Approximate Algorithms for k-Anonymity *

Hyoungmin Park Kyuseok Shim
School Of Electrical Engineering and Computer Science
Seoul National University
Seoul, Korea
{hmpark@kdd,shim@ee}.snu.ac.kr

ABSTRACT

When a table containing individual data is published, disclosure of sensitive information should be prohibitive. A naive approach for the problem is to remove identifiers such as name and social security number. However, linking attacks which joins the published table with other tables on some attributes, called quasi-identifier, may reveal the sensitive information. To protect privacy against linking attack, the notion of k-anonymity which makes each record in the table be indistinguishable with k-1 other records has been proposed previously. It is shown to be NP-Hard to k-anonymize a table minimizing the number of suppressed cells. To alleviate this, $O(k \log k)$-approximation and $O(k)$-approximation algorithms were proposed in previous works.

In this paper, we propose several approximation algorithms that guarantee $O(\log k)$-approximation ratio and perform significantly better than the traditional algorithms. We also provide $O(\beta \log k)$-approximate algorithms which gracefully adjust their running time according to the tolerance $\beta(\geq 1)$ of the approximation ratios. Experimental results confirm that our approximation algorithms perform significantly better than traditional approximation algorithms.

Categories and Subject Descriptors

H.2.8 [**Database Application**]: Data Mining; K.4.1 [**Public Policy Issues**]: Privacy

General Terms

Algorithms, Security, Performance, Theory

Keywords

Privacy preservation, anonymity, data publishing, data mining, local recoding

*This research is supported by the Ministry of Information and Communication, Korea, under the College Information Technology Research Center Support Program, grant number IITA-2006-C1090-0603-0031.

SIGMOD'07, June 12–14, 2007, Beijing, China.

1. INTRODUCTION

Data privacy and data utility are quite naturally opposite to each other. If we perturb, generalize or prevent some data to protect privacy, the amount of information which we can use will be reduced. But, in typical data mining application such as tracking epidemics and product marketing we are not concerned about individual information. Interesting trends or correlations are the object of data miners. On the other hand, with respect to privacy, individual information must be protected but the trends may be exposed.

Ideally, we want to remove as little information as possible, but want to allow inferring overall trends in data without releasing much information about individuals. A naive approach is to remove the identifiers such as name and social security number of each individual in the data. However, although the trends in the data are preserved, this sanitization still does not ensure yet the privacy of individuals.

An adversary can link individuals with other data using certain attributes and identify the individuals in the data. For example, consider the table in Figure 1-(a). Assume that we know Bob is a 24 year old single male from USA and his record definitely exists in the table. Then we can easily conclude that the first record must be Bob's. In this attack of the adversary, we call the set of attributes such as age, marital status, home country and gender used in linking a *quasi identifier* since the adversary identifies the individuals using those attributes. To prevent this attack, Samarati and Sweeny proposed in [15, 16] the notion of *k-anonymity* on quasi identifier in which we alter some values for quasi-identifier attribute set so that k individuals are not distinguishable and an adversary can not identify individuals exactly.

Suppose we want to release a table of private data to the public, and we are allowed to suppress various entries in the table. If this suppression is done in such a way that every record becomes textually indistinguishable (entry for entry) from k-1 other records in the table, we say that the new modified table is *k-anonymized*. In this paper, we will only consider the special case of suppressions where each entry of every attribute is either included in the output, or replaced with '*' character.

Consider the relation in Figure 1-(a). Suppose our task is to 2-anonymize the relation before its release. If the data has been permitted to use the value '*' for generalization of attributes, one possible 2-anonymization of the table would be the one in Figure 1-(b). With the anonymization, the first and third records become the same while the other two records are identical. The replacement with '*' for the value

Age	Marital status	Home country	Gender
20∼29	Single	USA	Male
30∼39	Divorce	China	Female
20∼29	Single	USA	Female
30∼39	Separation	Korea	Female

(a) An employee table

Age	Marital status	Home country	Gender
20∼29	Single	USA	*
30∼39	*	*	Female
20∼29	Single	USA	*
30∼39	*	*	Female

(b) A 2-anonymized table

Figure 1: An example of 2-anonymization

of an attribute is called suppression. Note that we performed 6 suppressions to achieve 2-anonymity for the table.

Many researchers have studied *k-anonymity*. (See [1, 3, 10, 12, 15, 16].) Their approaches are generally classified into two categories, *global recoding* and *local recoding* [17]. In order to anonymize a database, global recoding models map the values in the domains of the quasi-identifier attributes to generalized or altered values. However, local recoding models map individual instances of data items to generalized or suppressed values. Intuitively, local recoding amends less data than global recoding. Therefore, information loss in local recoding is generally smaller than global recoding. For that reason, we will focus on local recoding in the paper.

Note that k-anonymity prevents privacy breach but we do not know the quality of the k-anonymized data quantitatively. To evaluate how useful k-anonymized data is, Meyerson proposed a measure in [12] which counts the number of suppressed cells to achieve k-anonymity. We will adopt this measure for evaluating data utility in this paper. We want to simultaneously ensure the anonymity of individuals to a group of size k but want to withhold a minimum amount of information to achieve this privacy level. This optimization problem is shown to be generally NP-hard in [12]. For this problem, as far as we know, there exist two approximations that are $O(k \log k)$-approximation and $O(k)$-approximation presented in [12] and [3] respectively.

In this paper, we will present $O(\log k)$-approximation algorithms for k-anonymity. The contributions of this paper are as follows:

- We present $O(\log k)$-approximate algorithms which have significantly better approximation ratio than the best known approximation ratio of $O(k)$ for the same problem. We also provide a heuristic algorithm that performs well even though we cannot show the approximation bound.
- We provide $O(\beta \log k)$-approximate algorithms with $\beta \geq 1$ which gracefully adjust their running times according to the tolerance β of the approximation ratio.
- We present the result of detailed experimental study for these algorithms on real-life data sets. Experiments confirm that that our algorithms outperform the existing algorithms in both approximation ratios and execution times.

2. RELATED WORK

We will first study the global recoding methods[5, 9] and next discuss the local recoding approaches[3, 10, 12].

In [5], to k-anonymize a table, we partition the domain of each quasi-identifier attribute into intervals and replace the values in the attribute with the intervals to which the values belong. Then the records in the table are naturally grouped by the same intervals of the quasi-identifiers. If the sizes of all groups are at least k, then the modified table satisfies k-anonymity. Two goodness measures for k-anonymity such as the discernibility measure and the classification measure were proposed in [5]. To find a partition which satisfies k-anonymity and minimizes the given measure, all possible partitions are explored with possible prunings.

