

Flexible Anonymization For Privacy Preserving Data Publishing: A Systematic Search Based Approach

Bijit Hore, Ravi Chandra Jammalamadaka, Sharad Mehrotra
{bhore, rjammala, sharad}@ics.uci.edu

Abstract

k -anonymity is a popular measure of privacy for data publishing: It measures the risk of identity-disclosure of individuals whose personal information are released in the form of published data for statistical analysis and data mining purposes (e.g. census data). Higher values of k denote higher level of privacy (smaller risk of disclosure). Existing techniques to achieve k -anonymity use a variety of “generalization” and “suppression” of cell values for multi-attribute data. At the same time, the released data needs to be as “information-rich” as possible to maximize its utility. Information loss becomes an even greater concern as more stringent privacy constraints are imposed [4]. The resulting optimization problems have proven to be computationally intensive for data sets with large attribute-domains. In this paper, we develop a systematic enumeration based branch-and-bound technique that explores a much richer space of solutions than any previous method in literature. We further enhance the basic algorithm to incorporate heuristics that potentially accelerate the search process significantly.

1 Introduction

The problem of anonymizing multi-attribute personal data (*microdata*) for public release, has generated a lot of interest in the research community over the recent years. Privacy concerns arise due to danger of disclosure of confidential information when individual-specific data sets are released to public. Since most data sets of interest contain one or more attributes which could be considered confidential by the owner, e.g., medical condition, financial information, ethnicity, level of education etc., if data sets are “insufficiently anonymized”, they may be linked with other available information, thereby disclosing identity of individuals and possibly their confidential information. A simple approach to preventing information leakage is to k -anonymize the released data set [7, 5]. A k -anonymized data set has the following properties: (i) All attributes that are explicitly identifying, example “Name”, “Address”, “Social security number” etc. are removed from the released set. (ii) A subset of the remaining attributes is determined (by an expert) to be the set of “Quasi Identifiers” (potentially

identifying set): these attribute values taken together might form an unique combination and therefore could be linked with external data to identify the record of an individual in the published data. (iii) It contains one or more *sensitive* attributes, the association of which with an individual could be considered confidential, e.g., type of disease, financial details like credit worthiness etc. Now, assume there are q attributes in the quasi identifier set, then the goal of k -anonymization is to “minimally” modify the released data set, such that for each record, the q -tuple of values of its quasi-identifier attribute set is indistinguishable from at least $k - 1$ other records in the released set. This is referred to as the anonymity set or equivalence class of the record.

The most commonly used technique for anonymization is that of *generalization*. Generalization refers to the action of replacing the original value by some more general value (e.g., replacing an exact numerical value by an interval). Occasionally some of the records might need to be suppressed (i.e., dropped from the published table). Generalizing a data set results in *information loss* which can be captured quantitatively by an associated *cost* metric for each of these operations. Typically, the cost associated with suppression is higher than generalization, since it leads to a greater loss of information. A number of cost metrics have been proposed in literature [9, 1, 2, 4] which capture various notions of information loss, ranging from application specific measures to very generic ones.

Generalization techniques in anonymization can be classified into two categories [3]:

Single dimensional techniques: Also referred to as *Attribute generalization* (AG) techniques, such transformations can be viewed as partitioning the multidimensional space of quasi-identifiers into a (possibly non-uniform) grid (see fig. 1(a) below) where each partition boundary cuts across the whole space. Generalization techniques are often mixed with *suppression* (i.e., records are completely dropped from the table) when good generalization is not possible. However, suppressions is not desirable due to the large information loss it results in.

