POkA : Identifying Pareto-Optimal k-Anonymous Nodes in a Domain Hierarchy Lattice

Rinku Dewri, Indrajit Ray, Indrakshi Ray and Darrell Whitley
Department of Computer Science
Colorado State University
Fort Collins, CO 80523, USA
{rinku,indrajit,iray,whitley}@cs.colostate.edu

ABSTRACT

Data generalization is widely used to protect identities and prevent inference of sensitive information during the public release of microdata. The k-anonymity model has been extensively applied in this context. The model seeks a generalization scheme such that every individual becomes indistinguishable from at least $k-1$ other individuals and the loss in information while doing so is kept at a minimum. The search is performed on a domain hierarchy lattice where every node is a vector signifying the level of generalization for each attribute. An effort to understand privacy and data utility trade-offs will require knowing the minimum possible information losses of every possible value of k. However, this can easily lead to an exhaustive evaluation of all nodes in the hierarchy lattice. In this paper, we propose using the concept of Pareto-optimality to obtain the desired trade-off information. A Pareto-optimal generalization is one in which no other generalization can provide a higher value of k without increasing the information loss. We introduce the *Pareto-Optimal k-Anonymization (POkA)* algorithm to traverse the hierarchy lattice and show that the number of node evaluations required to find the Pareto-optimal generalizations can be significantly reduced. Results on a benchmark data set show that the algorithm is capable of identifying all Pareto-optimal nodes by evaluating only 20% of nodes in the lattice.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*security, integrity, and protection*

General Terms

Algorithms

Keywords

k–Anonymity, Microdata disclosure control, Pareto-optimality

1. INTRODUCTION

Privacy violations emanating from the sharing of personal information have raised an important concern in recent years. A study on the year 2000 census data of the U.S. population reveals that 53% of the individuals can be uniquely identified by their gender, city and date of birth; 63% if the ZIP code is known in addition [4]. Such attributes, called quasi-identifiers, can be linked with other publicly available information to enable the re-identification. A classic experiment demonstrating the possibility is presented by Sweeney [15] where she managed to obtain the medical records of the Governor of Massachusetts from a medical insurance data set, containing no explicit identifying information, and a voter's registration list. Much research in information assurance has therefore delved into the protection of respondent's identity. The question to answer is how such information be modified so that the data remains useful for statistical studies while protecting the respondents' identities.

To attend to such privacy concerns, Samarati and Sweeney proposed the concept of data generalization to be used to satisfy a property called k-anonymity [13, 14]. Generalization of data is performed by grouping together data attribute values into a more general one, for example, replacing the age by an age range. A transformed data set of this nature is then said to be k-anonymous if each record in it is same as at least $k-1$ other records. This property implies that any record can be related to at least k underlying individuals.

Data generalization directly affects the utility of the data set. Enforcing higher k values during an anonymization makes more records indistinguishable from each other, thus reducing the statistical utility of the data set. Therefore, the amount of generalization to perform is chosen such that the k-anonymity property is satisfied for a given k while the information loss resulting from it is minimized [1, 3, 6, 7, 8, 9, 10, 12, 16]. Most algorithms identify such a solution from the set of all possible generalizations arranged in the form of a graph, called the domain hierarchy lattice, where every node is a vector signifying the amount of generalization for each quasi-identifier. Efficient traversal of the lattice has been particularly explored so that the number of nodes that undergo evaluation (determining equivalence classes and computing loss) can be reduced.

Although optimal k-anonymization is an important research problem, the applicability of available techniques is severely restricted. This is because the choice of a specific k remains dependent on the data publisher. More often than not, the choice is made in an arbitrary manner. Thus, it is often difficult to justify the choice of a particular value for k. Moreover, finding an optimal generalization with a fixed value of k provides no information on how the loss in information changes if a comparatively higher value of k is chosen instead. Fixating on a specific k value prevents one from determining if a higher k is possible with the same amount of information loss, or even with an information loss that is a tad more. Using existing algorithms for optimal k-anonymization to explore this trade-off would imply trying out all possible values of k. This can become expensive owing to the number of nodes in the domain hierarchy lattice that need to be evaluated in the process. Algorithms meant for optimal k-anonymization with a fixed k, cannot correlate nodes in terms of trade-offs.

In this paper, we propose using the concept of Pareto-optimality to identify generalizations that cannot be simultaneously improved both along the privacy and data utility dimensions. Our principle contribution is the Pareto-Optimal k-Anonymization (POkA) algorithm for identifying Pareto-optimal nodes in a domain hierarchy lattice. POkA utilizes a combination of depth first traversals of the lattice to efficiently move from one Pareto-optimal node to another. Two key properties, namely *height boundary property* and *ground node property*, have been proposed to guide the search in a manner that requires minimal node evaluations while assuring that all Pareto-optimal nodes are identified. Performance analysis on a benchmark data set shows that POkA can prune a large number of sub-optimal nodes from being evaluated and can still obtain all Pareto-optimal nodes.

The rest of the paper is organized as follows. Section 2 presents some related work in k-anonymization. Background concepts are introduced in Section 3. Theoretical groundwork for the algorithm is presented in Section 4. Section 5 describes our algorithm. Performance analysis on a standard benchmark data set is presented in Section 6. Finally, Section 7 concludes the paper.

2. RELATED WORK

Several algorithms have been proposed to find effective k-anonymization. The *μ-argus* algorithm is based on the greedy generalization of infrequently occurring combinations of quasi-identifiers and suppresses outliers to meet the k-anonymity requirement [6]. The *Datafly* approach uses a heuristic method to first generalize the quasi-identifier containing the most number of distinct values [14]. Sequences of quasi-identifier values occurring less than k times are suppressed.

On the more theoretical side, Sweeney propose the *MinGen* algorithm [14] that exhaustively examines all potential generalizations to identify the optimal generalization that minimally satisfies the anonymity requirement. However, the approach is impractical even on modest sized data sets. Meyerson and Williams have recently proposed an approximation algorithm that achieves an anonymization with $O(k \log k)$ of the optimal solution [11].

Samarati proposed an algorithm [12] that identifies all generalizations satisfying k-anonymity. Choice of an optimal generalization can then be made based on certain preference information provided by the data recipient. The approach in *Incognito* [8] is also aimed towards finding all generalizations that satisfy k-anonymity for a given value of k. The basic Incognito algorithm starts with the generalization lattice of a single attribute and performs a modified bottom-up breadth-first search to determine the possible generalized domains of the attribute that satisfy k-anonymity. Thereafter, the generalization lattice is updated to include more and more number of attributes.

A genetic algorithm based formulation is proposed by Iyengar to perform k-anonymization [7]. Bayardo and Agrawal propose a complete search method that iteratively constructs less generalized solutions starting from a completely generalized data set [1]. The algorithm starts with a fully generalized data set and systematically specializes it into one that is minimally k-anonymous. The idea of a *solution cut* is presented by Fung et al. in their approach to top down specialization [3]. A generalization is visualized as a “cut” through the taxonomy tree of each attribute. A cut of a tree is a subset of values in the tree that contains exactly one value on each root-to-leaf path. A solution cut is a cut that satisfies the anonymity requirement.