The work in [9] uses a hierarchy of values of each domain such as a taxonomy tree and performs an apriori style algorithm such as the one in [4]. It finds every k-anonymization for all possible subsets of quasi-identifier attributes.

It is shown in [12] that producing an optimal k-anonymization of a table for $k \geq 3$ in a way that minimizes the number of entries suppressed is NP-Hard. In [3], the k-anonymization problem is shown to be NP-Hard even when the attribute values are ternary. An $O(k \log k)$-approximate algorithm for that problem is provided in [12]. Moreover, the best known approximate algorithm with the ratio $O(k)$ for the number of suppressed cells is presented in [3]. However, due to its loose approximate ratio, the approximate algorithm shows poor performance for large k.

The work in [10] provides a 2-approximate algorithms for other goodness measures such as the discernibility measure and normalized average equivalence class size measure. The other work in [2] proposes 4- and 80-approximate algorithms using clustering respectively, but they minimize the maximum radius of clusters' radii or the sum of products of the radius and size of each cluster.

Recently, ℓ-diversity[11] which is a more strong privacy protection model is proposed. They point out that for records with sensitive values which are the values people do not want to disclose to public, although the records are indistinguishable for quasi-identifier, if they have the same sensitive values, their sensitive values are disclosed exactly. To prevent such disclosure, they insist that indistinguishable records for quasi-identifier have at least ℓ different sensitive values.

3. PRELIMINARIES

3.1 Definitions

Consider a relational table T with m attributes in quasi-identifier. A tuple $t_i \in T$ is drawn from $\sum^m$ where $\sum$ is a finite set of possible values for all attributes of quasi-identifier. $t_i[j]$ is the value of the j-th attribute in t_i. Any instance of the table can be formally represented as a subset $T \subseteq \sum^m$. Let the symbol $*$ be a symbol not in $\sum$.

DEFINITION 3.1: *Let us assume that f is a mapping function from T to $(\sum \cup \{*\})^m$. The mapping f is defined as a suppressor on T if for all $t \in T$ and $i = 1, 2, \ldots, m$, $f(t)[i]$ is either $t[i]$ or $*$.*

EXAMPLE 3.2: *Consider a table* $T = \{1010, 1110, 0110\}$ *and a mapping function* f *such that* $f(b_1b_2b_3b_4) = **b_3b_4$. *Then, the mapping* f *results in* $T' = \{**10, **10, **10\}$. ∎

A suppressor is one of anonymizations in which every tuple $t \in T$ has a corresponding anonymized tuple t' $(= f(t))$ in an anonymized table T' of T where the attribute values of t' are identical to the attribute values of t, except the suppressed attribute values that are represented by the new symbol $*$. A suppressor can be similarly generalized to work on a relation T. Thus, $f(T)$ is considered as a multiset when two or more tuples in T map to the same suppressed tuple, *i.e.* for $t, t' \in T$ with $t \neq t'$, we have $f(t) = f(t')$. The following is the definition of k-anonymization.

DEFINITION 3.3: *Let* f *be a suppressor on a table* $T = \{t_1, ..., t_n\}$. *Then,* $f(T)$ *is* k*-anonymous if and only if for every* $t_i \in T$, *there exist* $k-1$ *indices* $i_1, i_2, ..., i_{k-1} \in \{1, 2, ..., n\}$ *such that* $i, i_1, i_2, ..., i_{k-1}$ *are all distinct and* $f(t_i) = f(t_{i_1}) = f(t_{i_2}) = \cdots = f(t_{i_{k-1}})$. *We call* f *a* k*-anonymizer on* T.

In other words, when a suppressor on a table makes the table k-anonymous, every anonymized tuple is a member of a multiset of (at least) k identical anonymized tuples.

3.2 $O(k \log k)$-Approximation

We now present the basic idea of the $O(k \log k)$-approximate algorithm proposed for k-anonymity in [12]. We will restate the definitions, lemma and its corollary in [12]. Let us define the diameter of a given table S as follows:

DEFINITION 3.4: *For* $u, v \in S$, *let* $d(u, v)$ *be the distance between* u *and* v, *and define* $d(u, v) := |\{j : u[j] \neq v[j]\}|$. *The diameter of* S *is*

$$d(S) := \max_{u,v \in S} d(u, v)$$

Intuitively, the diameter of S is the maximum number of attributes in which two records of S differ. It is useful to note that this function is a metric.

DEFINITION 3.5: *For positive two integers* k_1, k_2 *s.t.* $k_1 \leq k_2$, *a* (k_1, k_2)*-cover of a relation* T *is defined as a collection of tables* $\{S_1, ..., S_\ell\}$ *in which* $S_i \subseteq T$, $k_1 \leq |S_i| \leq k_2$ *for* $i = 1, ..., \ell$, *and for every* $t \in T$, *there is an* S_i *such that* $t \in S_i$. *Similarly, a* (k_1, k_2)*-partition of* T *is defined as a* (k_1, k_2)*-cover where all* S_i*s are disjoint.*

It is shown in [12] that any $(k, |T| - k)$-partition can be transformed into a $(k, 2k-1)$-partition without increasing the number of $*$'s. Thus we can compute a $(k, 2k-1)$-partition for k-anonymity.

DEFINITION 3.6: *For a table* T, $OPT(T)$ *is defined as the minimum number of* $*$*'s to be inserted in the tuples of* T *for an optimal solution of* k*-anonymization and for* $S \subseteq T$, $ANON(S)$ *is defined as the total number of entries (i.e. cells) of the tuples in* S *that must be replaced with the symbol* $*$ *in order for all tuples in* S *to be identical.*

For a table T, $OPT(T) = \min_\Pi(\sum_{S \in \Pi} ANON(S))$ holds, where the minimum is taken over all possible partitions Π of T in which every set has at least k tuples. Let Π^* be an optimal partition of T for k-anonymity, *i.e.* $\sum_{S \in \Pi^*} ANON(S) = OPT(T)$. For a partition Π of a table T, let us define $d(\Pi) = \sum_{S \in \Pi} d(S)$. Intuitively, $d(\Pi)$ represents the sum of the diameters of all subsets in Π. Then, we have the following lemma:

LEMMA 3.7: *For a relation* T, *we have*

$$k \cdot \min_\Pi d(\Pi) \leq k \cdot d(\Pi^*) \leq OPT(T) \leq 3k^2 \cdot d(\Pi^*)$$

where the minimum is taken over every possible partition Π *of* T, *in which the cardinality of every set is in the range of* $[k, 2k-1]$.

