mar
5	p	09/64	n_r	02138	mar
6	p	03/63	n_r	02141	mar
7	p	03/63	n_r	02141	mar
8	p	09/64	n_r	02138	mar
9	p	09/64	n_r	02138	mar
10	p	05/61	n_r	02138	sin
11	p	05/61	n_r	02138	sin

$GT_{[1,1,1,0,0]}$

Figure 9: Example of minimal generalized tables for the table of Figure 7

www.manaraa.com