Incognito: Efficient Full-Domain K-Anonymity

Kristen LeFevre David J. DeWitt Raghu Ramakrishnan
University of Wisconsin - Madison
1210 West Dayton St.
Madison, WI 53706

ABSTRACT

A number of organizations publish microdata for purposes such as public health and demographic research. Although attributes that clearly identify individuals, such as Name and Social Security Number, are generally removed, these databases can sometimes be joined with other public databases on attributes such as Zipcode, Sex, and Birthdate to re-identify individuals who were supposed to remain anonymous. "Joining" attacks are made easier by the availability of other, complementary, databases over the Internet.

K-anonymization is a technique that prevents joining attacks by generalizing and/or suppressing portions of the released microdata so that no individual can be uniquely distinguished from a group of size k. In this paper, we provide a practical framework for implementing one model of k-anonymization, called full-domain generalization. We introduce a set of algorithms for producing minimal full-domain generalizations, and show that these algorithms perform up to an order of magnitude faster than previous algorithms on two real-life databases.

Besides full-domain generalization, numerous other models have also been proposed for k-anonymization. The second contribution in this paper is a single taxonomy that categorizes previous models and introduces some promising new alternatives.

1. INTRODUCTION

Numerous organizations publish microdata[1] for a variety of different purposes, including demographic and public health research. In most cases, data that is deemed sensitive is "de-identified" by removing attributes known to uniquely identify individuals, such as Name or Social Security #. However, this mechanism fails to account for the possibility of combining seemingly innocuous attributes with external data to uniquely identify individuals. For example, according to one study, approximately 87% of the population of the United States can be uniquely identified on the basis of their 5-digit zipcode, sex, and date of birth [18].

Voter Registration Data

Name	Birthdate	Sex	Zipcode
Andre	1/21/76	Male	53715
Beth	1/10/81	Female	55410
Carol	10/1/44	Female	90210
Dan	2/21/84	Male	02174
Ellen	4/19/72	Female	02237

Hospital Patient Data

Birthdate	Sex	Zipcode	Disease
1/21/76	Male	53715	Flu
4/13/86	Female	53715	Hepatitis
2/28/76	Male	53703	Brochitis
1/21/76	Male	53703	Broken Arm
4/13/86	Female	53706	Sprained Ankle
2/28/76	Female	53706	Hang Nail

Figure 1: Tables vulnerable to a joining attack.

The uniqueness of such attribute combinations leads to a class of attacks where data is "re-identified" by joining multiple (often publicly-available) data sets. This type of attack was illustrated in [18], where the author was able to join a public voter registration list and the de-identified patient data of Massachusetts's state employees to determine the medical history of the state's governor. Figure 1 shows an example of such an attack, where a malicious individual is able to determine Andre's medical information by joining the two databases on Birthdate, Sex, and Zipcode. We will use the Patients table as a running example.

1.1 Basic Definitions

Quasi-Identifier Attribute Set *A quasi-identifer set Q is a minimal set of attributes in table T that can be joined with external information to re-identify individual records (with sufficiently high probability)*[15].

We assume that quasi-identifier attribute sets are known based on specific knowledge of the domain.

Frequency Set *Consider relation T and some set Q of n attributes. The frequency set of T with respect to Q is a mapping from each unique combination of values $\langle q_0, ..., q_n \rangle$ of Q in T (the value groups) to the total number of tuples in T with these values of Q (the counts).*

K-Anonymity Property *Relation T is said to satisfy the k-anonymity property (or to be k-anonymous) with respect to attribute set Q if every count in the frequency set of T with respect to Q is greater than or equal to k.*

Frequency = insieme di quante volte una tupla compare

[1]The term "microdata" has been used in the Statistics literature to refer to data published in its raw, non-aggregated, form [20].

SIGMOD 2005 June 14-16, 2005, Baltimore, Maryland, USA

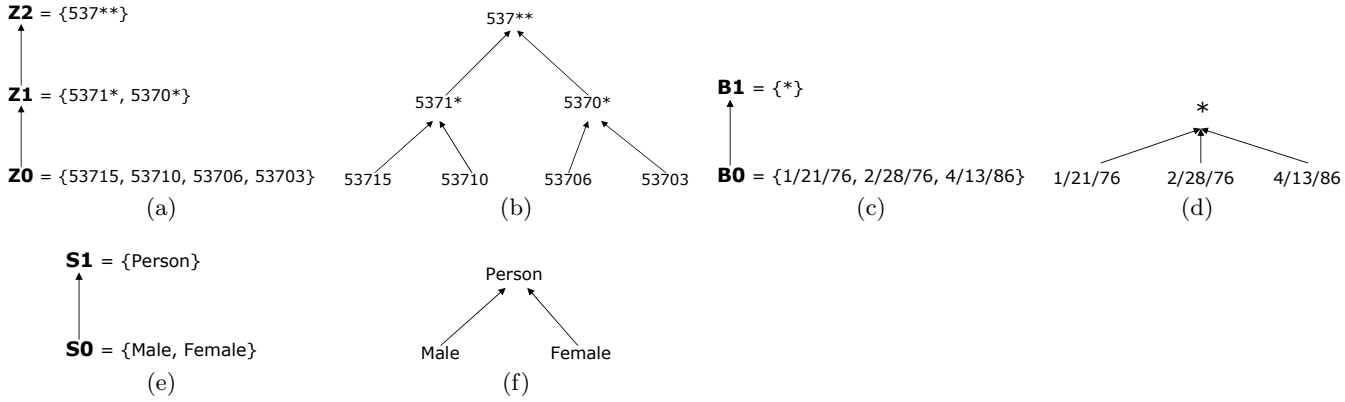

Figure 2: Domain and value generalization hierarchies for Zipcode (a, b), Birth Date (c, d), and Sex (e, f)

In SQL, the frequency set is obtained from T with respect to a set of attributes Q by issuing a COUNT(*) query, with Q as the attribute list in the GROUP BY clause. For example, in order to check whether the Patients table in Figure 1 is 2-anonymous with respect to ⟨Sex, Zipcode⟩, we issue a query SELECT COUNT(*) FROM Patients GROUP BY Sex, Zipcode. Since the result includes groups with count fewer than 2, Patients is not 2-anonymous with respect to ⟨Sex, Zipcode⟩.

K-Anonymization *A view V of a relation T is said to be a k-anonymization of T if the view modifies, distorts, or suppresses the data of T according to some mechanism such that V satisfies the k-anonymity property with respect to the set of quasi-identifier attributes. T and V are assumed to be multisets of tuples.*

Throughout this paper, we consider the problem of generating a single k-anonymization of the microdata in a single table T. Other problems, including inference, arise when multiple different anonymizations of the same microdata are made available[18], but we assume that only a single anonymization is released.

1.2 Paper Organization and Contributions

In Section 2 we give an overview of the generalization and suppression framework for k-anonymization, in particular a model called full-domain generalization, and we describe previous algorithms implementing minimal full-domain generalization.

Our first main contribution addresses implementation of full-domain k-anonymization, and is presented in Sections 3 and 4. In Section 3 we introduce an implementation framework for full-domain generalization using a multi-dimensional data model, together with a suite of algorithms, which we call Incognito. Incognito takes advantage of two key variations of dynamic programming [4] that have been used previously in the query processing literature for other purposes: bottom-up aggregation along dimensional hierarchies [6] and a priori aggregate computation [2]. In Section 4 we present the results of the largest-scale performance experiments that we are aware of for minimal k-anonymization, and we show that the Incognito algorithms outperform previous algorithms by up to an order of magnitude. The results demonstrate the feasibility of performing minimal k-anonymization on large databases.

Though our algorithms and framework focus primarily on the full-domain generalization model, there have been a number of other k-anonymization models proposed, but the differences among these techniques have not been clearly articulated. Our second contribution, presented in Section 5, is a unifying taxonomy of several alternative approaches to k-anonymization. Previous proposals can be understood as instances of this taxonomy, differing primarily in the granularity at which anonymization is applied. Further, the taxonomy exposes some interesting new alternatives that offer the promise of more flexible anonymization. Extending the algorithmic framework presented in this paper to some of these novel alternatives presents a broad class of problems for future work.

We discuss related work in Section 6 and present our conclusions in Section 7.

2. GENERALIZATION AND SUPPRESSION

Samarati and Sweeney [14, 15, 17, 18] formulated mechanisms for k-anonymization using the ideas of generalization and suppression. In a relational database, there is a domain (e.g., integer, date) associated with each attribute of a relation. Given this domain, it is possible to construct a more "general" domain in a variety of ways. For example, the Zipcode domain can be generalized by dropping the least significant digit, and continuous attribute domains can be generalized into ranges.