We are now interested in the optimization problem of finding a $(k, 2k-1)$-partition Π of T such that $d(\Pi)$ is minimized. Let us call this the *k-minimum diameter sum* problem. The interesting relationship between k-anonymity and this problem can be illustrated with the following corollary.

COROLLARY 3.8: *Let* $\alpha \geq 1$, *and* Π *be a* $(k, 2k-1)$*-partition with diameter sum at most* α *times that of an optimal solution of* k*-minimum diameter sum. Then the algorithm that anonymizes each* $S \in \Pi$ *by setting* $t[i] := *$ *to each* $t \in S$ *for every suppressing attribute* i *with which there exit a pair of* $u, v \in S$ *with* $u[i] \neq v[i]$ *is a* $3\alpha k$*-approximation algorithm to optimal* k*-anonymity.*

Therefore, a $3k(1 + \ln 2k)$-approximation to optimal k-anonymity can be obtained if there is a $(1+\ln 2k)$-approximation to the k-minimum diameter sum problem.

EXAMPLE 3.9: *Suppose that we want to 4-anonymize the table in Figure 2-(a). Since any* $(k, |T| - k)$*-partition can be transformed into a* $(k, 2k-1)$*-partition without violating* k*-anonymity, we want an optimal (4,7)-partition for 4-anonymity. The anonymized table of 8 records should always be split into 2 subsets each of which has exactly 4 tuples to guarantee a (4,7)-partition. Among possible (4,7)-partitions, we can easily see that the partition in Figure 2-(b) is an optimal partition for 4-anonymity. The number of* $*$*'s to achieve 4-anonymity with the partition is 24.*

*On the other hand, the partition shown in Figure 3 is an optimal (4,7)-partition for 4-minimum diameter sum. Since we should have 2 subsets each of which has 4 records and there does not exist any subset of 4 records with the diameter of 0 or 1, the partition shown in Figure 3 is an optimal solution for 4-minimum diameter sum. The diameter of every subset is 2 and thus the diameter sum of this partitioning is 4. The number of *'s for this partitioning is 32 for 4-anonymity which is 33% more than that of the optimal 4-anonymity partitioning.* ∎

4. APPROXIMATION ALGORITHMS FOR K-ANONYMITY

4.1 k-Minimum Length Sum and k-Anonymity

The k-minimum diameter sum problem is introduced in the previous section to develop a $3k(1 + \log k)$-approximate algorithm. Instead, we will propose the *k-minimum length sum* problem and show how we can use the solution of the problem to achieve $2(1 + \ln 2k)$-approximation.

DEFINITION 4.1: *Let us define* $a(S)$ *as the number of attributes with multiple distinct values in a table* S *as follows:*

$$a(S) := |\{i : \exists u, v \in S, u[i] \neq v[i]\}|$$

Age	Marital status	Home country	Gender	Education
20∼29	Single	USA	Female	Master
20∼29	Single	USA	Female	Doctor
20∼29	Single	China	Male	Master
20∼29	Single	China	Male	Doctor
20∼29	Divorce	USA	Male	Master
20∼29	Divorce	USA	Male	Doctor
30∼39	Single	USA	Male	Master
30∼39	Single	USA	Male	Doctor

(a) An employee table

Age	Marital status	Home country	Gender	Education
20∼29	Single	*	*	*
20∼29	Single	*	*	*
20∼29	Single	*	*	*
20∼29	Single	*	*	*
*	*	USA	Male	*
*	*	USA	Male	*
*	*	USA	Male	*
*	*	USA	Male	*

(b) An optimal 4-anonymized table

Figure 2: An example of 4-anonymization.

Age	Marital status	Home country	Gender	Education
20∼29	Single	USA	Female	Master
20∼29	Single	China	Male	Master
20∼29	Divorce	USA	Male	Master
30∼39	Single	USA	Male	Master

Age	Marital status	Home country	Gender	Education
20∼29	Single	USA	Female	Doctor
20∼29	Single	China	Male	Doctor
20∼29	Divorce	USA	Male	Doctor
30∼39	Single	USA	Male	Doctor

Figure 3: An optimal (4,7)-partition for 4-minimum diameter sum

where $u[i]$ and $v[i]$ are the values of i-th attribute of the tuples u and v respectively.

Note that $a(S)$ and $|S|a(S)$ are actually the number of attributes and the number of cells to convert all tuples in S to be identical with suppression. Thus, we call $a(S)$ the *suppression length* of S interchangeably.

EXAMPLE 4.2. *Consider a table $S = \{1010, 1110, 0110\}$ with 4 attributes. Since the third and fourth attribute values of the records in S are identical, $a(S) := |\{1, 2\}| = 2$. Thus, the suppression length of S is 2. The first and second attributes have 3 cells for each respectively to be suppressed. Thus, the total number of suppression cells for S is 6.* ∎

Recall that $OPT(T)$ is the minimum number of $*$'s to be inserted for the optimal solution of a k-anonymization and $ANON(S)$ is the total number of cells of the tuples in S that must be replaced with the symbol $*$ in order for all tuples in S to be identical. As we mentioned previously in Section 3.2, a k-anonymized relation of T (which is a $(k, |T| - k)$-partition) can be converted into $(k, 2k-1)$-partition without increasing the number of *'s. For a table T, we have $OPT(T) = \min_\Pi(\sum_{S\in\Pi} ANON(S))$, where the minimum is taken over all possible partitions Π of T, in which every set has at least k tuples.

Let Π^* be an optimal partition of T for k-anonymity, *i.e.* $\sum_{S\in\Pi^*} ANON(S) = OPT(T)$. For a partition Π of T, let us define $a(\Pi) = \sum_{S\in\Pi} a(S)$. Intuitively, $a(\Pi)$ represents the sum of the number of suppressing attributes over all subsets in Π. The following lemma bounds the number of cells to be suppressed for optimal k-anonymity in terms of the suppression lengths.

LEMMA 4.3. *For a relation T, we have*

$$k \cdot \min_\Pi a(\Pi) \le k \cdot a(\Pi^*) \le OPT(T) \le (2k-1) \cdot a(\Pi^*)$$

where the minimum is taken over every possible partition Π of T, in which the cardinality of every set is in the range of $[k, 2k-1]$.