LeFevre et al. extend the notion of generalizations on attributes to generalization on tuples in the data set [9]. The authors argue that such multidimensional partitioning of the generalization domain show better performance in capturing the underlying multivariate distribution of the attributes, often advantageous in answering queries with predicates on more than just one attribute.

The first known attempt of exploring the privacy and utility trade-offs is undertaken by Dewri et al. [2]. The work focus on multi-objective optimization formulations involving a privacy parameter and an utility metric. A similar concept is presented by Huang and Du in the problem of optimizing randomized response schemes for privacy protection [5].

Our work significantly differs from earlier approaches either in terms of the cardinality of the solutions reported or in terms of the solution methodology where attempts have been made to find trade-off information.

3. PRELIMINARIES

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

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

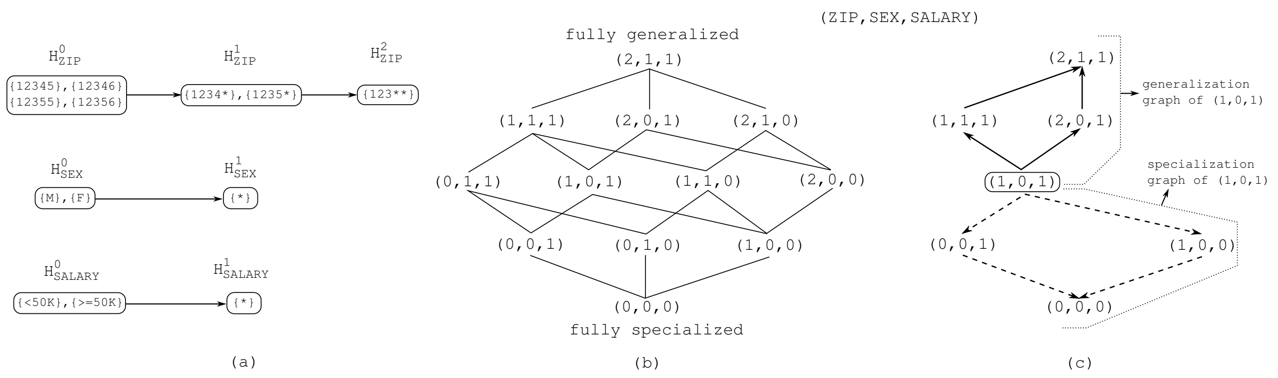

Figure 1: (a) Example domain generalization hierarchies of attributes *ZIP, SEX* and *SALARY*. (b) Domain hierarchy lattice using example DGHs. (c) Generalization and specialization graphs of node $(1,0,1)$ in the lattice.

signified by an integer between 0 and N_j. A generalization level of 0 signifies that all values are distinguishable from each other, while a level of N_j signifies that no two values can be distinguished from each other.

Given a DGH for each quasi-identifier in the data set, a tuple is said to be in an *anonymized* form when a generalization is applied on the attribute values. The anonymized form is represented as follows. Let us assume a tuple $\langle v_1, \ldots, v_s\rangle$ in the data set. Let $(n_1, \ldots, n_s); 0 \le n_i \le N_i$ be the vector representing the generalization level for each attribute; n_i is the level to use in the DGH for attribute A_i. To map the value v_1 to its generalized form we replace it by the index of the set to which it belongs in the generalized domain at level n_1. For example, if $H_1^{n_1} = A_{11}, \ldots, A_{1m}$ and $v_1 \in A_{1p_1}$, then v_1 is replaced by p_1. After performing similar operations for the other attribute values, the tuple is anonymized to the form $\langle p_1, \ldots, p_s\rangle$, p_i being the set index for value v_i in $H_i^{n_i}$. Transforming all tuples in the data set in this manner results in an anonymized data set.

3.1 k-Anonymity

The anonymized tuples of a data set can be grouped together into equivalence classes. Two anonymized tuples $\langle p_1, \ldots, p_s\rangle$ and $\langle q_1, \ldots, q_s\rangle$ belong to the same equivalence class if $p_i = q_i; 1 \le i \le s$. The *k-anonymity* property requires that every such equivalence class should be of size at least k. Table 1 shows an example of this property. With reference to Fig. 1a, *ZIP* is generalized at level 1, *SEX* at level 1, and *SALARY* at level 0. Note that as higher k values are desired, higher generalization levels need to be used for the attributes. In principle, the anonymized tuples have categorical labels instead of the set index. Thus, two anonymized tuples in the same equivalence class have the same labels making them indistinguishable from each other. Higher k values therefore signify higher preservation of privacy for the individuals whose information is represented in the tuples.

3.2 Domain hierarchy lattice

A *domain hierarchy lattice* DHL is a graph with $\prod_i(N_i+1)$ nodes. Every node $(n_1, \ldots, n_s); 0 \le n_i \le N_i$ is a vector of s dimensions where the i^{th} element n_i specifies the generalization level for attribute A_i. The 3-anonymous table in Table 1 corresponds to the node $(1,1,0)$. An edge exists between two nodes $(n_1, \ldots, n_s)$ and $(m_1, \ldots, m_s)$ if and only

ZIP	SEX	SALARY	ZIP	SEX	SALARY
12345	M	<50K	1234*	*	<50K
12346	M	<50K	1234*	*	<50K
12345	F	<50K	1234*	*	<50K
12355	F	≥50K	1235*	*	≥50K
12355	M	≥50K	1235*	*	≥50K
12356	M	≥50K	1235*	*	≥50K

Table 1: 3-anonymous version (right) of a table.

if the vectors differ in exactly one element and the difference is one. To put it formally, $\sum_i |n_i - m_i| = 1$. The node $(0, \ldots, 0)$ (s times) is the *fully specialized* node of the lattice and corresponds to the un-anonymized data set. The node $(N_1, \ldots, N_s)$ is the *fully generalized* node and corresponds to no disclosure of the data. Fig. 1b illustrates these terms. In a typical k-anonymization algorithm, a node is sought in this lattice such that it satisfies k-anonymity and results in minimum information loss for the specified value of k. An exhaustive search is often not desired since evaluation of equivalence class sizes on a moderately sized data set can be computationally intensive. Most algorithms therefore perform some form of pruning of the lattice. Note that the algorithms we refer to here are meant to find optimal generalization levels for a given value of k. Pruning of the lattice when no k value is specified is an unresolved problem until now.