We denote this *domain generalization relationship* by $<_D$, and we use the notation $D_i \leq_D D_j$ to denote that domain D_j is either identical to or a domain generalization of D_i. For two domains D_i and D_j, the relationship $D_i <_D D_j$ indicates that the values in domain D_j are the generalizations of the values in domain D_i. More precisely, a many-to-one *value generalization function* $\gamma : D_i \rightarrow D_j$ is associated with each domain generalization $D_i <_D D_j$.

A *domain generalization hierarchy* is defined to be a set of domains that is totally ordered by the relationship $<_D$. We can think of the hierarchy as a chain of nodes, and if there is an edge from D_i to D_j, we call D_j the *direct generalization* of D_i. Note that the generalization relationship is transitive, and thus, if $D_i <_D D_j$ and $D_j <_D D_k$, then $D_i <_D D_k$. In this case, we call domain D_k an *implied generalization* of D_i. Paths in a domain hierarchy chain correspond to implied generalizations, and edges correspond to direct generalizations. Figure 2 (a,c,e) shows possible domain generalization hierarchies for the Zipcode, Birthdate and Sex attributes.

We use the notation γ^+ as shorthand for the composition of one or more value generalization functions, producing the *direct* and *implied value generalizations*. The value-

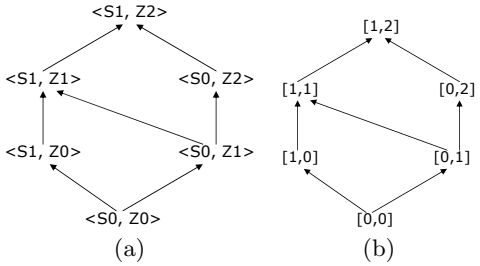

Figure 3: Generalization lattice for the Zipcode and Sex attributes (a), and the corresponding lattice of distance vectors (b)

generalization functions associated with a domain generalization hierarchy induce a corresponding value-level tree, in which edges are defined by γ and paths are defined by γ^+. To illustrate, Figure 2 (b,d,f) associates a value generalization with each value in the Zipcode, Birthdate, and Sex domains. For example, (b) indicates that $5371* = \gamma(53715)$ and $537** \in \gamma^+(53715)$.

For a quasi-identifier consisting of multiple attributes, each with its own domain, the domain generalization hierarchies of the individual attributes can be combined to form a multi-attribute generalization lattice. An example lattice for Sex and Zipcode is shown in Figure 3 (a).

Formally, consider a vector of n domains with corresponding domain hierarchies $H_1 \ldots H_n$. A vector of n domains $\langle D_{A_1}, ..., D_{A_n} \rangle$ is said to be a *direct multi-attribute domain generalization* (also denoted $<_D$) of another vector of n domains $\langle D_{B_1}, ..., D_{B_n} \rangle$ if the following conditions hold:

1. There exists a single value j in $1 \ldots n$ such that domain hierarchy H_j contains the edge $D_{A_j} \longrightarrow D_{B_j}$ (i.e., D_{B_j} is a direct domain generalization of D_{A_j}).
2. For all other i in $1 \ldots n$, i.e., for $i \neq j$, $D_{A_i} = D_{B_i}$.

A *multi-attribute generalization lattice* over n single-attribute domain generalization hierarchies is a complete lattice of n-vectors of domains in which

1. Each edge is a direct multi-attribute domain generalization relationship.
2. The bottom element is the n-vector $\langle D_{A_1}, ..., D_{A_n} \rangle$, where, for all i, D_{A_i} is the source of the hierarchy chain H_i (i.e., the most specific domain associated with domain hierarchy H_i).
3. The top element is the n-vector $\langle D_{A_1}, ..., D_{A_n} \rangle$, where, for all i, D_{A_i} is the sink of the hierarchy chain H_i (ie., the most general domain associated with domain hierarchy H_i).

In the example lattice shown in Figure 3, the domain vector $\langle S_0, Z_2 \rangle$ is a direct multi-attribute generalization of $\langle S_0, Z_1 \rangle$ and an implied multi-attribute generalization of $\langle S_0, Z_0 \rangle$.

From the generalization lattice, a lattice of distance vectors can be derived. The *distance vector* between two domain vectors $\langle D_{A_1}, .., D_{A_n} \rangle$ and $\langle D_{B_1}, .., D_{B_n} \rangle$ is a vector $DV = [d_1, ..., d_n]$, where each value d_i denotes the length of the path between the domain D_{A_i} and the domain D_{B_i} in domain generalization hierarchy H_i. A lattice of distance vectors can be defined from the zero-generalization domain vector. This lattice for Sex and Zipcode is given in Figure 3(b). The *height* of a multi-attribute generalization is the sum of the values in the corresponding distance vector. For example, the height of $\langle S_1, Z_1 \rangle$ is 2.

2.1 Full-Domain Generalization

There are a number of models for producing an anonymization V from table T. One class of models, called *global-recoding* [20], map the values in the domains of quasi-identifier attributes to other values.

This paper is primarily concerned with a specific global-recoding model, called *full-domain generalization*, though we will describe a number of alternative schemes in Section 5. Full-domain generalization was proposed by Samarati and Sweeney [14, 15] and maps the entire domain of each quasi-identifier attribute in T to a more general domain in its domain generalization hierarchy. This scheme guarantees that all values of a particular attribute in V belong to the same domain.

Full-Domain Generalization *Let T be a relation with quasi-identifier attributes $Q_1, ..., Q_n$. A full-domain generalization is defined by a set of functions, $\phi_1, ..., \phi_n$, each of the form $\phi_i : D_{Q_i} \rightarrow D_{A_i}$, where $D_{Q_i} \leq_D D_{A_i}$. ϕ_i maps each value $q \in D_{Q_i}$ to some $a \in D_{A_i}$ such that $a = q$ or $a \in \gamma^+(q)$. A full-domain generalization V of T is obtained by replacing the value q of attribute Q_i in each tuple of T with the value $\phi_i(q)$.*

In addition, the idea of a tuple-suppression threshold has been suggested [14, 15], and can be used as a simple extension of full-domain generalization. The idea is that there are some number of records in T that can be considered outliers. For this reason, up to a certain number of records (the *maximum suppression threshold*) may be completely excluded from V. Under this combined scheme, V is obtained through full-domain generalization, with selected outlier tuples removed entirely.

For any anonymization mechanism, it is desirable to define some notion of minimality. Intuitively, a k-anonymization should not generalize, suppress, or distort the data more than is necessary to achieve k-anonymity. Indeed, there are a number of ways to define minimality. One notion of minimal full-domain generalization was defined in [14, 15] using the distance vector of the domain generalization. Informally, this definition says that a full-domain generalization V is minimal if V is k-anonymous, and the height of the resulting generalization is less than or equal to that of any other k-anonymous full-domain generalization.

However, in many cases, it is likely that users would want the flexibility to introduce their own, possibly application-specific, notions of minimality, as was described in [11, 14]. For example, it might be more important in some applications that the Sex attribute be released intact, even if this means additional generalization of Zipcode. The previous definition does not allow this flexibility.

2.2 Previous Full-Domain Generalization Algorithms

Several search algorithms have been proposed with accompanying guarantees about the minimality of the resulting anonymization. In [14], Samarati describes an algorithm for finding a single minimal k-anonymous full-domain generalization, based on the specific definition of minimality outlined in the previous section. The algorithm uses the observation that if no generalization of height h satisfies k-anonymity, then no generalization of height $h' < h$ will satisfy k-anonymity. For this reason, the algorithm performs a binary search on the height value. If the maximum height in the generalization lattice is h, it begins by checking each

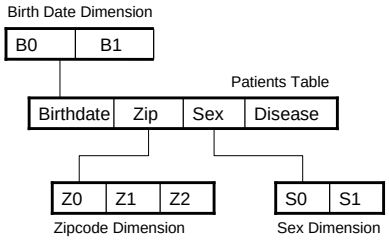

Figure 4: Star-schema including generalization dimensions for quasi-identifier attributes.

generalization at height $\lfloor \frac{h}{2} \rfloor$. If a generalization exists at this height that satisfies k-anonymity, the search proceeds to look at the generalizations of height $\lfloor \frac{h}{4} \rfloor$. Otherwise, it searches the generalizations of height $\lfloor \frac{3h}{4} \rfloor$, and so forth. This algorithm is proven to find a *single* minimal full-domain k-anonymization according to this definition.