Multidimensional partitioning techniques: These techniques lead to more flexible generalization

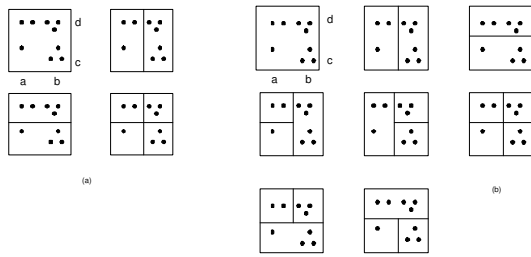


Figure 1: (a) 1-dimensional (b) Multidimensional schemes

schemes of the data as compared to those allowed by single dimensional approaches mentioned above. The rectangular partitions generated could belong to the family of *hierarchical* partitionings (also called *guillotine* partitionings) or *arbitrary rectangular partitionings*. In such schemes, a partition boundary need not cut across the entire space. Example of a hierarchical partitioning scheme is given in fig. 1(b). The class of arbitrary partitionings allows for any rectangular partitioning of the space and is therefore a superset of the class of single dimensional as well as hierarchical partitions.

A more detailed section on the related works can be found in the extended version of this paper [14].

Our Approach: In this paper we will investigate the class of hierarchical partitioning schemes. The key property of multidimensional approaches is that it allows one to change the nature of partitioning in one region of the data space (locally) without affecting the partitions in the other regions. The advantage is that it helps avoid suppression in many data distributions where single dimensional schemes would have enforced suppression. For instance, consider the generalizations of the 9 data points in figures 1 (a) and (b). If the required anonymization level was $k = 3$, the latter scheme allows a generalization where each class has size exactly 3 data points and there is no suppression at all (5th scheme) whereas the former is unable to do so. These extra degrees of freedom in multidimensional schemes substantially reduces the information loss in the final anonymized data sets.

Specifically, we develop an enumeration based branch-and-bound approach that enables one to explore the space of hierarchical partitionings of a multidimensional space. One of the key challenges is how to enumerate all distinct partitionings without duplication. We devise a novel algorithm to enumerate all hierarchical partitionings in the form of an enumeration tree. We are not aware of any prior work on systematic enumeration¹ for this class of partitionings. Further, based on our enumeration tree, we develop a fast lower-bound computation scheme that exploits monotonicity of the cost functions for lower bounding and pruning of unpromising solutions from the search space. We summa-

¹A scheme to enumerate each distinct partition exactly once.

rize the key contributions and advantages of our proposed solution methodology below:

Versatility of search: Our techniques can be used with a large class of information-loss metrics and privacy constraints like k-anonymity, *entropy l-diversity*, *(c,l)-recursive diversity* etc.

Progressive improvement in solution quality: At all times our algorithm maintains a lower bound to the true optimum (global minimum) cost and as a result one can stop the algorithm as soon as a solution with the required approximation bound is reached.

Incorporation of search heuristics: We enhance the basic enumeration-based search algorithm to allow a variety of application-driven heuristics to be incorporated in order to accelerate the search. We introduce a few of these heuristics in section 4.

Next, we describe our enumeration algorithm.

2 Enumerating Partitions

In this section we describe the enumeration algorithm for hierarchical partitions of the space. From here on we will use the term “partition” and “partitioning” interchangeably to refer to a complete partitioning scheme of the space.

Let us consider a d -dimensional space where each dimension corresponds to one attribute of the data set. Fig. 2 shows a 2-dimensional space where one dimension is *Age* and the other *Salary* with domains [30yrs, 60yrs] and [50K, 200K] respectively. Let the split set comprise of the four splits: (1):Age=40, (2):Age = 50, (3):Salary = 100K and (4):Salary = 150K. The figure shows the finest level of partition (i.e., where all splits are used and all of them cut across the whole space). A hierarchical partition of a d -dimensional space consists of a set of disjoint d -dimensional hyper-rectangles that cover the entire space and can be generated by recursively splitting the space. Such partitionings can be represented by binary trees, see fig. 3 for example.