Proof: For any $S \subseteq T$, by definition of $a(S)$, exactly $a(S)$ cells in each tuple must be suppressed for all tuples in S to be identical. Thus, we have $|S| \cdot a(S) = ANON(S)$. Furthermore, since we have $k \le |S| \le 2k-1$ and $OPT(T) = \sum_{S\in\Pi^*} ANON(S) = \sum_{S\in\Pi^*} |S|a(S)$, $k \cdot \min_\Pi a(\Pi) \le k \cdot a(\Pi^*) \le OPT(T) \le (2k-1) \cdot a(\Pi^*)$ holds. ∎

Let us consider the optimization problem of finding a $(k, 2k-1)$-partition Π of T such that $a(\Pi)$ is minimized. We call it the *k-minimum suppression length sum* problem. The following corollary illustrates the interesting relationship between this problem and k-anonymity.

COROLLARY 4.4. *Let $\alpha \ge 1$, and Π be a $(k, 2k-1)$-partition with suppression length sum at most α times that of the optimal solution of k-minimum suppression length sum. Then the algorithm that anonymizes each $S \in \Pi$ by setting $t[i] := *$ to each $t \in S$ for every suppression attribute i with which there exit a pair of $u, v \in S$ with $u[i] \ne v[i]$ is a 2α-approximation algorithm to optimal k-anonymity.*

Proof: Suppose a partition $\hat{\Pi}$ achieves the optimal k-minimum suppression length sum. From the proof of the previous lemma, the total number of $*$'s inserted into the tuples of T to k-anonymize by the algorithm in the above corollary is

$$\begin{aligned}
&\sum_{S\in\Pi} |S| \cdot a(S) \\
&\quad \le \sum_{S\in\Pi} (2k-1) \cdot a(S) && \text{(since } |S| \le 2k-1\text{)} \\
&\quad = (2k-1) \cdot a(\Pi) && \text{(by the definition of } a(\Pi)\text{)} \\
&\quad \le \alpha(2k-1) \cdot a(\hat{\Pi}) && \text{(since } a(\Pi) \le \alpha a(\hat{\Pi})\text{)} \\
&\quad \le \alpha(2k-1) \cdot a(\Pi^*) && \text{(since } a(\hat{\Pi}) = \min_\Pi a(\Pi) \le a(\Pi^*)\text{)} \\
&\quad \le \alpha\frac{2k-1}{k} OPT(T) && \text{(by Lemma 4.3)} \\
&\quad < 2\alpha OPT(T).
\end{aligned}$$

∎

Thus we conclude that the above corollary allows $2(1+\ln 2k)$-approximate solution for optimal k-anonymity if there is an $(1+\ln 2k)$-approximate solution for the k-minimum suppression length sum problem.

EXAMPLE 4.5**:** *Consider the table in Figure 2-(a). We want to find an optimal (4,7)-partitioning for the 4-minimum suppression length sum problem to use it for 4-anonymity. Since we should have exactly 2 subsets with the size of 4 records for each to guarantee the property of (4,7)-partitions and there cannot be any subset with 4 records whose suppression length is less than 3, the partitioning appeared in Figure 4 is an optimal (4,7)-partitioning for 4-minimum suppression length sum. Surprisingly, the partitioning is also an optimal (4,7)-partitioning for 4-anonymity. Note that the diameter sum of the optimal partitioning for 4-anonymity in Figure 4 is* $3+3=6$ *while that of the optimal partitioning for diameter sum in Figure 3 is* $2+2=4$*. Thus, we can see that suppression length sum captures the characteristics of k-anonymity better than diameter sum.* ∎

4.2 APPROX-NAIVE: $2(1+\ln 2k)$-approximation

We will discuss how an $(1+\ln 2k)$-approximate solution for the k-minimum suppression length sum problem can be constructed and how to use the solution to find a $2(1+\ln 2k)$-approximation of k-anonymity. Our approximation algorithm for k-minimum suppression length sum consists of the following two steps:

1. We generate a $(k, 2k\text{-}1)$-cover whose suppression length sum is at most $(1+\ln 2k)$ times that of the optimal solution.
2. Convert the obtained $(k, 2k\text{-}1)$-cover into a $(k, 2k\text{-}1)$-partition without increasing in the suppression length sum.

Computation of a $(k, 2k$-1$)$-Cover: To compute a $(k, 2k$-1$)$-cover for the k-minimum suppression length sum problem, we run a well-known greedy approximate algorithm for the set cover problem in [7] on the collection F which denotes all possible subsets of a table T with cardinalities in the range of $[k, 2k\text{-}1]$. We call this algorithm *GEN-COVER* and present the pseudo-code in Figure 5.

The algorithm works as follows. The set D contains the covered elements in T at each stage. The set Π keeps the cover being constructed so far. The greedy selection is done in lines 3-5 where a subset S is chosen so that its suppression length is as small as possible per uncovered record in S. $r(S)$ is used to measure the suppression length per uncovered record here. After S is chosen, its elements are added to D and S is inserted in Π. Since every selected subset $S \in F$ must contain an element in T-D before D is unioned with S in line 6, the while-loop in lines 2–8 takes $O(|T|)$ iterations. Therefore, *GEN-COVER* returns a $(k, 2k\text{-}1)$-cover, in which every subset is of cardinality between k and $2k$-1, and for every $t \in T$, there exists an $S \in \Pi$ containing t.

Adapting the analysis of the greedy approximate algorithm for the set cover problem in [6] taking into account that the cardinalities of all subsets in its input F are at most $2k$-1, the computed cover Π is an $(1+\ln 2k)$-approximation to the k-minimum suppression length sum. Since we can choose at most $|T|$ sets from F, *GEN-COVER* takes $O(|F||T|)$ time.

Procedure GEN-COVER(T, F)
// T: a table
// F: a set of subsets of T
begin
// Π: a current cover
// D: currently covered tuples in T
1. $\Pi := \emptyset, D := \emptyset$
2. **while** $(D \neq T)$ **do** {
3. **for each** $S \in F$ **do**
4. Compute the ratio $r(S) = \frac{a(S)}{|S \cap (T-D)|}$
5. Choose an S with the minimum $r(S)$
6. $D := D \cup S$
7. $\Pi := \Pi \cup \{S\}$
8. }
9. **return** Π
end