For arbitrary definitions of minimality, this binary search algorithm is not always guaranteed to find the minimal generalization. Instead, a naive bottom-up breadth-first search of the generalization lattice could be used. This algorithm uses the multi-attribute generalization lattice for the domains of the quasi-identifier attributes. Starting with the least general domain at the root of the lattice, the algorithm performs a breadth-first search, checking whether each generalization encountered satisfies k-anonymity. This algorithm can be used to find a single (weighted) minimal generalization, or it can be used to find the set of all k-anonymous domain generalizations.

In addition, we considered refining the bottom-up breath-first search using bottom-up aggregation ("rollup") along the domain generalization hierarchies. In this case, the search pattern is also breath-first, and the algorithm computes the frequency set of the data table with respect to each domain generalization encountered. However, for all generalizations other than the root, this frequency set is computed based on the frequency set of (one of) the generalization(s) of which the node is a direct generalization.

3. INCOGNITO

We noticed a number of convincing parallels between Samarati and Sweeney's generalization framework [14, 15] and ideas used in managing multi-dimensional data [6, 8] and mining association rules [2, 16]. By bringing these techniques to bear on the full-domain generalization problem, we developed a core algorithm (as well as several variations) that perform substantially better than previous algorithms.

There are a number of notable similarities between the generalization framework and multi-dimensional data models. For the problem of k-anonymization, the COUNT measure is of primary interest, and it is easy to think of each domain generalization hierarchy as a dimension. For this reason, it seemed reasonable to think of the table T, and the domain generalization hierarchies associated with the quasi-identifier attributes of T, as a relational star-schema. For example, the star schema for the quasi-identifier ⟨Birthdate, Sex, Zipcode⟩ is given in Figure 4.

A full-domain k-anonymization is produced by joining T with its dimension tables, and projecting the appropriate domain attributes. For simplicity, if A_1 is a domain generalization of some attribute A in the quasi-identifier, we refer to attribute A_1, which is produced by joining T with the dimension table of A, and projecting A_1.

Two properties of these generalization dimensions play a key role in our algorithms. The first is the generalization property of dimension hierarchies.

Generalization Property *Let T be a relation, and let P and Q be sets of attributes in T such that $D_P <_D D_Q$. If T is k-anonymous with respect to P, then T is also k-anonymous with respect to Q.*

Proof By the definition of multi-attribute domain generalization, there is some attribute pair $P_i \in P$ and $Q_i \in Q$ such that $D_{P_i} <_D D_{Q_i}$ and a many-to-one function γ from from D_{P_i} to D_{Q_i}. Thus the counts in the frequency set of T with respect to Q must be greater than or equal to the corresponding counts in the frequency set of T with respect to P. □

For example, because the Patients table in Figure 1 is 2-anonymous with respect to $\langle S_0 \rangle$, then it must also be 2-anonymous with respect to $\langle S_1 \rangle$, a generalization of S_0.

The key second property is reminiscent of operations along dimension hierarchies in OLAP processing.

Rollup Property *Let T be a relation, and let P and Q be sets of attributes such that $D_P \leq_D D_Q$. If we have f_1, the frequency set of T with respect to P, then we can generate each count in f_2, the frequency set of T with respect to Q, by summing the set of counts in f_1 associated by γ with each value set of f_2.*

Proof The proof follows from the definitions of frequency set and value generalization. □

For example, consider F_1, the relational representation of the frequency set of the Patients table from Figure 1 with respect to ⟨Birthdate, Sex, Zipcode⟩. Recall that in SQL the frequency set is computed by a COUNT(*) query with Birthdate, Sex, Zipcode as the GROUP BY clause. The frequency set (F_2) of Patients with respect to ⟨Birthdate, Sex, Z_1⟩ can be produced by joining F_1 with the Zipcode dimension table, and issuing a SUM(count) query with Birthdate, Sex, Z_1 as the GROUP BY clause.

We also noticed a strong connection between k-anonymity and the a priori observation, a dynamic programming approach that formed the basis for a number of algorithms for mining frequent itemsets [2, 16]. The a priori observation is easily applied to full-domain k-anonymization by way of the subset property.

Subset Property *Let T be a relation, and let Q be a set of attributes in T. If T is k-anonymous with respect to Q, then T is k-anonymous with respect to any set of attributes P such that $P \subseteq Q$.*

Proof Consider the frequency set of T with respect to Q, which partitions T based on the values of Q. If we remove any attribute Q_i from Q, then each of these partitions will remain the same, or will merge with another partition. Thus each count in the frequency set will either remain the same or increase. □ Q non ha una sola riga...

For example, Patients (Figure 1) is 2-anonymous with respect to $\langle S_1$, Zipcode⟩. Based on the subset property, we know that Patients must also be 2-anonymous with respect to both ⟨Zipcode⟩ and $\langle S_1 \rangle$. Similarly, we noted that Patients is not 2-anonymous with respect to ⟨Sex, Zipcode⟩. Based on this observation and the subset property, we know that Patients must not be 2-anonymous with respect to ⟨Birthdate, Sex, Zipcode⟩.

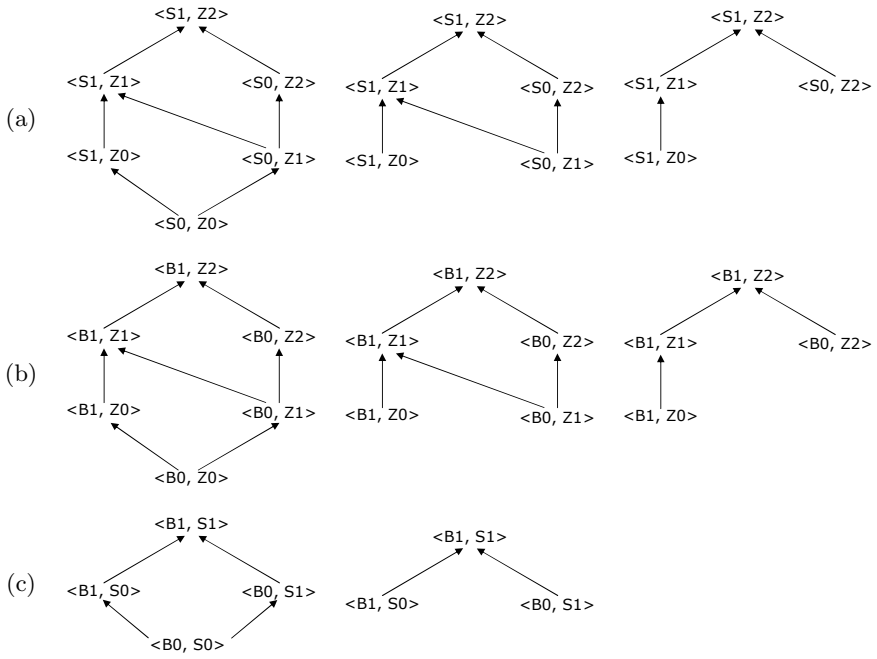

Figure 5: Searching the candidate 2-attribute generalization graphs for Patients example (Figure 1)

3.1 Basic Incognito Algorithm

The Incognito algorithm generates the set of all possible k-anonymous full-domain generalizations of T, with an optional tuple suppression threshold. Based on the subset property, the algorithm begins by checking single-attribute subsets of the quasi-identifier, and then iterates, checking k-anonymity with respect to increasingly large subsets, in a manner reminiscent of [2, 16]. Each iteration consists of two main parts: (The basic algorithm is given in Figure 8.)

1. Each iteration considers a graph of candidate multi-attribute generalizations (nodes) constructed from a subset of the quasi-identifier of size i. We denote the set of candidate nodes C_i. The set of direct multi-attribute generalization relationships (edges) connecting these nodes is denoted E_i. A modified breadth-first search over the graph yields the set of multi-attribute generalizations of size i with respect to which T is k-anonymous (denoted S_i).
2. After obtaining S_i, the algorithm constructs the set of candidate nodes of size $i+1$ (C_{i+1}), and the edges connecting them (E_{i+1}) using the subset property.

3.1.1 Breadth-First Search

The i^{th} iteration of Incognito performs a search that determines the k-anonymity status of table T with respect to all candidate generalizations in C_i. This is accomplished using a modified bottom-up breadth-first search, beginning at each node in the graph that is not the direct generalization of some other node, with the optimization of bottom-up aggregation based on the rollup property.

The breadth-first search also makes use of the generalization property. If a node satisfying k-anonymity is encountered, we are guaranteed by the generalization property that all of its generalizations must also satisfy k-anonymity. For this reason, when a node is found to be k-anonymous, all of its direct generalizations are marked, and not checked in subsequent iterations of the search.