The objective of our enumeration algorithm is to generate the set of all possible hierarchical partitions of a rectangular space using a given set of *splits*. The difficulty arises in trying to enumerate them in a duplicate-free manner. To see the challenge of duplicates, consider the partitioning scheme and its binary tree representation in fig. 3: The tree could have been created by the choosing the following sequence of splits: (3), (1), (4), (2) and (2), where the integer within ‘(’ and ‘)’ is the split-id and the integer within ‘[’ and ‘]’ is the timestamp at which the node was split. We follow the rule that nodes that occur earlier in DFS order in the binary tree are split before those that appear later (if they are split at all that is). Now, notice that this is not the only way this partition could have been generated, it could have also been generated by the following sequence of splits:

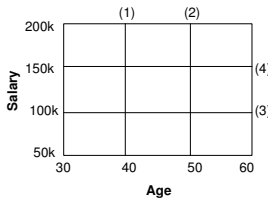


Figure 2: Finest partition of the space

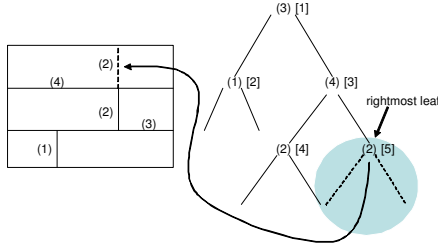


Figure 3: A sequence of splits generating a partition

(4), (3), (1), (2) and (2). Though the final partitions are identical, the binary trees resulting from these two sequences are structurally very different.

The enumeration algorithm that generates the *partition enumeration tree (PET)* is described in the extended version [14]. Fig. 4 shows a partial PET of all possible hierarchical partitioning schemes of a two dimensional space using 3 splits (2 vertical and 1 horizontal).

Next, we describe the setup of the optimization problem: objective function, constraints and pruning techniques employed to find an optimal partitioning scheme.

3 Search for Optimal Partitioning

Our optimization problem is a constrained minimization problem where the *Cost* denotes the information-loss due to the generalized representation of the data set. It is denoted by a non-negative real-valued function $Cost : N \rightarrow \mathbb{R}$, where N is the set of all nodes in PET (ie., set of all partitioning schemes). *Constraint* denotes the privacy criteria (e.g., k -anonymity, l -diversity etc.) which determines the set of all admissible partitioning schemes. Different combinations of these two parameters define different classes of optimization problems. In this paper we consider one cost function and 3 different privacy-constraints.

Cost function: Discernibility metric (DM): Proposed in [2], it assigns a penalty to each tuple based on how many tuples in the transformed data set are indistinguishable from it. If a tuple belongs to an anonymity group S_i of size n (i.e. has $n - 1$ other tuples present in it), then that tuple is assigned a penalty of n . If a tuple is suppressed a penalty equal to the size of the data set $|T|$ is assigned.

$$DM = \sum_{\forall i, |S_i| \geq k} |S_i|^2 + \sum_{\forall \text{ Suppressed Tuples}} |T|$$

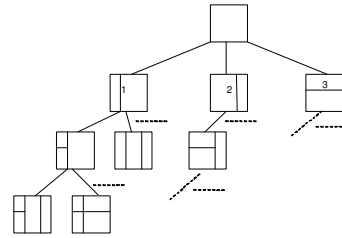


Figure 4: Partition enumeration tree

Constraints: The 3 classes of constraints we used in our experiments are the following:

(1) k -Anonymity: For every tuple t' in the released table, there should at least be l distinct indices (including its own) (i_1, i_2, \dots, i_l) , where $l \geq k$, such that $t_{i_1}(Q) = t_{i_2}(Q) = \dots = t_{i_l}(Q) = t'$, where $t(Q)$ denotes the projection of t on the quasi-identifier attribute set Q .

(2) Entropy l -diversity: Using the definition from [4], a table is entropy l -diverse if for every equivalence class (partition block) b , the following holds.

$$-\sum_{r \in R} p_{(b,r)} \log(p_{(b,r)}) \geq \log(l)$$

where $p_{(b,r)}$ denotes the fraction of tuples in the block b with sensitive attribute value equal to r , and R denotes the set of all values (categorical) for the sensitive attribute. This constraint ensures that each equivalence class has at least l “well represented” values of the sensitive class.