Figure 5: The GEN-COVER

Procedure CONVERT(Π)
// Π: $(k, 2k-1)$-cover
begin
1. **for each** $t \in T$ {
2. Let $\tilde{\Pi}$ be a list of pointers to every $S_i \in \Pi$ s.t. $t \in S_i$
3. **while** $\tilde{\Pi}$ contains at least two subsets {
4. Let S_i, S_j be the first two subsets in $\tilde{\Pi}$
5. **if** $|S_i| > k$ or $|S_j| > k$
6. Remove t from the larger set
7. Remove the pointer to the larger set from $\tilde{\Pi}$
8. **else** //i.e. $|S_i| = |S_j| = k$.
9. Replace S_i and S_j with $S_i \cup S_j$ in Π
10. Remove the pointers to S_i and S_j from $\tilde{\Pi}$
11. and insert the pointer to $S_i \cup S_j$ to $\tilde{\Pi}$
12. }
13. }
14. //Now, Π becomes a $(k, 2k-1)$-partition
15. **return** Π.
end

Figure 6: The CONVERT

EXAMPLE 4.6**:** *To see how* GEN-COVER *works, consider the table in Figure 2-(a).* GEN-COVER *iteratively chooses a subset with the minimum* $r(S)$*. In the first iteration, every subset* $S \in F$ *with* $|S| = 4$ *has* $a(S) \geq 3$ *and the subsets* $S \in F$ *with* $|S| = 5$ *or* $|S| = 6$ *have* $a(S) \geq 4$*. The subsets* $S \in F$ *with* $|S| = 7$ *have* $a(S) = 5$*. Since* $r(S) = a(S)/|S|$ *in the first iteration, the greedy decision step selects a subset with the first 6 records which has the smallest* $r(S)$ *value of* 4/6.

In the second iteration, we will select a subset containing the last remaining 2 records. Since $|S \cap (T-D)| = 2$ *and* $r(S) = \frac{a(S)}{|S \cap (T-D)|}$*, we have* $r(S) \geq 3/2$ *for the subsets* S *with* $|S| = 4$ *containing the last remaining 2 records. Actually, every* $r(S)$ *with* $|S| = 5, 6, 7$ *is larger than* 3/2 *and thus we select the subset which has the last four records in* T*. We return this (4,7)-cover which covers all 8 records in* T*.* ∎

Transformation of a $(k, 2k$-1$)$-Cover into a $(k, 2k$-1$)$-Partition: Since the $(k, 2k$-1$)$-cover Π returned by *GEN-*

Age	Marital status	Home country	Gender	Education
20∼29	Single	USA	Female	Master
20∼29	Single	USA	Female	Doctor
20∼29	Single	China	Male	Master
20∼29	Single	China	Male	Doctor

Age	Marital status	Home country	Gender	Education
20∼29	Divorce	USA	Male	Master
20∼29	Divorce	USA	Male	Doctor
30∼39	Single	USA	Male	Master
30∼39	Single	USA	Male	Doctor

Figure 4: An optimal (4,7)-partition for 4-minimum suppression length sum

COVER is not necessarily a $(k, 2k\text{-}1)$-partition of T, a pair of sets in Π may have non-empty intersection with each other. To generate a $(k, 2k\text{-}1)$-partition from the $(k, 2k\text{-}1)$-cover, we perform the procedure *CONVERT* presented in Figure 6. A tuple $t \in T$ may not belong to several S_i's in a $(k, 2k\text{-}1)$-partition. Thus, for every record $t \in T$, we first make and maintain a list $\tilde{\Pi}$ of the pointers to $S_i \in \Pi$ containing t. We repeatedly modify S_i's pointed by $\tilde{\Pi}$ as long as there exist two subsets $S_i, S_j \in \tilde{\Pi}$ as follows:

- **when** $|S_i| > k$ **or** $|S_j| > k$**:** We simply remove the tuple $t \in S_i \cap S_j$ from the larger one between S_i and S_j. Since deleting an element from a set can not increase the number of suppression attributes, the sum of the numbers of suppression attributes in the resulting partition can not be larger than that of Π.
- **when** $|S_i| = |S_j| = k$**:** We replace S_i and S_j in Π with $S_i \cup S_j$. Since there is the tuple $t \in S_i \cap S_j$, we have $k \leq |S_i \cup S_j| \leq 2k-1$. Furthermore, since $a(S_i \cup S_j) \leq a(S_i) + a(S_j)$, $a(\Pi)$ can not be increased. Thus, the number of suppression attributes in the resulting partition can not be bigger than that of Π.

Since every subset S_i selected by *GEN-COVER* should contain at least one uncovered record, *GEN-COVER* produces a cover Π with $O(\min(|F|, |T|))$ number of subsets. Since the $\tilde{\Pi}$ can take all subsets in Π and each iteration of the while-loop in lines 3-12 of *CONVERT* reduces the size of $\tilde{\Pi}$ by one, the while-loop iterates $O(\min(|F|, |T|))$ times per record. The removal of the first two elements in the list and insertion of a pointer to the list take constant time. Hence the procedure *CONVERT* takes $O(\min(|F|, |T|)|T|)$.

APPROX-NAIVE: The $2(1 + \ln 2k)$-approximate algorithm *APPROX-NAIVE* for k-anonymity is presented in Figure 7. It invokes *GEN-COVER* and *CONVERT* to produce an $(1+\ln 2k)$-approximation of k-minimum suppression length sum. It then generates a $2(1 + \ln 2k)$-approximation to k-anonymity by Corollary 4.4. Since the time complexities of *GEN-COVER* and *CONVERT* are $O(|F||T|)$ and $O(\min(|F|, |T|)|T|)$ respectively, *APPROX-NAIVE* takes $O(|F||T|)$ time. Since there are $O(\binom{T}{2k-1})$ sets in F, we can alternatively say that *GEN-COVER* takes $O(|T|^{2k})$ time.

THEOREM 4.7**:** *k-anonymity has a $2(1+\ln 2k)$-approximation that runs in $O(|F||T|)$ time.*

4.3 APPROX-FQ: Approximation with Frequent Itemsets

To improve *APPROX-NAIVE*, we develop the procedure *GEN-COVER-FQ* which runs faster by restricting the size

Procedure APPROX-NAIVE(T, F)
begin
1. Π := GEN-COVER(T,F).
2. Π := CONVERT(II)
3. **for each** $S \in \Pi$
4. Insert the minimum number of *'s to the tuples in S
5. s.t. all tuples in S are identical
6. **return** Π

end

Figure 7: The APPROX-NAIVE

of the collection F to *GEN-COVER* while still guaranteeing the approximation ratio of $2(1 + \ln 2k)$. We call the modified algorithm *APPROX-FQ* which invokes *GEN-COVER-FQ* instead of *GEN-COVER* in *APPROX-NAIVE*.