Example 3.1 Consider again the Patients table in Figure 1 with quasi-identifier ⟨Birthdate, Sex, Zipcode⟩. The first iteration of Incognito finds that T is k-anonymous with respect to $\langle B_0\rangle$, $\langle S_0\rangle$, and $\langle Z_0\rangle$, the un-generalized domains of Birthdate, Sex, and Zipcode respectively. The second iteration performs three breadth-first searches to determine the k-anonymity status of T with respect to the multi-attribute generalizations of ⟨Birthdate, Sex⟩, ⟨Birthdate, Zipcode⟩, and ⟨Sex, Zipcode⟩. Figure 5 (a, b ,c) shows these searches. In (a), for example, the algorithm first generates the frequency set of T with respect to $\langle S_0, Z_0\rangle$, and finds that 2-anonymity is not satisfied. It then rolls up this frequency set to generate the frequency sets with respect to $\langle S_1, Z_0\rangle$ and $\langle S_0, Z_1\rangle$, and uses these results to check k-anonymity. In this example, Patients is 2-anonymous with respect to $\langle S_1, Z_0\rangle$. Therefore, all generalizations of $\langle S_1, Z_0\rangle$ (ie., $\langle S_1, Z_1\rangle$, $\langle S_1, Z_2\rangle$) must also be 2-anonymous given the generalization property, so they are marked. Patients is not 2-anonymous with respect to $\langle S_0, Z_1\rangle$, so the algorithm then checks the 2-anonymity status of only $\langle S_0, Z_2\rangle$. Finding that Patients is 2-anonymous with respect to this attribute set, the search is complete.

Lemma *The breadth-first search of the graph defined by C_i and E_i determines the k-anonymity status of T with respect to all i-attribute generalizations in C_i.*

Proof During the breadth-first search, the k-anonymity status of each node n is determined in one of two ways. Either the frequency of n is computed with respect to T, and k-anonymity checked, or n is the (direct or implied) multi-attribute generalization of some node that is determined to be k-anonymous. In this case, we know n is k-anonymous based on the generalization property. □

3.1.2 Graph Generation

We implemented each multi-attribute generalization graph as two relational tables: one for the nodes and one for the edges. Figure 6 shows the relational representation of the

Nodes

ID	dim_1	$index_1$	dim_2	$index_2$
1	Sex	0	Zipcode	0
2	Sex	1	Zipcode	0
3	Sex	0	Zipcode	1
4	Sex	1	Zipcode	1
5	Sex	0	Zipcode	2
6	Sex	1	Zipcode	2

Edges

Start	End
1	2
1	3
2	4
3	4
3	5
4	6
5	6

Figure 6: Representation of sample generalization lattice (Figure 3(a)) as nodes and edges relations

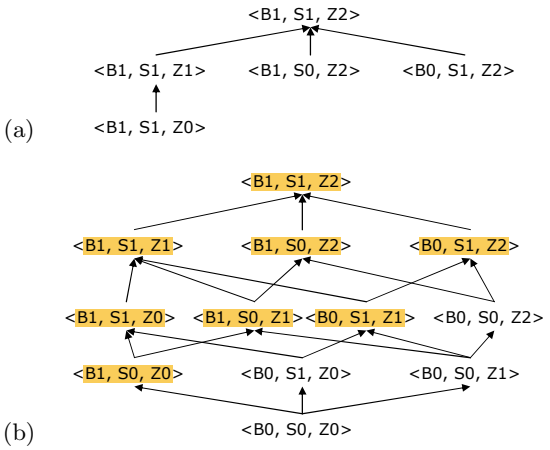

Figure 7: (a) The 3-attribute graph generated from 2-attribute results in Figure 5 and (b) The 3-attribute lattice that would have been searched without a priori pruning

lattice depicted in Figure 3 (a). Notice that each node is assigned a unique identifier (ID).

The graph generation component consists of three main phases. First, we have a join phase and a prune phase for generating the set of candidate nodes C_i with respect to which T could potentially be k-anonymous given previous iterations; these phases are similar to those described in [2]. The final phase is edge generation, through which the direct multi-attribute generalization relationships among candidate nodes are constructed. Costruisco gli archi tra i nodi candidati

The join phase creates a superset of C_i based on S_{i-1}. The join query is as follows, and assumes some arbitrary ordering assigned to the dimensions. As in [2], this ordering is intended purely to avoid generating duplicates.

INSERT INTO C_i(dim_1, $index_1$,...,
  dim_i, $index_i$, $parent_1$, $parent_2$)
SELECT p.dim_1, p.$index_1$,..., p.dim_{i-1}, p.$index_{i-1}$,
  q.dim_{i-1}, q.$index_{i-1}$, p.ID, q.ID
FROM S_{i-1} p, S_{i-1} q
WHERE p.dim_1 = q.dim_1 ∧ p.$index_1$ = q.$index_1$ ∧ ...
  ∧ p.dim_{i-2} = q.dim_{i-2} ∧ p.$index_{i-2}$ = q.$index_{i-2}$
  ∧ p.dim_{i-1} < q.dim_{i-1}

The result of the join phase may include some nodes with subsets not in S_{i-1}, and during the prune phase, we use a hash tree structure similar to that described in [2] to remove these nodes from C_i.

Once C_i has been determined, it is necessary to construct the set of edges connecting the nodes (E_i). Notice that during the join phase we tracked the unique IDs of the two nodes in C_{i-1} that were combined to produce each node in C_i ($parent_1$ and $parent_2$). forall c ∈ Ci do
  forall (i-1)-subsets s of c do
    if s ∉ Si-1 then
      delete c from Ci

E_i is constructed using C_i and E_{i-1} based on some simple observations. Consider two nodes A and $B \in C_i$. We observe that if there exists a generalization relationship between the first parent of A and the first parent of B, and the second parent of B is either equal to or a generalization of the second parent of A, then B is a generalization of A. In some cases, the resulting generalization relationships may be implied, but they may only be separated by a single node. We remove these implied generalization relationships explicitly from the set of edges. The edge generation process can be expressed in SQL as follows:

INSERT INTO E_i(start, end)
WITH CandidateEdges (start, end) AS (
  SELECT p.ID, q.ID
  FROM C_i p, C_i q, E_{i-1} e, E_{i-1} f
  WHERE (e.start = p.$parent_1$ ∧ e.end = q.$parent_1$
    ∧ f.start = p.$parent_2$ ∧ f.end = q.$parent_2$)
  ∨ (e.start = p.$parent_1$ ∧ e.end = q.$parent_1$
    ∧ p.$parent_2$ = q.$parent_2$)
  ∨ (e.start = p.$parent_2$ ∧ e.end = q.$parent_2$
    ∧ p.$parent_1$ = q.$parent_1$)
)
SELECT D.start, D.end
FROM CandidateEdges D
EXCEPT
SELECT D_1.start, D_2.end
FROM CandidateEdges D_1, CandidateEdges D_2
WHERE D_1.end = D_2.start

Di tutti gli archi candidati tolgo quelli che danno "implied"...cioè roba del tipo 1->2->4 che sarebbe l'implied 1->4

Example 3.2 Consider again the Patients table in Figure 1 with quasi-identifier ⟨Birthdate, Sex, Zipcode⟩. Suppose the results of the second-iteration graph search are those shown in the final steps of Figure 5 (a, b, c). Figure 7 (a) shows the 3-attribute graph resulting from the join, prune, and edge generation procedueres applied to the 2-attribute graphs. In many cases, the resulting graph is smaller than the lattice that would have been produced without a priori pruning. For example, see Figure 7(b).

3.2 Soundness and Completeness

As mentioned previously, Incognito generates the set of all possible k-anonymous full-domain generalizations. For example, consider the generalization lattice in Figure 7 (a). If relation T is k-anonymous with respect to $\langle B_1, S_1, Z_0\rangle$, then this generalization will be among those produced as the result of Incognito. If T is not k-anonymous with respect to this generalization, then it will not be in the result set. In this section, we sketch out a proof of soundness and completeness. The full proof is omitted due to space constraints.