Given a domain hierarchy lattice and a node $(n_1, \ldots, n_s)$, we can also define a *specialization graph* which contains the nodes $(p_1, \ldots, p_s)$ such that $p_i \le n_i; 1 \le i \le s$. Edges between nodes in this graph are drawn similar to as in a DHL. The node $(n_1, \ldots, n_s)$ is called the *root node* of the specialization graph. Along similar lines, we can also define a *generalization graph* if nodes instead satisfy $p_i \ge n_i; 1 \le i \le s$. Fig. 1c highlights the specialization and generalization graph of the node $(1,0,1)$. Intuitively, the specialization graph of a node contains all other nodes which are more specialized in one or more attributes, and in effect induce comparatively smaller k and lower loss. Contrary to that, a generalization graph of a node contains all other nodes which are more generalized in one or more attributes, and in effect induce comparatively larger k and higher loss. Hence, we shall often use the term “move down” and “move up” while traversing a specialization and generalization graph respectively. These two graphs will be used later while performing the search for optimal nodes.

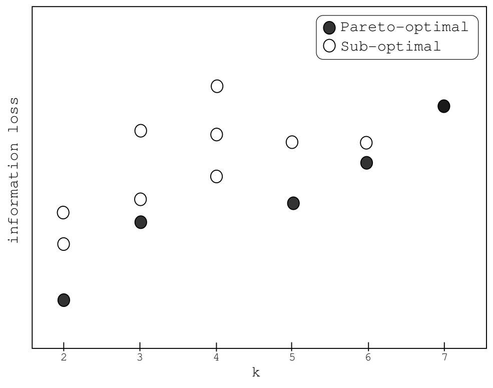

Figure 2: Depiction of Pareto-optimal nodes.

3.3 Pareto-optimal generalization

Let k_N and $\mathcal{L}_N$ signify the k and information loss associated with a node N in the domain hierarchy lattice. The node N is a *Pareto-optimal generalization* if there is no other node M in the lattice such that one of the following two conditions hold.

- $k_M \geq k_N$ and $\mathcal{L}_M < \mathcal{L}_N$, or
- $k_M > k_N$ and $\mathcal{L}_M \leq \mathcal{L}_N$.

If one of the conditions is true, then M is said to dominate N. Therefore, a Pareto-optimal node is one whose k value cannot be improved upon by another node without increasing the information loss, and the information loss at the induced k value is minimal. Note that Pareto-optimal nodes need not always exist at every possible value of k. For example, in Fig. 2, no Pareto-optimal node appears with $k = 4$ since the node with $k = 5$ offers a higher value of k but with lower information loss. As is evident from the figure, trade-off behavior becomes clear when all Pareto-optimal solutions are known. The advantage of finding Pareto-optimal nodes is two fold. First, the minimal information loss at relevant k values is computed. Second, the choice of a particular solution can be based on the change of information loss rather than on arbitrary selection of k. In the figure, the choice of $k = 5$ can perhaps be made under the light that there is not much difference in information loss from the $k = 3$ node. Therefore, our objective is to search the DHL in an efficient manner and identify the Pareto-optimal nodes. Note that this process does not require the specification of a k value by the data publisher. Instead, optimal k values are reported as part of the set of Pareto-optimal generalizations.

4. PARETO SEARCH

The basic search strategy we adopt is to start from an already known Pareto-optimal node and prune nodes that cannot be Pareto-optimal. We shall start with the Pareto-optimal node with the highest possible k value and then use it as a starting step to find the next Pareto-optimal node. The *next Pareto-optimal node* is the one with a k value closest to that of the previous one but with lower loss. Hence, given a Pareto-optimal node $N = (n_1, \ldots, n_s)$, the next Pareto-optimal node M is the one with minimum $(k_N - k_M); k_M < k_N$ and $\mathcal{L}_M < \mathcal{L}_N$. We shall call the node N a *base node.* In Fig. 2, the next Pareto-optimal node for the node with $k = 5$ is the node with $k = 3$. The node M is found by combining a DFS traversal of the specialization graph of N with a DFS traversal of the generalization graph of another node.

The first step in this process is to have a known Pareto-optimal node to begin with – the first base node. This is not difficult since the node $(N_1, \ldots, N_s)$ is bound to be Pareto-optimal. This node induces a k value equal to the size of the data set since all tuples get transformed to the same anonymized form. No other node can produce a k value higher than this. Once the next Pareto-optimal node is found, the process is repeated using this newly found node as the base node.

The next step is to assure that the node M can be reached from the base node N. Note that $k_M < k_N$. Hence, M will not be present in the generalization graph of N. Recall that all nodes in the generalization graph of N will have more generalization in one or more attributes and will result in a higher k value. This observation results in the first level of pruning of the DHL. Given the Pareto-optimal node N, the number of nodes pruned by not searching the generalization graph of N is $\prod_i (N_i - n_i + 1)$. At first glance it may seem that the node M should be somewhere in the specialization graph of N and is hence directly reachable from N. In fact, if the node M is in the specialization graph of N, then it would be the one with the highest k value in the set $\{(n_1, \ldots, n_{j-1}, n_j - 1, n_{j+1}, \ldots, n_s) | 1 \leq j \leq s; n_j \neq 0\}$ – immediate neighbors of N in the graph. Since other nodes in the specialization graph are indeed specializations of a node in this set, they would have k values lower than the highest possible k in these immediate neighbors.

However, there is still a set of nodes that are not present in the specialization graph of N but can potentially include M. These nodes are the ones that can be generated from N by performing generalization in some attributes, specialization in some and no change in others. A positive observation in this context is that the node M can still be reached from N through a node common in the specialization graph of M and N, called a *ground node.*

4.1 Ground nodes

A *ground node* is a node common in the specialization graphs of two nodes in the domain hierarchy lattice. A trivial ground node for any two nodes is the fully specialized node $(0, \ldots, 0)$. Other non-trivial nodes also do exist. Given the nodes N and M, any node in the set $\mathcal{G} = \{(g_1, \ldots, g_s) | 0 \leq g_i \leq \min(n_i, m_i)\}$ is a ground node for N and M. M can then be reached from N by first moving down the specialization graph of N to a ground node and then moving up in the generalization graph of the ground node. The first phase of this process, i.e. moving down to the ground node, is called a *depth search* rooted at N. The phase of moving up from the ground node is called a *height search* rooted at the ground node. Fig. 3 illustrates this process.

Note that although a depth search is essential to find a ground node, nodes traversed in the process need not be

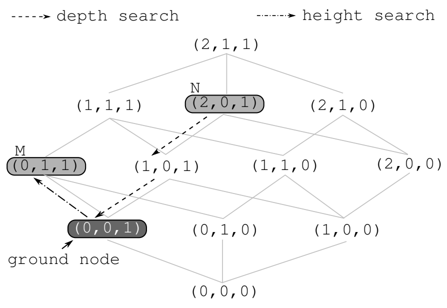

Figure 3: Use of depth search and height search to reach node M from node N through a ground node.

evaluated if they are not immediate neighbors of the base node. This follows from the earlier observation that if M resides in the specialization graph of N, then it will be one of the immediate neighbors. This observation leads to the second level of pruning in node evaluation. However, nodes in the height search are to be evaluated since k values will increase progressively. The only exception are nodes that are also part of the depth search but not immediate neighbors of the base node. A brute force method to perform the searches would mean traversing the specialization graph of N all the way down to the fully specialized node and then traversing the generalization graph of the trivial ground node. Clearly, this is an exhaustive search. The following two sections discuss the theoretical properties that bound the extent of search to be performed in both phases. Efficiently determining the minimum extent to search will further reduce the number of nodes to be evaluated.