Consider a set F that is a collection of subsets $S_i \subseteq T$ with $k \leq |S_i| \leq 2k-1$ for $i = 1, 2, \ldots, |F|$. Let m be the total number of the quasi-identifier attributes in the relation T. If the number of suppression attributes of $S_i \in F$ is p (i.e. $a(S_i) = p$), every $t \in S_i$ has identical values in the other $(m-p)$ attributes. Without loss of generality, we can assume that the attributes $A_1, A_2, \cdots, A_p$ are suppression attributes of S_i. The *representative* of S_i is defined as the itemset consisting of only the values in the other $(m-p)$ attributes $A_{p+1}, A_2, \cdots, A_m$ in S_i. We let r_{S_i} denote the representative of S_i.

There is a close relationship between the representative of S_i and frequent itemsets in association rule mining[4, 8]. The *support* of an itemset is defined as the number of tuples in T that contains the same values for the attributes appearing in the the itemset. Given the minimum support of k, an itemset is called *frequent* if the support of the itemset is at least k. Thus, the representative r_{S_i} of an S_i in F has the support of at least k in the tuples of T, since S_i has at least k records and every record in S_i already contains r_{S_i}. Thus, the frequent itemsets with the minimum support of k in T are desirable candidates for F given to *GEN-COVER*.

Let us define T_S:=$\{t : t \in T$ contains the representative r_S of $S\}$ for a subset $S \subseteq T$. In other words, T_S is the set of all records in T containing the representative of $S \subseteq T$. Note that the representative r_S of T_S and S are the same and $S \subseteq T_S$ always holds for a subset $S \subseteq T$. Moreover, $|T_S|$ is the support of the representative r_S of S based in the table T.

EXAMPLE 4.8**:** *Consider the table T with 4 records in Figure 1. We represent each tuple in T with t_i for $i = 1, 2, 3, 4$ according to the order appearing from top in the table. For every subset S_i of T with $|S_i| = 2$, its representative r_{S_i} and the set of records T_{S_i} containing r_{S_i} in T are listed in*

S_i	Representative r_{S_i}	T_{S_i}
S_1=$\{t_1, t_2\}$	$\emptyset$	T_{S_1}=$\{t_1, t_2, t_3, t_4\}$
S_2=$\{t_1, t_3\}$	{20∼29, Single, USA}	T_{S_2}=$\{t_1, t_3\}$
S_3=$\{t_1, t_4\}$	$\emptyset$	T_{S_3}=$\{t_1, t_2, t_3, t_4\}$
S_4=$\{t_2, t_3\}$	{Female}	T_{S_4}=$\{t_2, t_3, t_4\}$
S_5=$\{t_2, t_4\}$	{30∼39, Female}	T_{S_5}=$\{t_2, t_4\}$
S_6=$\{t_3, t_4\}$	{Female}	T_{S_6}=$\{t_2, t_3, t_4\}$

Figure 8: An example of r_{S_i} and T_{S_i}

the Figure 8. Since the first two tuples t_1={ 20∼29, Single, USA, Male } and t_2 = {30∼39, Divorce, China, Female } in T do not have any attribute with the same value, the representative r_{S_1} of the subset $S_1 = \{t_1, t_2\}$ is $\emptyset$ and thus we let $T_{S_1} = \{t_1, t_2, t_3, t_4\} = T$. The subset $S_2 = \{t_1, t_3\}$ consisting of the first and third tuples t_1={ 20∼29, Single, USA, Male } and t_3 = {20∼29, Single, USA, Female } in T has the representative r_{S_2}={20∼29, Single, USA} and thus $T_{S_2} = \{t_1, t_3\}$. ∎

We can discover all frequent itemsets of a table T efficiently by running association rule mining algorithms such as the ones in [4, 8]. The numeric attributes are discretized so that the traditional association rule mining algorithms for market basket data can be still utilized to compute frequent itemsets.

Let f_i represent a frequent itemset in the table T with the minimum support of k and let $S(f_i)$ be the set of records in T that contain f_i. Furthermore, let F_{FQ} be the set consisting of all $T_{S(f_i)}$'s and T. Since the representative of a set S may be an empty set, we insert T into F_{FQ}. Thus, every $S \in F$ always has T_S such that $T_S \in F_{FQ}$.

Supplying F_{FQ} instead of F to *GEN-COVER* as input changes its time complexity to be $O(|F_{FQ}||T|)$. However, since the size of any $T_S \in F_{FQ}$ may be larger than $2k$-1, we cannot apply Corollary 4.4 any more. To achieve the same approximation ratio by Corollary 4.4, we modify *GEN-COVER* so that whenever the greedily selected subset S has the size larger than $2k$-1, we create a subset selecting only $2k$-1 records arbitrarily in S and insert the subset to Π. To take this modification into account for greedy selection process, if the size of $S \cap (T - D)$ is larger than $2k - 1$, we divide $a(S)$ by $2k-1$ for $r(S)$. In other words, we use $r(S) = a(S)/\min\{|S \cap (T-D)|, 2k-1\}$ instead of $r(S) = a(S)/|S \cap (T-D)|$ used in *GEN-COVER*. We call this procedure *GEN-COVER-FQ* that is presented in Figure 9. The procedure works as follows.

In each iteration, if the size of chosen subset S with minimum $r(S)$ is less than $2k$, we have $k \leq |S| \leq 2k - 1$ and we thus do the same thing (line 7) as *GEN-COVER* does. However, when $|S| > 2k-1$, we can split into two cases and perform the following:

- **when** $|S \cap (T - D)| > 2k - 1$**:** We produce a subset S^R by selecting only $2k - 1$ elements arbitrarily from $S \cap (T - D)$ and put it into the cover Π found so far.
- **when** $|S \cap (T-D)| \leq 2k-1$**:** We let $S^R = S \cap (T-D)$. If $k \leq |S \cap (T - D)|$, put S^R into Π. Otherwise (i.e. $|S \cap (T - D)| < k$), we add previously covered tuples in S (i.e. elements in $S \cap D$) to S^R in addition to $S \cap (T - D)$ until we get $|S^R| = k$ and put S^R into Π.

Procedure GEN-COVER-FQ(T, F_{FQ})
begin
1. $\Pi := \emptyset, D := \emptyset$
2. **while** $(D \neq T)$ {
3. **for each** $S \in F_{FQ}$ **do**
4. Compute the ratio $r(S) = \frac{a(S)}{\min\{|S \cap (T-D)|, 2k-1\}}$
5. Choose an S such that $r(S)$ is minimum
6. **if** $|S| \leq 2k - 1$
7. $S^R := S$
8. **else if** $|S \cap (T - D)| > 2k - 1$
9. Choose an $S^R \subseteq S \cap (T - D)$ s.t. $|S^R| = 2k - 1$
10. **else**
11. Choose an $S^R \subseteq S$ s.t. $S^R \supseteq S \cap (T - D)$ and $|S^R| = \max\{k, |S \cap (T - D)|\}$
12. $D := D \cup S^R$
13. $\Pi := \Pi \cup \{S^R\}$
14. }
15. **return** Π
end

Figure 9: The GEN-COVER-FQ

We next show that *GEN-COVER-FQ* achieves the same approximation ratio of *GEN-COVER*. We begin with the following lemma.