Theorem *Basic Incognito is sound and complete for producing k-anonymous full-domain generalizations.*

Proof Sketch Consider a table T and its quasi-identifier attribute set Q. Let Q' denote the set of multi-attribute domain generalizations of Q. Incognito determines the k-anonymity status of each generalization q in Q' in one of three ways:

1. The k-anonymity of T with respect to some subset of q is checked explicitly and found not to be k-anonymous. In this case, we know by the subset property that T is not k-anonymous with respect to q.
2. The k-anonymity of T with respect to some quasi-identifier subset p is checked, and found not to be k-anonymous, and some subset of q is a (multi-attribute) generalization of p. In this case, we know based on the

```
Input: A table T to be k-anonymized, a set Q of n quasi-identifier attributes, and a set of dimension tables (one for each quasi-identifier in Q)
Output: The set of k-anonymous full-domain generalizations of T
  C_1 = {Nodes in the domain generalization hierarchies of attributes in Q}
  E_1 = {Edges in the domain generalization hierarchies of attributes in Q}
  queue = an empty queue
  for i = 1 to n do
    //C_i and E_i define a graph of generalizations
    S_i = copy of C_i
    {roots} = {all nodes ∈ C_i with no edge ∈ E_i directed to them}
    Insert {roots} into queue, keeping queue sorted by height
    while queue is not empty do
      node = Remove first item from queue
      if node is not marked then
        if node is a root then
          frequencySet = Compute frequency set of T with respect to attributes of node using T.
        else
          frequencySet = Compute frequency set of T with respect to attributes of node using parent's frequency set.
        end if
        Use frequencySet to check k-anonymity with respect to attributes of node
        if T is k-anonymous with respect to attributes of node then
          Mark all direct generalizations of node
        else
          Delete node from S_i
          Insert direct generalizations of node into queue, keeping queue ordered by height
        end if
      end if
    end while
    C_{i+1}, E_{i+1} = GraphGeneration(S_i, E_i)
  end for
  return Projection of attributes of S_n onto T and dimension tables
```

Figure 8: Basic Incognito algorithm.

generalization and subset properties that q is not k-anonymous.

3. Generalization q is checked explicitly, and it is determined that T is k-anonymous with respect to q. □

Soundness and completeness is a key distinction between Incognito and Samarati's binary search algorithm [14]. Incognito will find all k-anonymous full-domain generalizations, from which the "minimal" may be chosen according to any criteria. The binary search is guaranteed to find only a single k-anonymous full-domain generalization, which is minimal only according to the specific definition described in Section 2.1. Bottom-up breadth-first search is also sound and complete if run exhaustively.

3.3 Algorithm Optimizations

3.3.1 Super-roots

A candidate node n in C_i is a "root" if there is no generalization edge in E_i directed from another node in C_i to n. During each iteration of Incognito, the database is scanned once per root to generate its frequency set. Because of the a priori pruning optimization, however, we are not guaranteed that the candidate nodes at each iteration will form lattices, so some of these roots might come from the same "family" (generalizations of the same quasi-identifier subset). In this case, we observed that it was more efficient to group roots according to family, and then scan the database once, generating the frequency set corresponding to the least upper bound of each group (the "super-root"). We refer to the basic algorithm, augmented by this optimization, as **Super-roots Incognito**.

For example, in Figure 7(a), $\langle B_1, S_1, Z_0\rangle$, $\langle B_1, S_0, Z_2\rangle$, and $\langle B_0, S_1, Z_2\rangle$ are all roots of the 3-attribute graph, but they come from the same family. Rather than scanning the database once for each of these roots, Super-roots Incognito would first compute the frequency set of Patients with respect to $\langle B_0, S_0, Z_0\rangle$, and would then use this to compute the frequency set for each of these roots.

3.3.2 Bottom-Up Pre-computation

Even Super-roots Incognito scans T once per subset of the quasi-identifier in order to generate the necessary frequency sets. This drawback is fundamental to the a priori optimization, where single-attribute subsets are processed first. For example, we are not able to use the frequency set of T with respect to ⟨Zipcode⟩ to generate the frequency set of T with respect to ⟨Sex, Zipcode⟩. On the other hand, in the context of computing the data cube, these group-by queries would be processed in the opposite order [8], and rather than re-scanning the database, we could compute the frequency set of T with respect to ⟨Zipcode⟩ by simply rolling up the frequency set with respect to ⟨Sex, Zipcode⟩.

Based on this observation, and with the hope of having it both ways, we considered first generating the frequency sets of T with respect to all subsets of the quasi-identifier at the lowest level of generalization. These frequency sets can be computed using bottom-up aggregation, similar to that used for computing the data cube. When Incognito is run, it processes the smallest subsets first, as before, but these zero-level frequency sets can used on each iteration instead of scanning the entire database. We refer to the basic algorithm that first pre-computes the zero-generalization frequency sets as **Cube Incognito**.

4. PERFORMANCE ANALYSIS

To assess the performance of Incognito, we performed a number of experiments using real-world data. We evaluated Basic Incognito, Super-roots Incognito, and Cube Incognito

Adults

	Attribute	Distinct Values	Generalizations
1	Age	74	5-, 10-, 20-year ranges(4)
2	Gender	2	Suppression(1)
3	Race	5	Suppression(1)
4	Marital Status	7	Taxonomy tree(2)
5	Education	16	Taxonomy tree(3)
6	Native country	41	Taxonomy tree(2)
7	Work Class	7	Taxonomy tree(2)
8	Occupation	14	Taxonomy tree(2)
9	Salary class	2	Suppression(1)

Lands End

	Attribute	Distinct Values	Generalizations
1	Zipcode	31953	Round each digit(5)
2	Order date	320	Taxonomy Tree(3)
3	Gender	2	Suppression(1)
4	Style	1509	Suppression(1)
5	Price	346	Round each digit(4)
6	Quantity	1	Suppression(1)
7	Cost	1412	Round each digit(4)
8	Shipment	2	Suppression(1)

Figure 9: Descriptions of the Adults and Lands End Databases used for performance experiments

against previous minimal full-domain generalization algorithms, including Samarati's Binary search [14], Bottom-up search (without rollup), and the Bottom-up search (with rollup) described in Section 2.2. Both bottom-up variations were run exhaustively to produce all k-anonymous generalizations. Throughout our experiments, we found that the Incognito algorithms uniformly outperformed the others.

The databases used in our performance experiments represent the largest-scale evaluation that we are aware of for minimal full-domain k-anonymization. Previously, full-domain k-anonymity techniques were demonstrated using only a toy database of 265 records [15]. No experimental evaluation was provided for binary search [14]. The genetic algorithm in [11] was evaluated using a somewhat larger database, but this algorithm does not guarantee minimality.

4.1 Experimental Data and Setup

We evaluated the algorithms using two databases. The first was based on the Adults database from the UC Irvine Machine Learning Repository [5], which is comprised of data from the US Census. We used a configuration similar to that in [11], using nine of the attributes, all of which were considered as part of the quasi-identifier, and eliminating records with unknown values. The resulting table contained 45,222 records (5.5 MB). The second database was much larger, and contained point-of-sale information from Lands End Corporation. The database schema included eight quasi-identifier attributes, and the database contained 4,591,581 records (268 MB).

The experimental databases are described in Figure 9, which lists the number of unique values for each attribute, and gives a brief description of the associated generalizations. In some cases, these were based on a categorical taxonomy tree, and in other cases they were based on rounding numeric values or simple suppression. The height of each domain generalization hierarchy is listed in parentheses. We implemented the generalization dimensions as a relational star-schema, materializing the value generalizations in the dimension tables.

We implemented the three Incognito variations, Samarati's binary search[2], and the two variations of bottom-up breadth-first search using Java and IBM DB2. All frequency sets were implemented as un-logged temporary tables. All experiments were run on a dual-processor AMD Athlon 1.5 GHz machine with 2 GB physical memory. The software included Microsoft Windows Server 2003 and DB2 Enterprise Server Edition Version 8.1.2. The buffer pool size was set to 256 MB. Because of the computational intensity of the algorithms, each experiment was run 2-3 times, flushing the buffer pool and system memory between runs. We report the average cold performance numbers, and the numbers were quite consistent across runs.

[2]We implemented the k-anonymity check as a group-by query over the star schema. Samarati suggests an alternative approach whereby a matrix of distance vectors is constructed between unique tuples [14]. However, we found constructing this matrix prohibitively expensive for large databases.

4.2 Experimental Results

The complexity of each of the algorithms, including Incognito, is ultimately exponential in the size of the quasi-identifier. However, we found that in practice the rollup and a priori optimizations go a long way in speeding up performance.

Figure 10 shows the execution time of Incognito and previous algorithms on the experimental databases for varied quasi-identifier size ($k = 2, 10$). We began with the first three quasi-identifier attributes from each schema (Figure 9), and added additional attributes in the order they appear in these lists.