(3) k -Anonymity with length restrictions: Here in addition to the k -anonymity constraint we impose a minimum length constraint on the edge length(s) of a partition block in the anonymized data set, e.g., “In the released data set, no individual’s salary should be specified to an interval of length smaller than 10K” and/or “Age of any individual should not be specified to an interval less than 5 years” etc.

Now the optimization problem can be formally defined as follows.

DEFINITION 3.1. (Optimal Anonymization Problem): Given a cost function $Cost$ and a constraint, find a hierarchical partitioning of the multidimensional data set such that the constraint is not violated and the cost of the solution is minimized.

3.1 Searching for the Optimum Solution The efficiency of the search algorithm critically depends on how much of the search space can be pruned away during enumeration. The following condition must hold for pruning a node n (at its generation time) in our PET:

CONDITION 3.1. (Condition for pruning) The subtree rooted at n in the PET can be pruned if at least

one of the following two conditions hold at each node in the subtree: (1) The partition corresponding to the node is not an admissible solution (violates the privacy constraint) (2) All partitions in the subtree rooted at n have a cost larger than the current_minimum_cost in the algorithm.

If a node n (and the subtree rooted at n) in the PET cannot be pruned for the first condition, then we check for the second condition. To carry out this check efficiently (i.e., one that does not traverse the subtree rooted at n), one needs to define a function LB that estimates a lower bound to the minimum cost achievable in the subtree. We discuss properties of our enumeration tree and provide examples in the extended paper.

Next, we describe a priority-queue based optimization for speeding up the basic search algorithm.

4 Accelerating Search using Priorities

As it turns out, more often than not in spite of pruning, the search space continues to remain extremely large. As a result, the depth-first traversal order of nodes (as in the basic recursive algorithm presented in [14]) turns out to be an inefficient way to explore the solution space for most objective functions of interest. Here, by efficiency we mean “how soon does the algorithm finds a solution close to optimal”. To do away with these limitations of the recursive algorithm, we propose a new solution-space exploration scheme that is both theoretically sound (i.e. guarantees completeness of search) and at the same time, allows one to incorporate heuristics to accelerate convergence to the optimum solution. The side-effect being: good approximations to the optimal are generated much earlier in the sequence. Now, instead of traversing the PET in a fixed order which is agnostic of the objective function, our new algorithm uses a *priority* function to direct its search at each step.

The non-negative real-valued function **Priority** : $\mathbf{N} \rightarrow \mathbb{R}$, where \mathbf{N} is the set of nodes in PET, assigns a numeric value that is used as the key to insert n into the priority queue. Our algorithm admits the usage of arbitrary priority generating functions. We experimented with the following functions to generate node priorities.

- 1) LB (lower bound function): The intuition being, the node of PET that has lowest lower bound is the most promising one to explore (i.e., expand further below).
- 2) $Cost$ (cost function): Cost of the solution at a node n was used to prioritize the search.
- 3) $LB/Cost$ (ratio of the lower bound and the cost): Similarly, the rationale behind $LB/Cost$ was to go down branches which had the highest potential of cost improvement.

There are two modes in which the priority-based algorithm can be executed: (A) Geared towards finding

an optimum solution, where the lower bound $LB(n)$ is used to decide whether to prune node n or not. (B) Geared towards finding a c -**approximate solution** to the optimum, where the ratio $\alpha(n) = LB(n)/Cost(n)$ is used to prune node n from the search space. For instance, if the user wants to find a 3-approximate solution, a node with α value greater than $1/3$ can be safely dropped from the priority queue.

More details of the priority-queue based algorithm is given in [14]. Next, we present some of our experimental results.

5 Experiments

Experimental setup: All the experiments were run on a Pentium 4, 3.00 GHZ processor machine with 4 GB RAM. The machine was running windows XP operating system and our enumeration algorithm was implemented using the VC++ development environment.

Data sets: There were three different datasets that were used in the experiments. The first real data set used in the experiments is the *Adult* data set from the Irvine machine learning repository [8], which has in some ways become the benchmark of measuring the k-anonymization algorithms. This data set has 9 attributes (Age, Geography, Gender, Race, Working Class, Occupation, Education, Class, Marriage) and reports the actual census data. The data set has 30,162 tuples.