LEMMA 4.9**.** *If S and S^R are those selected in each iteration of* GEN-COVER-FQ*, we always have $r(S) = r(S^R)$*

Proof: According to the function $r(S)$ in *GEN-COVER-FQ*, we have

$$r(S) = \frac{a(S)}{\min\{|S \cap (T - D)|, 2k - 1\}}$$

and

$$r(S^R) = \frac{a(S^R)}{\min\{|S^R \cap (T - D)|, 2k - 1\}} = \frac{a(S^R)}{|S^R \cap (T - D)|}. \quad (\text{since } |S^R| \leq 2k - 1)$$

To show $r(S) = r(S^R)$, we will first prove that $\min\{|S \cap (T - D)|, 2k - 1\} = |S^R \cap (T - D)|$ and next show $a(S) = a(S^R)$.

In each step of *GEN-COVER-FQ*, we compute an S^R differently depending on $r(S)$, $|S|$ and $|S \cap (T - D)|$. We can show $\min\{|S \cap (T - D)|, 2k - 1\} = |S^R \cap (T - D)|$ for each case below:

- **when** $|S \cap (T - D)| > 2k - 1$**:** Since S^R has $2k$-1 elements from $S \cap (T - D)$, we have $\min\{|S \cap (T - D)|, 2k - 1\} = 2k - 1 = |S^R \cap (T - D)|$.
- **when** $k \leq |S \cap (T - D)| \leq 2k - 1$**:** Because S^R is the same as the newly covered elements by S (i.e. $S \cap (T - D)$), we have $\min\{|S \cap (T - D)|, 2k - 1\} = |S \cap (T - D)| = |S^R \cap (T - D)|$.
- **when** $|S \cap (T - D)| < k - 1$**:** Since we make S^R by adding previously covered elements from S to $S \cap (T - D)$ until we have $|S^R| = k$, we have $\min\{|S \cap (T - D)|, 2k - 1\} = |S \cap (T - D)| = |S^R \cap (T - D)|$.

We will prove next that $a(S) = a(S^R)$ using contradiction method.

Let us assume that $a(S) \neq a(S^R)$. Since $S^R \subset S$ and $a(S) \neq a(S^R)$, it must be $a(S) > a(S^R)$. Larger suppression length means that the number of suppression attributes to make all records in the subset be identical is larger. Thus, the number of identical attributes in the records of S is actually smaller than that of S^R. In other words, the size of the representative of S is smaller than that of S^R. Note that S^R is made to have at least k records, and thus both S and S^R have at least k records. For every frequent itemset f_i, we generate the set consisting of the records containing f_i in T and put it to F_{FQ}. Thus, there should exist a set $S' \in F_{FQ}$ that is exactly the same as the representative r_{S^R} (i.e. $a(S') = a(S^R)$). Since S' contains all records containing r_{S^R} from T, we have $S^R \subseteq S'$ and $|S'| \geq |S^R| \geq k$. Furthermore, since $a(S) > a(S^R)$, we have $S' \neq S$. It implies that

$$r(S') = \frac{a(S')}{\min\{|S' \cap (T-D)|, 2k-1\}} \leq \frac{a(S^R)}{|S^R \cap (T-D)|} < \frac{a(S)}{\min\{|S \cap (T-D)|, 2k-1\}} = r(S).$$

However, in this case, *GEN-COVER-FQ* should have selected S' instead of S, and it contradicts that S is selected in this iteration. Hence, we can conclude that $a(S) = a(S^R)$.

Since we have shown $\min\{|S \cap (T-D)|, 2k-1\} = |S^R \cap (T-D)|$ and $a(S) = a(S^R)$, we conclude that $r(S) = r(S^R)$ always holds in each iteration of *GEN-COVER-FQ*. ∎

Lemma 4.9 states that we always select an S and an S^R such that $S^R \subseteq S$ and $r(S) = r(S^R)$ in each iteration of *GEN-COVER-FQ*. Actually *GEN-COVER* and *GEN-COVER-FQ* may produce different solutions for a given input F and F_{FQ} respectively, since there may exist $S_1, S_2, ...$ s.t. $r(S_1) = r(S_2) = ...$ in each iteration and both algorithms arbitrarily select one of them. Let SOL and SOL_{FQ} be the sets of possible solutions for *GEN-COVER* and *GEN-COVER-FQ*, respectively. If we show that $\text{SOL}_{FQ} \subseteq \text{SOL}$, it implies that the solution by *GEN-COVER-FQ* always have the same approximation ratio $(1 + \ln 2k)$ of *GEN-COVER*. We indeed have $\text{SOL}_{FQ} \subseteq \text{SOL}$ as the following lemma illustrates.

LEMMA 4.10.: *The possible solutions obtained by* GEN-COVER-FQ *are always possible solutions generated by* GEN-COVER.

Proof: Let SOL_{FQ} and SOL be the set of all possible solutions obtained by *GEN-COVER-FQ* with F_{FQ} and *GEN-COVER* with F respectively. Consider any solution $\Pi_{FQ} \in \text{SOL}_{FQ}$. We will show that there exists a solution $\Pi \in \text{SOL}$ such that $\Pi_{FQ} = \Pi$.

Let $\Pi_{FQ} = \{S_1'^R, S_2'^R, ..., S_m'^R\}$ where the subscript denotes the selection order by *GEN-COVER-FQ*. $S_i'^R$ is generated in lines 6–11 from S_i' selected greedily among the subsets in F_{FQ} in lines 3–5. We claim that there exists a solution $\Pi = \{S_1, S_2, ..., S_m\} \in \text{SOL}$ where the subscript denotes the selection order and we have $\{S_1, ..., S_i\} = \{S_1'^R, ..., S_i'^R\}$ for $i = 1, \ldots, m$. If this claim is true, we can conclude $\Pi_{FQ} = \Pi \in \text{SOL}$ with $i = m$. We will prove this claim by mathematical induction.