We found that Incognito substantially outperformed Binary Search on both databases, despite the fact that Incognito generates all possible k-anonymous generalizations, while Binary Search finds only one. Incognito also out-performed Bottom-up search (with and without rollup). The performance of Binary Search varied based on the pattern of the search performed. Bottom-up search was more consistent, but overall both were slower than Incognito.

4.2.1 *Effects of Rollup and the A Priori Optimization*

As mentioned previously, we observed that the bottom-up breadth-first search can be improved by using rollup aggregation, an idea incorporated into the Incognito algorithm. To gauge the effectiveness of this optimization, we compared the version of the bottom-up algorithm with rollup to the version without rollup. Figure 10 shows that bottom-up performs substantially better on the Adults database when it takes advantage of rollup.

We also found that the a priori optimization, the other key component of Incognito, went a long way in helping to prune the space of nodes searched, in turn improving performance. In particular, the number of nodes searched by Incognito was much smaller than the number searched by bottom-up, and the size of the frequency sets computed for each of these nodes is generally smaller. For the Adults database, k=2, and varied quasi-identifier size(QID), the number of nodes searched is shown below.

QID size	Bottom-Up	Incognito
3	14	14
4	47	35
5	206	103
6	680	246
7	2088	664
8	6366	1778
9	12818	4307

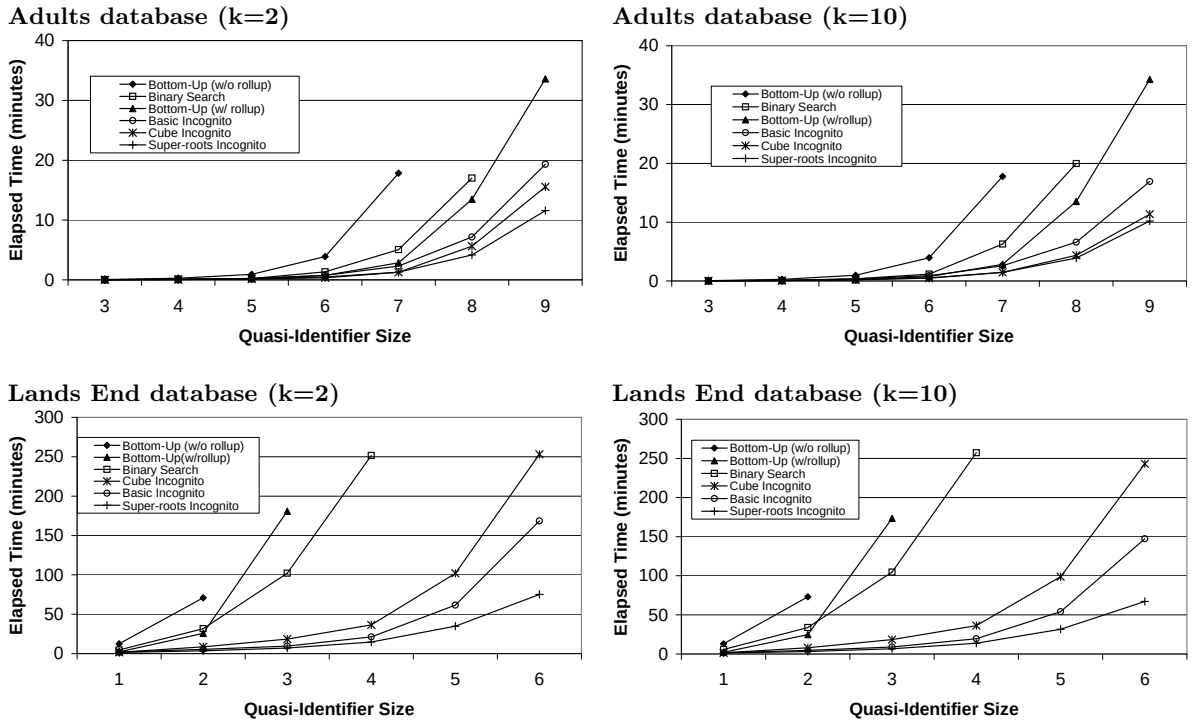

Figure 10: Performance by varied quasi-identifier size on Adults and Lands End databases for k = 2, 10

As the size of k increases, more generalizations are pruned as part of smaller subsets, and the performance of Incognito improves. For example, Figure 11 compares Incognito and the other algorithms as k increases, and Incognito trends downward. Because of the search pattern, binary search is more erratic.

4.2.2 Effects of Super-Roots

The super-roots optimization was very effective in reducing the Incognito runtime because it substantially reduced access to the original data, instead computing many of the frequency sets from other frequency sets. By creating a single super-root frequency set (which requires a single scan), in practice we eliminate up to 4 or 5 scans of the data. This performance gain is most pronounced in the larger Lands End database (See Figure 10).

4.2.3 Effects of Pre-computation and Materialization

Figure 10 shows the cost of Cube Incognito, which includes both the cost of building the zero-generalization frequency sets from the bottom-up and the cost of anonymization using these frequency sets. Figure 12 breaks down this cost. Because the Adults database is small, it is relatively inexpensive to build the zero-generalization frequency sets, and the Cube Incognito algorithm beats Basic Incognito. On the larger Lands End database, Cube Incognito is slower than the basic variation. However, the marginal cost of anonymization is faster than Basic Incognito once the zero-generalization frequency sets have been materialized.

Strategic materialization is an important future direction. In many cases, especially when many quasi-identifier attributes are considered, the zero-generalization frequency sets can be quite large. Because iterations of Incognito are actually likely to need frequency sets at higher levels of generalization, we suspect that materializing frequency sets at multiple levels of generalization is likely to provide substantial performance improvement.

5. TAXONOMY OF K-ANONYMIZATION MODELS

The algorithms considered throughout this paper have focused on finding k-anonymous full-domain generalizations. However, a wide variety of other anonymization models have been proposed, and it is our hope that a side-by-side comparison will ultimately lead to a suite of algorithms allowing us to make explicit tradeoffs between performance and flexibility. In this section, we distill these proposed models into a single taxonomy, and in doing so, we also expose several promising new models. From our perspective, the proposed models can be roughly categorized according to three main criteria:

- ***Generalization vs. Suppression*** Some models only consider suppressing data items altogether, while others consider generalizing values through some number of intermediate states.
- ***Global vs. Local Recoding*** Some models seek to anonymize a given database by mapping the values in the domains of quasi-identifier attributes to modified values. Following the terminology in [20], we refer to this as *global recoding*. Alternatively, some models modify individual instances of data items, using *local recoding*.
- ***Hierarchy-Based vs. Partition-Based*** Generalization models can be categorized into two main subgroups. First, there are those that use fixed value-generalization hierarchies, as described in Section 2. Also, there is a class of models that consider the domain of an attribute to be a totally-ordered set, and define generalizations by partitioning the set into disjoint ranges [3, 11]. Generally, the ordered-set partitioning models are most suitable for continuous- or numeric-valued data, and the hierarchy-based models are better-suited for categorical values.

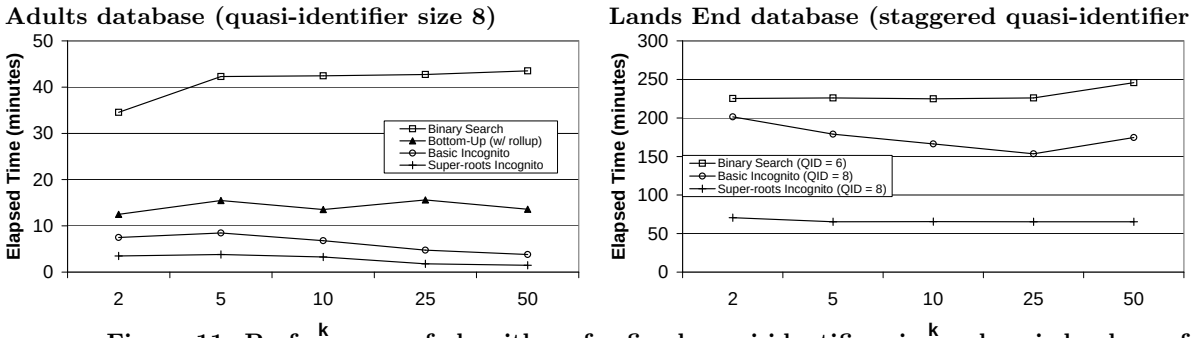

Figure 11: Performance of algorithms for fixed quasi-identifier size and varied values of k