The second real data set is the *Coil 2000* data set, also from the same repository. The data contains information on customers of an insurance company. We have taken the first five dimensions of the Coil data set, namely, customer subtype, number of houses, average size of a household, average age and customer main type. This data set has 9882 tuples.

The third dataset used in our experiments was synthetically generated, which we will refer to as *SimpleNormal*. This data set has two integer dimensions with values ranging from 0 to 100. The values follow a two dimensional normal distribution.

Purpose of the experiments: The purpose of our experiments was to measure: a) The scalability of our enumeration algorithm and b) To compare the solutions of the enumeration algorithm with those of the greedy algorithm proposed in [3]. Since the authors in [3] show that their greedy approach outperforms other previously suggested techniques, as comparison with their results illustrates the advantages of our approach over all others till date.

Scalability of our approach: We note the time the algorithms takes to find the optimal solution (in most cases we report how close it approximately gets to the theoretical best). Here, we will only give a summary of our findings due to space restrictions and we direct the interested reader to [14] for a more comprehensive

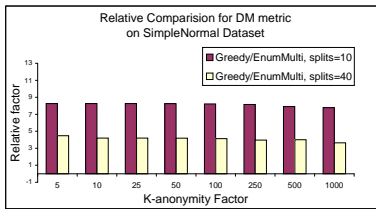


Figure 5: DM, SimpleNormal

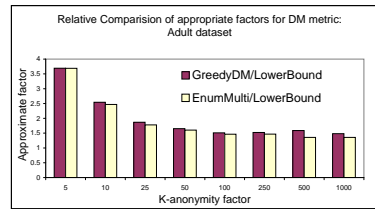


Figure 6: DM, Adult

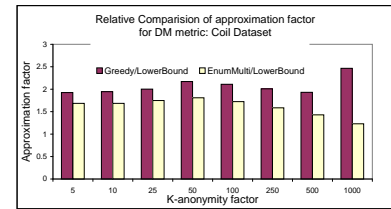


Figure 7: DM, Coil

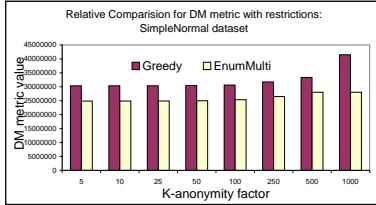


Figure 8: DM, SimpleNormal, Length

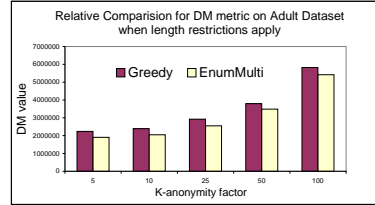


Figure 9: DM, Adult, Length

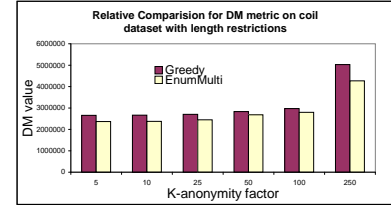


Figure 10: DM, Coil, Length

experimental analysis. The most critical parameter for the enumeration algorithm is the number of splits as it dictates the size of the solution space. The algorithm finds optimal solution when the number of distinct splits used are less than equal to 10, in under 30 minutes. When the number of splits were increased, we needed to drop some solutions as they cannot fit in the memory and thereby lost the guarantee of optimality of the final solution. When the splits were more than 10, our algorithm found solutions within a factor of 3.5 of the theoretical lower bound of the optimal solutions. Whereas our enumeration algorithm can determine the optimal solution or those within a specified factor of the lower bound (to the optimal) given sufficient space and time, the greedy algorithms proposed thus far typically return a locally optimal solution.