4.2 Height search

Height search from a ground node is a DFS traversal of the generalization graph with the ground node as the root. To clarify any ambiguity in language, we say that height search is a height first traversal of the graph. The search is said to be at height h when the current node H is h steps away from the ground node G, i.e $\sum_i |h_i - g_i| = h$. The height h is a dynamically chosen parameter in our approach. The assumption we make is that both k and information loss are non-decreasing quantities when performing more generalization in an attribute. Based on this assumption, the following property states how "high" should the height search proceed before coming back to its parent node.

Height Boundary Property: Let $\mathcal{A} = \{A_1, \ldots, A_s\}$ be a set of attributes. Given a Pareto-optimal node $N = (n_1, \ldots, n_s)$, the next Pareto-optimal node M can be found by a height search of a node in the specialization graph of N, further search up a node H being terminated whenever $k_H \geq k_N$ or $\mathcal{L}_H \geq \mathcal{L}_N$.

Proof: The node to start the height search is a ground node G present in the specialization graph of N. Since M is the next Pareto-optimal node, we have $k_M < k_N$ and $\mathcal{L}_M < \mathcal{L}_N$. Based on the aforementioned assumption, if M is in the generalization graph of G, then it must be reached in a height search before a node H with $k_H \geq k_N$ or $\mathcal{L}_H \geq \mathcal{L}_N$ is reached. Hence, whenever such a node is encountered, we have reached the maximum height along that path. □

The height to search is therefore bounded by the k and loss values of the base node. We shall start with the ground node and choose a child for subsequent search only if it has k and loss lower than that of N. Child nodes that are common to the specialization graph of N (and not immediate neighbors of N) are not evaluated and are always selected. All selected child nodes are subjected to the same evaluation for further search. Exact specification of how the next Pareto-optimal node is identified from height searches is presented in Section 5. The further down the ground node is from the base node, the more will be the number of nodes that will require evaluation in the height search. We therefore need a good estimate of the ground node closest to the next Pareto-optimal node. This is achieved by the depth search.

4.3 Depth search

Depth search from a base node N is a DFS traversal of the specialization graph with N as the root. The *depth* d of the search is the maximum total difference in generalization levels from the base node to an internal node. Therefore, a depth search of depth d implies the traversal of nodes D such that $\sum_i |n_i - d_i| \leq d$. Under this specification, an internal node is searched further only if its maximum total difference in generalization levels from the base node is less than d. Each node at depth d in the specialization graph of N is considered a candidate ground node and is subjected to a height search. Note that the DHL is a graph (not a tree) and hence the general intuition that the number of nodes subjected to height search will increase exponentially with larger d, is not true. In the following, we deduce the depth at which a ground node (not necessarily the closest one) for the next Pareto-optimal node is bound to exist and derive an estimate of the depth to actually search.

Lemma 1: The minimum equivalence class size of a data set with s attributes $\mathcal{A} = \{A_1, \ldots, A_s\}$ is the same as the minimum equivalence class size of a data set with two attributes $\mathcal{A}_1$ and $\mathcal{A}_2$, where $\mathcal{A}_1 = A_j$ and $\mathcal{A}_2 = A_1 \times \ldots \times A_{j-1} \times A_{j+1} \times \ldots \times A_s$.

Proof: The proof follows from the fact that $\mathcal{A}_2$ is simply a concatenated version of the values of the attributes $A_1, \ldots, A_{j-1}, A_{j+1}, \ldots, A_s$. Determining the equivalence class sizes in a data set with s attributes require finding the frequency of occurrence of every possible combination of values for $s-1$ attributes. In the case with two attributes, this step is performed while finding the equivalence class sizes for the attribute $\mathcal{A}_2$. Hence in both cases, a tuple with a particular sequence of values for $A_1, \ldots, A_s$ will belong to an equivalence class of the same size. This means that the minimum equivalence class size will also be same. □

Lemma 2: Let A_1 and A_2 be two attributes with DGH lengths of N_1 and N_2 respectively. If $N = (n_1, n_2)$, such that $0 \leq n_1 \leq N_1$ and $0 \leq n_2 \leq N_2$, is a Pareto-optimal node, then one of $G_1 = (n_1, 0)$ and $G_2 = (0, n_2)$ is a ground node for the next Pareto-optimal node.

Proof: The next Pareto-optimal node M is the one with k_M closest to k_N satisfying the constraints $k_M < k_N$ and $\mathcal{L}_M < \mathcal{L}_N$, i.e. there is no other node M' such that $k_M < k_{M'} < k_N$ and $\mathcal{L}_{M'} < \mathcal{L}_N$. The first set of possibilities are the immediate descendants of N, i.e. $D_1 = (n_1 - 1, n_2)$ or $D_2 = (n_1, n_2 - 1)$. Other descendant nodes of N, i.e. nodes of the form $L = (l_1, l_2); 0 \leq l_i < n_i$, have k values lower than $\max(k_{D_1}, k_{D_2})$ and hence do not satisfy the requirements for the next Pareto-optimal node. If one of D_1 or D_2 is Pareto-optimal then G_2 or G_1 respectively is a ground node for it.

If none of D_1 and D_2 is the next Pareto-optimal node then nodes of the form (l_1, h_2) or (h_1, l_2), where $n_i < h_i \leq N_i$, must dominate them (after satisfying the required constraints) and are likely candidates. Since G_1 is a ground node for any node of the form (h_1, l_2) and G_2 for nodes of the form (l_1, h_2), the result still holds. □

Ground Node Property: Let $\mathcal{A} = \{A_1, \ldots, A_s\}$ be a set of attributes with N_i as the DGH length of A_i. If $N = (n_1, \ldots, n_s); 0 \leq n_i \leq N_i$ is a Pareto-optimal node, then one or more nodes of the set $\mathcal{G}_{best} \cup \mathcal{G}_{worst}$, where $\mathcal{G}_{best} = \{(n_1, \ldots, n_{j-1}, 0, n_{j+1}, \ldots, n_s) | 1 \leq j \leq s\}$ and $\mathcal{G}_{worst} = \{(0, \ldots, n_j, \ldots, 0) | 1 \leq j \leq s\}$, are ground nodes of the next Pareto-optimal node.