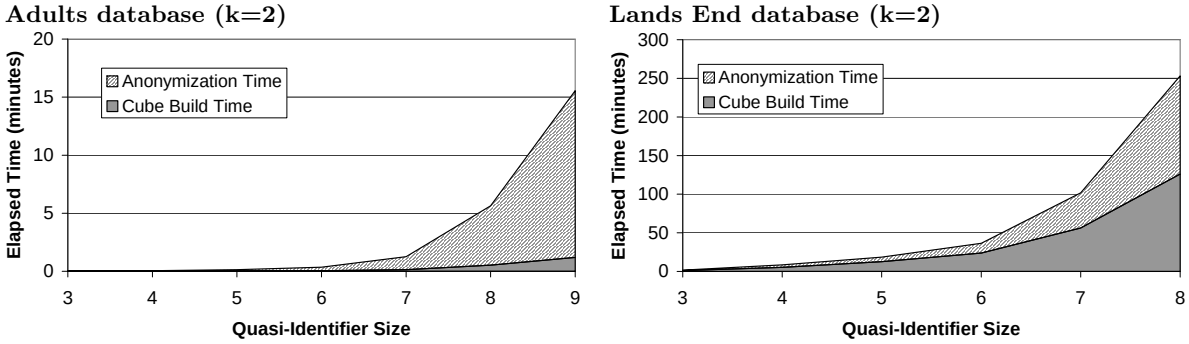

Figure 12: Combined cost of building zero-generalization cube and anonymization

These models provide varying amounts of flexibility in choosing what data is released, and at what level of generality. Some of the models encompass a space of anonymizations that others do not. For this reason, what is the optimal anonymization due to one scheme may be "better" than the optimal anonymization due to another scheme. In the following sections, we provide an overview of these different approaches.

5.1 Global Recoding Models

Full-domain generalization (Section 2.1) is one example of global-recoding. However, more flexible models have been proposed, and there are undoubtedly a number of other possibilities. Most of these models have focused on recoding the domains of the quasi-identifier attributes individually, which we term *single-dimension recoding*. A single-dimension recoding defines some function $\phi_i : D_{Q_i} \rightarrow D'$ *for each attribute* Q_i of the quasi-identifier. A generalization V of T is obtained by applying each ϕ_i to the values of Q_i in each tuple of T. Models of this variety have been used in a number of anonymization schemes [3, 7, 11, 19].

Alternatively, it is possible to construct an anonymization model that recodes the *multi-attribute domain* of the quasi-identifier, rather than recoding the domain of each attribute independently. We call this class of models *multi-dimension recoding*. A multi-dimension recoding is defined by a *single* function $\phi : \langle D_{Q_1} \times ... \times D_{Q_n} \rangle \rightarrow D'$, which is used to recode the domain of value vectors associated with the set of quasi-identifier attributes. Generalization V of T is obtained by applying ϕ to the vector of quasi-identifier values in each tuple of T. Recent results suggest that multi-dimension models might produce better anonymizations than their single-dimension counterparts [12].

5.1.1 Hierarchy-Based Single-Dimension Recoding

A single-dimension recoding defines some function ϕ_i for each attribute Q_i of the quasi-identifier. The single-dimension recoding models differ in how this function is defined. We first consider several hierarchy-based models.

The **Full-domain Generalization** model defines ϕ_i to map every value in D_{Q_i} to a generalized value at the same level of the value generalization hierarchy. There is also a special case of full-domain generalization, which we call **Attribute Suppression** [13]. If $*$ denotes a suppressed value, ϕ_i must either map every value in D_{Q_i} to its unmodified value q_i, or it must map every value in D_{Q_i} to $*$.

More flexible hierarchy-based models relax the requirements of full-domain generalization. One such model, which we will call **Single-Dimension Full-Subtree Recoding**, was used for categorical data in [11]. Recall from Section 2 the (single-dimension) many-to-one value generalization function ($\gamma : D_i \rightarrow D_j$), that mapped values from domain D_i into the more general domain D_j. Under the full-subtree recoding model, each ϕ_i is defined such that for each value $q \in D_{Q_i}$, $\phi_i(q) = q$ or $\phi_i(q) \in \gamma^+(q)$. If ϕ_i maps any q to some value $g \in \gamma^+(q)$, then it must map to g all values in the subtree of the value generalization hierarchy rooted at g. For example, consider the value generalization hierarchy for Zipcode in Figure 2 (b). If a single-dimension full-subtree recoding maps 53715 to 5371*, then it must also map 53710 to 5371*. If it maps any value to 537**, then it must map the entire subtree rooted at 537** to 537**.

We might consider further relaxing the requirements to obtain an even more flexible model that we call **Unrestricted Single-Dimension Recoding**.[3] Under this model, the only restriction on recoding function ϕ_i is that for each value q in the domain of Q_i, $\phi_i(q) = q$ or $\phi_i(q) \in \gamma^+(q)$.

[3]There are situations where the application of this model may lead to certain types of inference, for example mapping the value "Male" to "Person" while leaving "Female" ungeneralized. Nonetheless, we include it as a possibility.

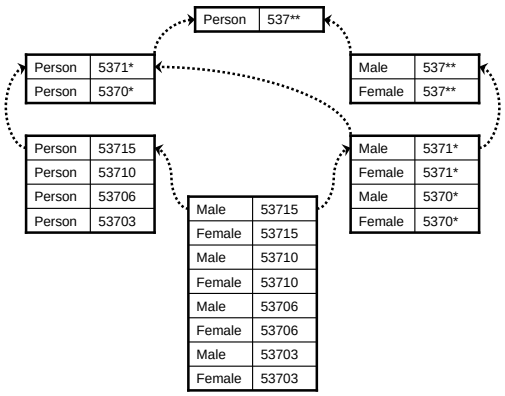

Figure 13: Multi-dimensional value generalization lattice for the Sex and Zipcode attributes.

5.1.2 Partition-Based Single-Dimension Recoding

Single-dimension recoding models have also been proposed that use an ordered-set partitioning approach [3, 11]. Under the **Single-Dimension Ordered-Set Partitioning** model, we assume that the domain of each attribute D_{Q_i} can be represented as a totally-ordered set. ϕ_i maps the set of values in D_{Q_i} into a set of disjoint intervals that cover D_{Q_i}. For example, consider $D_{Zipcode} = \langle 53703, 53706, 53710, 53715\rangle$ as an ordered-set, and a possible set of intervals: $\langle 53703, 53705, 53710\rangle$, $\langle 53715\rangle$. Under this model, $\phi_{zipcode}(53705) = [53703\text{-}53710]$, and $\phi_{zipcode}(53710) = [53710]$.

5.1.3 Hierarchy-Based Multi-Dimension Recoding

A multi-dimension recoding is defined by a single function $\phi : \langle D_{Q_1} \times ... \times D_{Q_n}\rangle \rightarrow D'$, which is used to recode the domain of n-vectors corresponding to the set of quasi-identifier attributes. The hierarchy-based models for single-dimension recoding are readily extended to multiple dimensions.

In order to define these models, we first extend the idea of a value generalization function to multiple dimensions. Let the *multi-attribute value generalization function* $\gamma : \langle D_{A_1} \times ... \times D_{A_n}\rangle \rightarrow \langle D_{B_1} \times ... \times D_{B_n}\rangle$ be a many-to-one function, and let $\langle D_{A_1}, ..., D_{A_n}\rangle <_D \langle D_{B_1}, ..., D_{B_n}\rangle$ denote the domain generalization relationship. (Again, we use the notation γ^+ to denote the composition of one or more multi-attribute value generalization functions.) Like the single-dimension case, the multi-attribute value generalization relationships associated with a multi-attribute domain induce a *multi-attribute value generalization lattice.* The directed edges in this lattice represent direct value generalizations (based on γ), while indirect paths represent implied generalizations (based on γ^+). The *sub-graph* rooted at node n in the lattice is the set of all nodes and edges encountered by recursively traversing all edges backwards from n.

For example, Figure 13 depicts the multi-attribute value generalization lattice for Sex and Zipcode. (For clarity, not all value generalization relationships are shown. The dotted arrows indicate the direct and implied multi-attribute value generalizations of $\langle$Male, 53715$\rangle$.) In this example, the subgraph rooted at $\langle$Person, 5371*$\rangle$ contains nodes $\langle$Person, 53715$\rangle$, $\langle$Person, 53710$\rangle$, $\langle$Male, 5371*$\rangle$, $\langle$Female, 5371*$\rangle$, $\langle$Male, 53715$\rangle$, $\langle$Female, 53715$\rangle$, $\langle$Male, 53710$\rangle$, and $\langle$Female, 53710$\rangle$.