Comparison to the greedy algorithms: Here we present results for simply one metric (DM) and three different constraints (k-anonymity, l-diversity and minimum length restrictions). Results on other metrics can be found in [14]. Each of these constraints have a parameter that is varied, k for anonymity, l for diversity and $length$ for the minimum allowed size of the range of attribute-value in anonymity classes. We applied a total of 4 different sets of constraints to each combination of the dataset and cost metric. We use the following notation “(D, M, S)” to denote an experiment on a dataset D with metric M with additional constraint S (besides the normal k-anonymity constraint). For all the following experiments the enumeration algorithm was run for maximum of 5 hours.

Experiment 1:- (SimpleNormal, DM, <K-anonymity>): We used two sets of splits (5×5 and 20×20) and ran both our enumeration algorithm and the greedy algorithm. Fig. 5 plots the ratio of greedy value found to value of the solution found by our enumeration algorithm (represented as “EnumMulti” in the graphs). For a total of 10 splits (5×5), EnumMulti picked optimal solutions which

were on an average 8.2 times better than the greedy solutions. All the optimal solutions were found under 30 mins. For the 40 splits case (i.e. 20×20), EnumMulti was only 4 times better than the greedy. As the number of splits increase, the quality of results generated by the enumeration algorithms and the greedy algorithms converge.

Experiment 2:- (Adult, DM, <K-anonymity>): We used a total of 101 splits to partition the Adult dataset. EnumMulti outperforms the greedy approach by a significant margin at higher values of k (greater than 100). Fig. 6 shows the ratio of the best solution found so far to the lowest lower bound (of the cost) of any partition tree currently in the priority queue. We achieved an approximation ratio of 1.35 times the lower bound for $k = 1000$.

Experiment 3:- (Coil, DM, <K-anonymity>): Similar comparisons were carried out using the Coil dataset. Here the advantage of EnumMulti over greedy is significantly more than it was for the Adult dataset as can be seen in fig. 7. But, the similar trend is seen, wherein the greedy solution approaches the EnumMulti’s solution at smaller values for k .

Experiment 4:- (SimpleNormal, DM, K-anonymity, Length): Here we used 50×50 (a total of 100) splits. We also imposed a minimum-length restriction of 10 units along each dimension (i.e., “no partition-block should have an edge-length of less than 10 units”). Fig. 8 shows the result. The enumeration algorithm outperforms the greedy algorithm for all values of k by a significant margin.

When constraints apply, the greedy algorithm does not have many splits at its disposal at each stage and it gets caught in some local minima with a high probability. In comparison, EnumMulti systematically enumerates this reduced solution space, and therefore more likely to do much better than greedy when constraints apply.

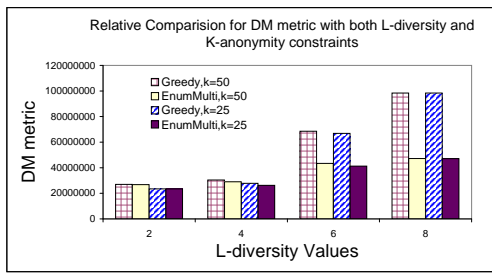


Figure 11: DM, Adult, L-diversity

Experiment 5:- (Adult, DM, $\langle K$ -anonymity, Length): This run was same as the previous one, but using the Adult data set. In seven out of the nine of the dimensions (Age, Geography, Race, Working_class, Occupation, Education, Marriage) we imposed a minimum length requirement equal to twice the inter-split distance. Fig. 9 shows the results. Here as well the enumeration algorithm does well for all values of k as compared to the greedy approach, therefore confirming our intuition from the previous experiment.

Experiment 6:- (Coil, DM, $\langle K$ -anonymity, Length): In this experiment a minimum length equal to twice the inter-split distance (minimum possible) was imposed on all the five dimensions of the Coil dataset. Fig. 10 shows the result.

Experiment 7:- (Adult, DM, $\langle K$ -anonymity, L-diversity): We compared both the approaches when l -diversity restriction together with the k -anonymity is applied on the Adult dataset. We have taken *occupation* dimension as the sensitive attribute which has 14 different values, similar to [4]. For this experiment we have kept the anonymity constraint k constant and varied the l -diversity. This was done for two values of k (25 and 50). Fig. 11 shows the result. EnumMulti outperforms the greedy approach when l -diversity desired is more than 4 for both values of k .