Proof: The proof follows from the observation that the problem of finding Pareto-optimal nodes for s properties can be transformed to the case of finding Pareto-optimal nodes for two properties. To do so, we map the attribute set $\mathcal{A}$ to two attributes $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $\mathcal{A}_1 = A_j$ and $\mathcal{A}_2 = A_1 \times \ldots \times A_{j-1} \times A_{j+1} \times \ldots \times A_s$. A tuple $\langle v_1, \ldots, v_s \rangle$ in the original data set is transformed to the form $\langle v_j, v_1; \ldots; v_{j-1}; v_{j+1}; \ldots; v_s \rangle$. Any anonymized version of a tuple is mapped in a similar manner. By Lemma 1, both data sets (generalized or not) will induce the same value of k. The loss metric can be modified so that the loss associated with a tuple in the data set with s attributes is proportional to that of the tuple in the data set with two attributes.

The step that remains is a specification for the DGH of $\mathcal{A}_2$. Nodes in the DGH of $\mathcal{A}_2$ are formed by taking every possible combination of nodes from the DGHs of the $s - 1$ attributes. A node is therefore of the form $\mathcal{N}_j = (n_1; \ldots; n_{j-1}; n_{j+1}; \ldots; n_s); 0 \leq n_i \leq N_i$. The total ordering of these nodes is obtained by first sorting them in ascending order of the k value they induce and then in ascending order of the loss (if k is same for two nodes). This follows the general notion of a DGH where k (and then loss) increases as we step from one node to the next. The domain hierarchy lattice for two attributes contains nodes of the form $(n_j, \mathcal{N}_j)$.

Therefore, there is a bijective mapping between the set of nodes in the domain hierarchy lattice with s attributes and the set of nodes in the domain hierarchy lattice with two attributes. Since, k and loss of two corresponding nodes are also same, any Pareto-optimal node in the lattice with s attributes will also be Pareto-optimal in the lattice with two attributes, and vice versa.

Hence by Lemma 2, given N and a particular value for j, the ground nodes for the next Pareto-optimal node are one or both of $(0, \mathcal{N}_j)$ or $(n_j, 0)$. This translates to nodes of the form $(n_1, \ldots, n_{j-1}, 0, n_{j+1}, \ldots, n_s)$ and $(0, \ldots, n_j, \ldots, 0)$ in the lattice for s attributes. Since the transformation into the case with two attributes can be performed in s different ways $(1 \leq j \leq s)$, the set of such possible ground nodes is given as $\mathcal{G}_{best} = \{(n_1, \ldots, n_{j-1}, 0, n_{j+1}, \ldots, n_s) | 1 \leq j \leq s\}$ and $\mathcal{G}_{worst} = \{(0, \ldots, n_j, \ldots, 0) | 1 \leq j \leq s\}$. □

Assuring that nodes in $\mathcal{G}_{worst}$ are covered while performing a depth search, i.e. $d = \max_j(\sum_i n_i - n_j)$, ensures all possible Pareto-optimal nodes before N (ones with k and loss lower than that of N) will be discovered. This is achieved by allowing the discovery of nodes that require specialization in all (but one) attributes. However, this is often not required if only the immediately next Pareto-optimal node has to be found. Covering nodes in $\mathcal{G}_{best}$, i.e. $d = \max_i(n_i)$, is sufficient for this purpose. Depth search that covers nodes in $\mathcal{G}_{best}$ allows the discovery of nodes that may require full specialization in at most one attribute. To be precise, it allows finding nodes that have full specialization in the attribute with the longest DGH length. A longer DGH is typically specified for an attribute with a bigger domain size. Hence, full specialization in such an attribute has the tendency to induce small equivalence classes, thereby a small value of k. The immediately next Pareto-optimal node is more likely to have a k value closer to k_N.

A good strategy to adopt here is to ensure that attributes which are close to full specialization in N, get a chance to become so while attributes that are far away from being fully specialized are explored in less depth. Consider the nature of nodes that get covered in a depth search of $d_{avg} = \lceil \frac{\sum_i N_i}{s} \rceil$ depth, d_{avg} being the average number of steps required for an attribute to become fully specialized from a fully generalized state. First, depth search to d_{avg} depth ensures that half of the number of attributes will have a chance to become fully specialized. Second, in d_{avg} number of steps, the attributes which are closer to being fully specialized stand a higher chance of becoming so. Third, taking d_{avg} steps allows combination of different levels of specialization for different attributes without making any of them fully specialized (if not already). Performance analysis in Section 6 corroborates that this strategy indeed provides the best balance between exploration of nodes and discovery of Pareto-optimal ones.

5. POkA ALGORITHM

The *Pareto-Optimal k-Anonymization (POkA)* algorithm is an iterative search method to identify the Pareto-optimal generalizations for a given data set. The attributes in the data set that are subjected to generalization (conventionally called quasi-identifiers) are specified and a DGH is defined for every such attribute. The algorithm is iterative because one combination of depth search and height search, and a base node, is required to identify one Pareto-optimal node. The process is applied repeatedly to identify subsequent nodes.

5.1 Handling outliers

Before looking into the implementation details of the algorithm, we must define a strategy to handle outliers in the data set. Outliers in a data set are uncommon combinations of attribute values in a tuple. The existence of such tuples often make the process of k-anonymization difficult. Enforcing a k-anonymity property in the presence of outliers may lead to excessive generalization in the attributes, thereby reducing the utility of the data set. A typical approach to handle outliers is *tuple suppression*. Given a value of k, the tuples which belong to equivalence classes of size less than k are removed from the data set. Impact of the removal is then captured in the information loss measurement. The method is not directly applicable while finding Pareto-optimal generalizations since a k value is not pre-specified. The approach applied here is to use an upper bound on the number of suppressed tuples.

Let η be the maximum number of tuples that is allowed for suppression and R be the total number of tuples in the data set. Consider the sets $E_1, \ldots, E_R$ where E_i contains tuples that are indistinguishable from $i - 1$ other tuples. In other words, all tuples in the set E_i are i-anonymous. Note

Function 1 *HeightSearch(Node P, boolean useNode)*

Input: A node P and a boolean value (*true* or *false*) *useNode*. N is the base node.
Output: $(M, k_M, \mathcal{L}_M)$: M is a node in $\mathcal{GG}(P)$ with the highest k value such that $k_M < k_N$ and $\mathcal{L}_M < \mathcal{L}_N$, or $NULL$ if no such node exists.

```
if [$useNode = true$] $N_o = (P, k_P, \mathcal{L}_P)$
else $N_o = (NULL, 0, 0)$

for every child node $C$ of $P$ in $\mathcal{GG}(P)$ {
  if [$C \in \mathcal{SG}(N)$ and $\sum_i |n_i - c_i| > 1$]
    $Q = HeightSearch(C, false)$
  else {
    $Evaluate(C)$
    if [$k_C < k_N$ and $\mathcal{L}_C < \mathcal{L}_N$]
      $Q = HeightSearch(C, true)$
    else $Q = (NULL, 0, 0)$
  }
  if [$k_Q > k_{N_o}$ or ($k_Q = k_{N_o}$ and $\mathcal{L}_Q < \mathcal{L}_{N_o}$)]
    $N_o = (Q, k_Q, \mathcal{L}_Q)$
}

return $N_o$
```