Several new recoding models can be defined in terms of the multi-dimensional value generalization lattice. One such model, **Multi-Dimension Full-Subgraph Recoding**, is an extension of single-dimension full-subtree recoding. Under this model, multi-dimension recoding function ϕ is defined such that for each tuple $\langle q_1, ..., q_n\rangle$ in the multi-attribute domain of the quasi-identifier attribute set, $\phi(\langle q_1, ..., q_n\rangle) = \langle q_1, ..., q_n\rangle$ or $\phi(\langle q_1, ..., q_n\rangle) \in \gamma^+(\langle q_1, ..., q_n\rangle)$. If ϕ maps any $\langle q_1, ..., q_n\rangle$ to some $\langle g_1, ..., g_n\rangle \in \gamma^+(\langle q_1, ..., q_n\rangle)$, then it must map to $\langle g_1, ..., g_n\rangle$ all values in the subgraph of the (multi-dimensional) value generalization lattice rooted at $\langle g_1, ..., g_n\rangle$. For example, suppose a multi-dimension full-subgraph value generalization mapped pair $\langle$Male, 53715$\rangle$, to $\langle$Person, 5371*$\rangle$. In this case, it must also map pairs $\langle$Female, 53715$\rangle$, $\langle$Male, 53710$\rangle$, and $\langle$Female, 53710$\rangle$ to value $\langle$Person, 5371*$\rangle$

Like the single-dimension model, we can relax the requirements to obtain a more flexible model, which we call the **Unrestricted Multi-Dimension Recoding**. Such a recoding is defined by a single function ϕ such that, for each tuple $\langle q_1, ..., q_n\rangle$ in the multi-attribute domain of the quasi-identifier attribute set, $\phi(\langle q_1, ..., q_n\rangle) = \langle q_1, ..., q_n\rangle$ or $\phi(\langle q_1, ..., q_n\rangle) \in \gamma^+(\langle q_1, ..., q_n\rangle)$.

5.1.4 Partition-Based Multi-Dimension Recoding

The ordered-set partitioning model can also be extended to multiple dimensions. Again, consider the domain of each attribute D_{Q_i} to be an ordered set of values, and let a *multidimensional interval* be defined by a pair of points $(p_1, ..., p_n)$, $(v_1, ..., v_n)$ in the multi-dimensional space such that $\forall i, p_i \leq v_i$. The **Multi-Dimension Ordered-Set Partitioning** model defines a set of disjoint multi-dimensional intervals that cover the domain $D_{Q_1} \times ... \times D_{Q_n}$. Recoding function ϕ maps each tuple $(q_1, ..., q_n) \in D_{Q_1} \times ... \times D_{Q_n}$ to a multi-dimensional interval in the cover such that $\forall i, p_i \leq q_i \leq v_i$.

5.2 Local Recoding Models

Consider a relation T with quasi-identifier attribute set Q. The local recoding models produce k-anonymization V of T by defining a bijective[4] function ϕ from each tuple $\langle a_1, ..., n_n\rangle$ in the projection of Q on T to some new tuple $\langle a'_1, ..., a'_n\rangle$. V is defined by replacing each tuple t in the projection of Q on T with $\phi(t)$. Two main local recoding models have been proposed in the literature. The first (**Cell Suppression**) produces V by suppressing individual cells of T [1, 13, 20]. The second (**Cell Generalization**) maps individual cells to their value generalizations using a hierarchy-based generalization model [17]. It should be noted that local recoding models are likely to be more powerful than global recoding.

6. RELATED WORK

Protecting anonymity when publishing microdata has long been recognized as a problem [20], and there has been much recent work on computing k-anonymizations for this purpose. The μ-Argus system was also implemented to anonymize microdata [10], but considered attribute combinations of only a limited size, so the results were not always guaranteed to be k-anonymous. The generalization and suppression framework employed by Incognito was originally defined by Samarati and Sweeney [15], and Samarati proposed a binary search algorithm for discovering a single minimal full-domain generalization [14]. A greedy heuristic algorithm for

[4]Recall that T and V are multisets.

full-domain generalization ("Datafly") was described in [17], and although the resulting generalization is guaranteed to be k-anonymous, there are no minimality guarantees.

Cost metrics intended to quantify loss of information due to generalization, both for general data use and in the context of data mining applications, were described in [11]. Given such a cost metric, genetic algorithms [11] and simulated annealing [21] have been considered for finding locally minimal anonymizations, using the single-dimension full-subtree recoding model for categorical attributes and the single-dimension ordered-set partitioning model for numeric data. Recently, top-down [7] and bottom-up [19] greedy heuristic algorithms were proposed for producing anonymous data that remains useful for building decision-tree classifiers.

In [3], Bayardo and Agrawal propose a top-down set-enumeration approach for finding an anonymization that is optimal according to a given cost metric, given the single-dimension ordered-set partitioning model. Subsequent work shows that optimal anonymizations under this model may not be as good as anonymizations produced with a multi-dimension variation [12]. Finally, the minimal cell- and attribute-suppression varieties of k-anonymization were proven to be NP-hard (with hardness proofs constructed based on the number of cells and number of attributes, respectively), and $O(klogk)$ [13] and $O(k)$ [1] approximation algorithms were proposed.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we showed that the multi-dimensional data model is a simple and clear way to describe full-domain generalization, and we introduced a class of algorithms that are sound and complete for producing k-anonymous full-domain generalizations using the two key ideas of bottom-up aggregation along generalization dimensions and a priori computation. Although our algorithms (like the previous algorithms) are ultimately exponential in the size of the quasi-identifier, we are able to improve performance substantially, and in some cases perform up to an order of magnitude faster. Through our experiments, we showed that it is feasible to perform minimal full-domain generalization on large databases.

In the future, we believe that the performance of Incognito can be enhanced even more by strategically materializing portions of the data cube, including count aggregates at various points in the dimension hierarchies, much like what was done in [9]. It is also important to perform a more extensive evaluation of the scalability of Incognito and previous algorithms in the case where the original database or the intermediate frequency tables do not fit in main memory.

The second main contribution of this paper is a taxonomy categorizing a number of the possible anonymization models. Building on the ideas of Incognito, we are looking at possible algorithms for the more flexible k-anonymization models described in Section 5.

8. ACKNOWLEDGEMENTS

This work was partially supported by a Microsoft Research graduate fellowship and National Science Foundation Grants IIS-0326328 and IIS-0086002.

Our thanks also to Roberto Bayardo, Chris Kaiserlian, Asher Langton, and three anonymous reviewers for their thoughtful comments on various drafts of this paper.

9. REFERENCES

[1] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *Proc. of the 10th Int'l Conference on Database Theory*, January 2005.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conference on Very Large Databases*, August 1994.

[3] R. Bayardo and R. Agrawal. Data privacy through optimal k-anonymity. In *Proc. of the 21st Int'l Conference on Data Engineering*, April 2005.

[4] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[5] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[6] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26, 1997.

[7] B. Fung, K. Wang, and P. Yu. Top-down specialization for information and privacy preservation. In *Proc. of the 21st Int'l Conference on Data Engineering*, April 2005.

[8] J. Gray, S.Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1), November 1996.

[9] V. Harinarayan, A. Rajaraman, and J. Ullman. Implementing data cubes efficiently. In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, June 1996.

[10] A. Hundepool and L. Willenborg. μ- and τ-ARGUS: Software for statistical disclosure control. In *Proc. of the Third Int'l Seminar on Statistical Confidentiality*, 1996.

[11] V. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, August 2002.

[12] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Multidimensional k-anonymity. Technical Report 1521, University of Wisconsin, 2005.

[13] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proc. of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, June 2004.

[14] P. Samarati. Protecting respondants' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6), November/December 2001.

[15] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical Report SRI-CSL-98-04, SRI Computer Science Laboratory, 1998.

[16] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of the 21st Int'l Conference on Very Large Databases*, August 1995.

[17] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):571–588, 2002.

[18] L. Sweeney. K-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):557–570, 2002.

[19] K. Wang, P. Yu, and S. Chakraborty. Bottom-up generalization: A data mining solution to privacy protection. In *Proc. of the 4th IEEE Internatioal Conference on Data Mining*, November 2004.

[20] L. Willenborg and T. deWaal. *Elements of Statistical Disclosure Control*. Springer Verlag Lecture Notes in Statistics, 2000.

[21] W. Winkler. Using simulated annealing for k-anonymity. Research Report 2002-07, US Census Bureau Statistical Research Division, November 2002.