In summary, the following are the significant results of our experiments: 1) The enumeration algorithm performs better than greedy in all experiments and performs significantly better than greedy for higher values of k (usually more than 100), when constraints are not applied. 2) For lower values of k (less than 100) and when constraints apply, greedy comes close to solutions found by EnumMulti. 3) When constraints are applied, EnumMulti significantly outperforms the greedy algorithm for all values of k . 4) EnumMulti picks close to optimal solutions under an hour for most datasets and constraints.

More experimental results characterizing the performance of priority functions and running times of our algorithm can be found in the extended version of our paper [14].

6 Conclusion

In this paper, we explored the problem of computing optimal generalizations of multidimensional data under a variety of privacy constraints. We developed an enumeration based search algorithm for the class of hierarchical rectangular partitioning schemes to that can be employed to generalize the data. In order to make the search for the optimal partitioning feasible we deduced pruning criteria to reduce the search space to a manageable size. Subsequently, we proposed a flexible priority queue based algorithm to incorporate a variety of search heuristics. We did extensive experimentation using a variety of combinations of the cost metrics and constraints motivated by some popular measures of privacy and information-loss from data mining literature.

Finally, we note that while this paper is centrally motivated by privacy-preservation for data publishing, the problem is in many ways similar to that of optimal histogram construction and several other data partitioning problems occurring in query optimization, image compression, parallel computing etc, where similar partitioning schemes have been considered [10, 11, 12, 13]. All these applications could also benefit from the current work.

References

- [1] Vijay S. Iyengar. "Transforming Data to Satisfy Privacy Constraints", SIGKDD'02 Edmonton, Alberta, Canada.
- [2] Roberto J. Bayardo, Rakesh Agrawal. "Data Privacy Through Optimal K-anonymization", ICDE 2005.
- [3] Kristen LeFevre, David DeWitt, Raghu Ramakrishnan. "Mon-drian Multidimensional K-Anonymity", ICDE 2006
- [4] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, Muthuramakrishnan Venkitasubramaniam. "L-Diversity: Privacy Beyond K-Anonymity", ICDE 2006
- [5] L. Sweeney. "Achieving K-anonymity privacy protecting using generalization and Protection", International Journal on Uncertainty, Fuzziness and Knowledge-Base Systems, 2002.
- [6] P. Samarati. "Protecting respondents identities in microdata release", IEEE Transactions on Knowledge and Data Engineering, 2001.
- [7] Pierangela Samarati, Latanya Sweeney. "Protecting Privacy When Disclosing Information: K-anonymity and its Enforcement through Generalization and Suppression", International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol. 10, No. 5 (2002) 571-588
- [8] UCI Machine Learning Repository, <http://kdd.ics.uci.edu>
- [9] Willenborg, L., Waal, T., D. "Elements of Statistical Disclosure Control", Lecture Notes in Statistics, 155, Springer-Verlag, New York.
- [10] Anily, S., Federgruen, A. "Structured Partitioning Problems", Operations Research Vol. 39, Issue 1, 1991, pp. 130 - 149.
- [11] Muthakrishnan, S., Poosala, V., Suel, T. "On Rectangular Partitionings in Two Dimensions: Algorithms, Complexity and Applications", International Conference on Database Theory, 1997.
- [12] Berman, P., Dasgupta, B., Muthukrishnan, S. "Exact Size of Binary Space Partitionings and Improved Rectangle Tiling Algorithms", SIAM J. Discrete Math, Vol. 15., No. 2, pp. 252-267.
- [13] Muthukrishnan, S., Suel, T. "Approximation Algorithms for Array Partitioning Problems", Journal of Algorithms 54, 2005, pp. 85-104.
- [14] Hore., B., Jammalamadaka, R., C., Mehrotra, S. "Systematic Search for Optimal k -Anonymization", UCI-ICS 2006, tech-report. <http://www.ics.uci.edu/~bhore/publication.html>