Function 2 *DepthSearch(Node P)*

Input: A node P in $\mathcal{SG}(N)$, N being the base node.
Output: $(M, k_M, \mathcal{L}_M)$: M is a node reachable by height search of some node in $\mathcal{SG}(P)$ and with the highest k value such that $k_M < k_N$ and $\mathcal{L}_M < \mathcal{L}_N$.

```
$N_o = (NULL, 0, 0)$

for every child node $C$ of $P$ in $\mathcal{SG}(P)$
  if [$\sum_i |n_i - c_i| = d$ ] {
    if [$d = 1$] {
      $Evaluate(C)$
      $Q = HeightSearch(C, true)$
    }
    else $Q = HeightSearch(C, false)$
  }
  else $Q = DepthSearch(C)$
  if [$k_Q > k_{N_o}$ or ($k_Q = k_{N_o}$ and $\mathcal{L}_Q < \mathcal{L}_{N_o}$)]
    $N_o = (Q, k_Q, \mathcal{L}_Q)$
}

return $N_o$
```

Function 3 *POkA()*

Output: The set $\mathcal{P}$ of Pareto-optimal nodes in the DHL

```
$N = (N_1, \ldots, N_s)$
$\mathcal{P} = \{N\}$

while [$k_N > 2$] {
  $(N, k_N, \mathcal{L}_N) = DepthSearch(N)$
  $\mathcal{P} = \mathcal{P} \cup \{N\}$
}

return $\mathcal{P}$
```

that some E_is may be empty sets. If the anonymized data set is to be made k-anonymous, then all tuples in the sets $E_1, \ldots, E_{k-1}$ must be suppressed. Given the hard limit on suppression, this will be possible only if the number of tuples in the union of these sets is less than or equal to η. The same strategy can be applied in a reverse manner. Tuples in all sets $E_1, \ldots, E_j$ are suppressed such that j satisfies the condition $\sum_{i=1}^{j+1} |E_i| > \eta$. The data set is then k-anonymous with $k = j + 1$. The number of tuples suppressed is $|E_1 \cup \ldots \cup E_j|$ and can be accounted for in the loss measurement. This provides us a method to make maximum possible usage of the suppression limit without specifying a k value.

5.2 POkA

Height search and depth search are the two crucial components of POkA. We use the notation $\mathcal{SG}(P)$ and $\mathcal{GG}(P)$ to signify the set of nodes in the specialization graph and generalization graph of a node P respectively. Let N be the base node. k_P and $\mathcal{L}_P$ signify the k and information loss value associated with node P. Further, we assume the existence of a function *Evaluate* which takes as input a node P in the DHL and returns k_P and $\mathcal{L}_P$. These returned values are computed after using the suppression strategy mentioned earlier.

Function 1 presents the pseudo-code for a height search implementation. Height search initiated at a node therefore returns the node in its generalization graph with the highest k and one which satisfies the constraints on the k and loss. Any node that belongs to $\mathcal{SG}(N)$ and is not an immediate neighbor of N is not evaluated and height search proceeds without considering the k and loss of such a node. Otherwise, the node is evaluated to determine if further search is required as determined by the height boundary property. The method is initiated at a candidate ground node decided in the depth search. d signifies the depth to search in the following.

Function 2 shows the pseudo-code for a depth search implementation. The implementation is a simple DFS traversal with a height search being initiated when nodes at depth d are encountered. The best M found in these height searches is translated upwards towards the root of the specialization graph. Hence, the node M returned from a call to *DepthSearch*(N) is the next identified Pareto-optimal node.

POkA starts by a call to *DepthSearch* with the fully generalized node. For every new Pareto-optimal node identified, *DepthSearch* is iteratively called until the Pareto-optimal node with $k \leq 2$ is found. Function 3 shows the pseudo-code of this process.

5.3 Improvements

Node traversal in *DepthSearch* and *HeightSearch* can be further reduced by taking into account the structure of the DHL. Since the structure is that of a graph, nodes in the specialization graph and generalization graph will share nodes as children. This structure results in repeated visits to a node during a depth/height search initiated by multiple parent nodes that share the node as a child. Although repeated visits to the same node do not increase the number of unique node evaluations required, there is redundancy involved as the results from searching the node further have already been taken into account. We therefore perform some bookkeeping at every visited node to prevent repeated visits.

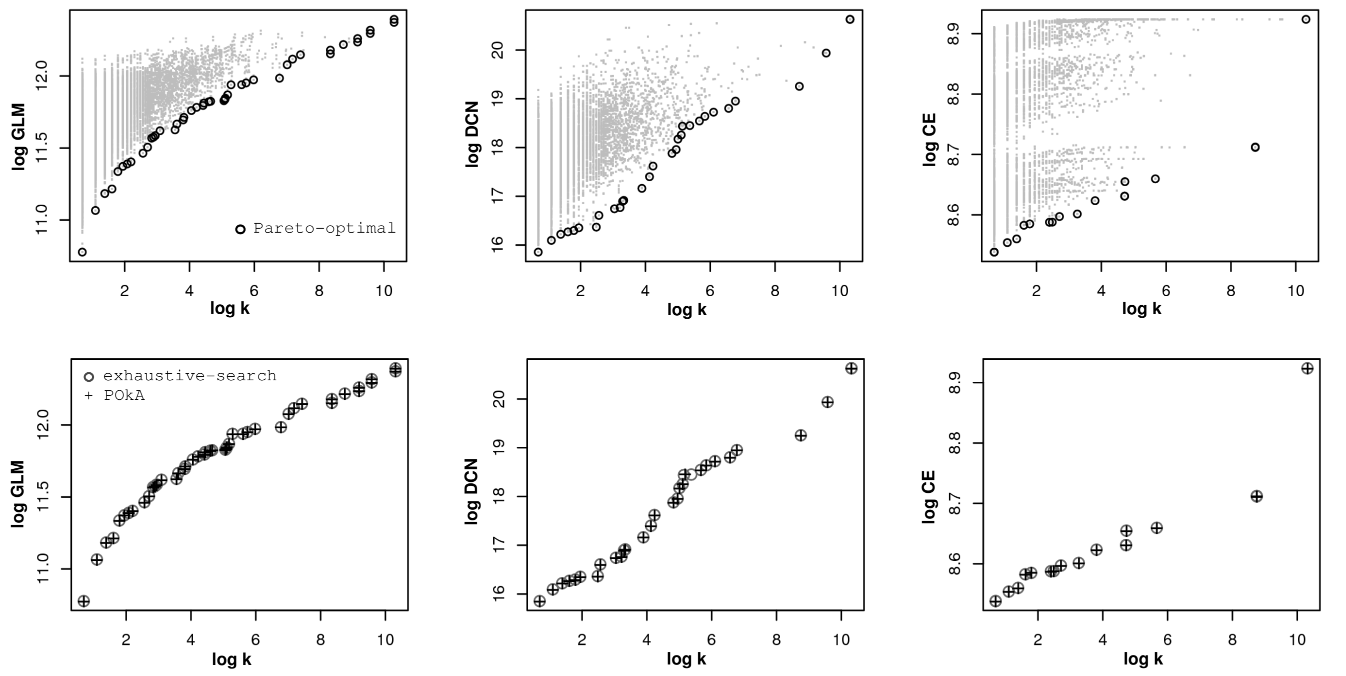

Figure 4: Top row shows the search space and true Pareto-optimal nodes while using different loss metrics. Bottom row shows the Pareto-optimal nodes found by exhaustive search and the nodes obtained by POkA.

The output from every node visited during a height search, i.e. the return values, is separately stored in a list *HBest*. Whenever a height search is to be initiated at a node, the list *HBest* is first checked to find if an entry corresponding to the node exists. If it does, then the node (and subsequent nodes) has already been searched and the stored values are returned. If not then the height search is done as usual. Similar to *HBest*, a list *DBest* is created for every node visited during a depth search. Depth search at a node is not performed if results for the node already exist in the list. Both lists are emptied before calling *DepthSearch* in Function 3. Functions 1 and 2 can be easily modified to maintain and use these lists.

6. PERFORMANCE ANALYSIS

We applied our methodology to the "adult.data" benchmark data set[1]. The data was extracted from a census bureau database and has been extensively used in studies related to k-anonymization. We prepared the data set as described in [1, 7]. All rows with missing values are removed to finally have a total of 30162 rows. The attributes used in this study along with their DGH lengths are listed in Table 2. Attributes with larger domains have been assigned a longer DGH. The DGHs used are not shown here due to space constraints. The total number of nodes in the lattice is 17920. The suppression limit η is set at 1% of the data set size, i.e. $\eta = 301$. Experiments are performed with three different loss metrics – namely general loss metric (GLM) [7], discernibility (DCN) [1] and classification error (CE) [7]. The attribute "Salary Class" is used as the class label while performing experiments with the CE metric. The lattice size in this case is 8960. Solutions reported by POkA are compared with those obtained by an exhaustive search of the entire DHL. Note that the number of nodes evaluated in the exhaustive search is equal to the size of the DHL, while that used by POkA is much less. Nonetheless, the exhaustive search provides us a definitive platform to judge the efficiency of POkA in finding true Pareto-optimal nodes. The depth d used in the experiment is set at $d_{avg} = \lceil \frac{\sum_i Ni}{s} \rceil = 3$, unless otherwise stated.

Attribute	Distinct values	DGH length
Age	74	6
Work Class	7	3
Education	16	3
Marital Status	7	3
Race	5	1
Gender	2	1
Native Country	41	4
Salary Class	2	1

Table 2: Attributes and DGH lengths used from the *adult census* data set.

6.1 Convergence

Pareto-optimal nodes identified by POkA for the three different loss metrics are shown in Fig. 4. The top row highlights the nature of the search space while using different loss metrics and the true Pareto-optimal nodes. All plots are in log scale. An interesting observation is that, for all three loss metrics, the search space is more dense towards lower values of k. This means as POkA proceeds towards finding the Pareto-optimal nodes in these regions, the number of nodes in the specialization graph of the base node decreases. Further, the Pareto-optimal nodes follow varied trends in the three metrics - concavity, convexity and disconnectedness.

[1]ftp://ftp.ics.uci.edu/pub/machine-learning-databases/adult/

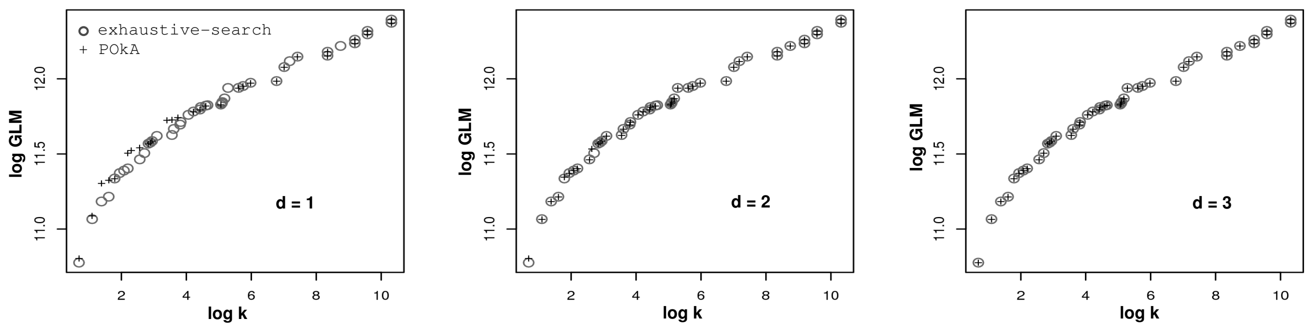

Figure 5: Comparison of Pareto-optimal nodes identified using depths of 1, 2 and 3 (=d_{avg}).

The bottom row in Fig. 4 compares the nodes identified by POkA with those obtained from an exhaustive search. POkA demonstrates noteworthy convergence to the true Pareto-optimal nodes across the different loss metrics. It manages to overcome the limitations that may be posed due to the arrangement of the Pareto-optimal nodes in the search space. While all solutions identified with GLM and CE are true Pareto-optimal nodes, one or two cases of sub-optimal or no identification is observed for DCN (notice the center of the plot). Nonetheless, identification of a sub-optimal node did not affect any subsequent searches. The requirement that the base node is a Pareto-optimal one is therefore not a strict one. POkA can very well be started from a sub-optimal node in the lattice and Pareto-optimal nodes with a k value lower than the starting node can still be discovered.

6.2 Impact of the depth parameter d

The depth d used in a depth search plays a crucial role in the identification of Pareto-optimal nodes. Table 3 shows the number of nodes evaluated in the lattice when using the GLM metric and varying depths. The maximum depth experimented with is 6, which is the equal to $\max(N_i)$, and ensures that nodes in $\mathcal{G}_{best}$ will be reached for any base node. However, using such a value results in the evaluation of more than 60% of the nodes. As discussed in Section 4.3, coverage of all nodes in $\mathcal{G}_{best}$ should not be required. The guiding principle derived is to search a depth of at least $d_{avg} = \lceil \frac{\sum_i Ni}{s} \rceil$ (which is equal to 3 with the DGH lengths used). Fig. 5 shows the Pareto-optimal nodes identified by using a depth of d_{avg} or less. All Pareto-optimal nodes have been identified by using a depth of d_{avg} (= 3). Using a depth of 2 resulted in some misidentification while a depth of 1 missed a number of the Pareto-optimal nodes. Using a depth higher than 3 did not prove to be of any advantage, less the number of nodes evaluated increased without necessity.

6.3 Node pruning efficiency

We found that the node pruning efficiency of POkA is much better in domain generalization lattices of bigger sizes. Bigger lattices may be formed either when the DGH lengths of the attributes considered are sufficiently long or when the number of attributes to anonymize is large. We experimented with the latter possibility and found that the number of nodes evaluated dropped exponentially with increasing lattice size. Fig. 6 shows the percentage of nodes evaluated when using $2, 3, \ldots, 8$ attributes for anonymization in the census data set. The depth to search is set to d_{avg} in each case. The higher number of evaluations for smaller lattices can be attributed to the fact that the observed high concentration of solutions in certain regions of the search space no longer holds. As nodes are spread out in the search space, potential number of Pareto-optimal nodes are also high, thereby resulting in the evaluation of a higher fraction of the nodes. On an average, node evaluations are observed to be around 20% across the three metrics when anonymizing for all attributes.

Depth d	Nodes evaluated	True optima
1	502 (2.8%)	22 (48.8%)
2	1945 (10.9%)	41 (91.1%)
3 (= d_{avg})	**4033 (22.5%)**	**45 (100%)**
4	6544 (36.5%)	45 (100%)
5	9205 (51.4%)	45 (100%)
6	11751 (65.6%)	45 (100%)

Table 3: Number of nodes evaluated when using different depth limits. Results are generated by using the GLM metric. The total number of nodes is 17920. **Number of true Pareto-optimal nodes is** 45.

6.4 Summary

To summarize the results, POkA can identify true Pareto-optimal nodes for a wide range of loss metrics that structure the search space in different ways. The experimental results corroborate the theoretical motivation behind using the average number of steps for full specialization of an attribute as the depth to search. The performance of POkA does not deteriorate even if certain nodes identified by it are not Pareto-optimal and used as base nodes. Finally, the number of nodes evaluated is a small percentage of the total number of nodes when the lattice is significantly bigger than the number of Pareto-optimal nodes it contain.

7. CONCLUSIONS

Privacy preserving data dissemination has to minimize the information loss in the anonymized data set while protecting the identity of underlying individuals to the maximum extent possible. In the context of k-anonymity, existing approaches address these aspects only partially by concentrating only on the issue of minimum information loss. Specifically, these approaches do not provide any information on the trade-off behavior between privacy and data utility.

In this paper, we proposed the POkA algorithm to find generalization schemes that are Pareto-optimal with respect to k-anonymity and a loss metric. By identifying Pareto-

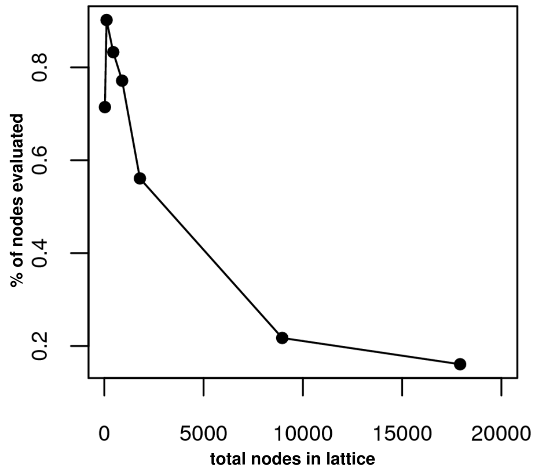

Figure 6: Percentage of nodes evaluated for varying lattice sizes. Results are generated by using the DCN metric. Varying lattice sizes are generated by considering varying number of attributes to anonymize.

optimal nodes in a domain hierarchy lattice we can guarantee that no other generalization can improve on the privacy aspect without deteriorating data utility. POkA uses a combination of depth first traversals of the lattice to efficiently find the Pareto-optimal nodes. Theoretical groundwork behind efficiently performing these traversals is presented. Results on a benchmark data set show that POkA has the potential to identify all Pareto-optimal nodes with a small percentage of node evaluations. They also demonstrate that the algorithm is applicable for a number of commonly used loss metrics.

Node evaluation can be further reduced if a better heuristic to stop the depth search can be found. An initial step in this direction is to investigate more stringent properties for the ground node. Another research direction is to extend the algorithm to other models of privacy. Pareto-optimality is used here as a two dimensional concept between privacy and data utility, while there exists privacy models that require the specification of more than a single parameter. Investigating Pareto-optimal anonymization with such models is a challenging area as well.

8. ACKNOWLEDGMENTS

This work was partially supported by the U.S. Air Force Office of Scientific Research under contract FA9550-07-1-0042. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Air Force or other federal government agencies.

9. REFERENCES

[1] Bayardo, R. J., and Agrawal, R. Data Privacy Through Optimal k-Anonymization. In *Proceedings of the 21st International Conference on Data Engineering* (2005), pp. 217–228.

[2] Dewri, R., Ray, I., Ray, I., and Whitley, D. On the Optimal Selection of k in the k-Anonymity Problem. In *Proceedings of the 24th International Conference on Data Engineering* (2008), pp. 1364–1366.

[3] Fung, B. C. M., Wang, K., and Yu, P. S. Top-Down Specialization for Information and Privacy Preservation. In *Proceedings of the 21st International Conference in Data Engineering* (2005), pp. 205–216.

[4] Golle, P. Revisiting the Uniqueness of Simple Demographics in the US Population. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society* (2006), pp. 77–80.

[5] Huang, Z., and Du, W. OptRR: Optimizing Randomized Response Schemes for Privacy-Preserving Data Mining. In *Proceedings of the 24th International Conference on Data Engineering* (2008), pp. 705–714.

[6] Hundepool, A., and Willenborg, L. Mu and Tau Argus: Software for Statistical Disclosure Control. In *Proceedings of the Third International Seminar on Statistical Confidentiality* (1996).

[7] Iyengar, V. S. Transforming Data to Satisfy Privacy Constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2002), pp. 279–288.

[8] LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. Incognito: Efficient Full-Domain k-Anonymity. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data* (2005), pp. 49–60.

[9] LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. Mondrian Multidimensional K-Anonymity. In *Proceedings of the 22nd International Conference in Data Engineering* (2006), p. 25.

[10] Loukides, G., and Shao, J. Capturing Data Usefulness and Privacy Protection in k-Anonymisation. In *Proceedings of the 2007 ACM Symposium on Applied Computing* (2007), pp. 370–374.

[11] Meyerson, A., and Williams, R. On the Complexity of Optimal k-Anonymity. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on the Principles of Database Systems* (2004), pp. 223–228.

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

[13] Samarati, P., and Sweeney, L. Generalizing Data to Provide Anonymity when Disclosing Information. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (1998), p. 188.

[14] Sweeney, L. Achieving k–Anonymity Privacy Protection Using Generalization and Suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10*, 5 (2002), 571–588.

[15] Sweeney, L. k–Anonymity: A Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10*, 5 (2002), 557–570.

[16] Wang, K., Yu, P., and Chakraborty, S. Bottom-Up Generalization: A Data Mining Solution to Privacy Protection. In *Proceedings of the 4th IEEE International Conference on Data Mining* (2004), pp. 249–256.