When $i = 0$, we have $\emptyset = \emptyset$, and thus the base case of our inductive proof holds. Assume that there exists a solution $\Pi \in \text{SOL}$ such that $\{S_1, ..., S_i\} = \{S_1'^R, ..., S_i'^R\}$ for $i = 1, 2, ..., j-1$. We want to show that it holds for $i = j$ too. Consider the case for $i = j$. If $S_j = S_j'^R$, the proof is done. If $S_j \neq S_j'^R$, we will show that there exists another solution $\Pi' \in \text{SOL}$ which contains $S_1, ..., S_{j-1}$ and $S_j'^R$. Assume that there does not exist such a solution. Then there are two cases below:

- **when** $r(S_j) < r(S_j'^R)$**:** By the definition of F_{FQ}, there exists $S'' \in F_{FQ}$ s.t. $S'' \supseteq S_j$, and S'' and S_j have the same representative. Then we have $r(S'') \leq r(S_j) < r(S_j'^R)$ at the j-th iteration of *GEN-COVER-FQ*. Since S_j' that is the greedily selected subset at the j-th iteration, we always have $r(S_j') = r(S_j'^R)$ by Lemma 4.9 and thus $r(S'') < r(S_j')$ holds. However, then we should have selected S'' instead of S_j' at the j-th iteration and this contradicts that $r(S_j')$ is minimum at j-th iteration of *GEN-COVER-FQ*.
- **when** $r(S_j) > r(S_j'^R)$**:** Since $k \leq |S_j'^R| \leq 2k-1$ and F is the collection of all subsets of T with cardinalities between k and $2k-1$, we have $S_j'^R \in F$. Then we should have selected $S_j'^R$ instead of S_j. This contradicts that $r(S_j)$ is minimum at j-th iteration of *GEN-COVER*.

Thus, we always have $r(S_j) = r(S_j'^R)$. Since we have $k \leq |S_j'^R| \leq 2k-1$, we must have $S_j'^R \in F$. Then, if *GEN-COVER* selected $\{S_1, ..., S_{j-1}\}$ until (j-1)-th iteration, it can select $S_j'^R$ instead of S_j since we have $r(S_j) = r(S_j'^R)$ at j-th iteration. It results in another solution. That is, there exists a solution $\Pi' \in \text{SOL}$ such that it contains $S_1, ..., S_{j-1}$ and $S_j'^R$. ∎

Therefore, we conclude the following:

THEOREM 4.11**:** *k-anonymity has a $2(1+\ln 2k)$-approximation that runs in $O(|F_{FQ}||T|)$ time.*

4.4 APPROX-CF: Approximation with Closed Frequent Itemsets

Since the number of frequent itemsets may be very large, some proposals have been made to generate only a concise representation of frequent itemsets quickly from which all frequent itemsets can be produced. We will introduce one of the concise representations, called *closed frequent itemset* proposed in [13] and provide the closed frequent itemsets only as input to *GEN-COVER-FQ*. We will call this modified approximation algorithm *APPROX-CF*.

DEFINITION 4.12.: *A frequent itemset is called* closed *if there exists no proper superset with the same support .*

EXAMPLE 4.13**:** *Let us compute the frequent itemsets and the closed frequent itemsets for the table in Figure 1. Suppose the minimum support is 2. The frequent itemsets and frequent closed itemsets are listed in Figure 10. The closed frequent itemsets are represented with bold cases. The sets S_1', S_8' and S_{10}' are the closed frequent itemsets. As we can see from this example, the number of closed frequent itemsets can be quite smaller than that of frequent itemsets.* ∎

Although all frequent itemsets can essentially be closed, in practice, our experiences show that a lot of frequent itemsets are not actually closed frequent itemsets. Furthermore, we can also show with the following lemma that the use of closed frequent itemsets F_{CF} instead of frequent itemsets F_{FQ} as input to *GEN-COVER-FQ* still guarantees the same approximation ratio.

S'_i	Frequent itemset
$\mathbf{S'_1 = \{t_1, t_3\}}$	**{20∼29, Single, USA}**
$S'_2 = \{t_1, t_3\}$	{20∼29, Single}
$S'_3 = \{t_1, t_3\}$	{20∼29, USA}
$S'_4 = \{t_1, t_3\}$	{Single, USA}
$S'_5 = \{t_1, t_3\}$	{20∼29}
$S'_6 = \{t_1, t_3\}$	{Single}
$S'_7 = \{t_1, t_3\}$	{USA}
$\mathbf{S'_8 = \{t_2, t_4\}}$	**{30∼39, Female}**
$S'_9 = \{t_2, t_4\}$	{30∼39}
$\mathbf{S'_{10} = \{t_2, t_3, t_4\}}$	**{Female}**

Figure 10: An example of frequent itemsets and closed frequent itemsets

THEOREM 4.14.: *Let F_{CF} be the set of all subsets in T whose representatives are closed frequent itemsets. Using F_{CF} instead of F_{FQ} for* GEN-COVER-FQ *produces a solution whose approximation ratio is still the same as the one using F_{FQ}.*

Proof: Consider a set $S' \in F_{FQ} - F_{CF}$. Since S' is not a closed frequent itemset, there exists a set $S'_C \in F_{CF}$ which contains all tuples in S' and whose representative is a superset of the representative of S' due to the definition of closed frequent itemset. Then, in each iteration of *GEN-COVER-FQ*, since we have $r(S') = \frac{a(S')}{\min\{|S' \cap (T-D)|, 2k-1\}} > \frac{a(S'_C)}{\min\{|S'_C \cap (T-D)|, 2k-1\}} = r(S'_C)$, we never select S'. Thus, any element in $F_{FQ} - F_{CF}$ is never selected by *GEN-COVER-FQ* even though F_{FQ} is given as input. Thus using F_{CF} does not affect the approximation ratio. ∎

EXAMPLE 4.15**:** *Consider the table T in the Figure 1 and the closed frequent itemsets listed in Example 4.13. We will show the detailed steps of applying both* GEN-COVER-FQ *and* GEN-COVER-CF *to the table T to illustrate the efficiency of using closed frequent itemsets. As shown in Figure 10, we have 10 frequent itemsets (i.e. $F_{FQ} = \{S'_1, \cdots, S'_{10}\} \cup \{T\}$) and 3 closed frequent itemsets (i.e. $F_{CF} = \{S'_1, S'_8, S'_{10}\} \cup \{T\}$).*

GEN-COVER-FQ *greedily selects S with the minimum value of $r(S)$. Here, S'_1 has the minimum $r(S)$ for both F_{FQ} and F_{CF}, and thus selected. Then, we exclude the selected records t_1 and t_3 in S'_1 to recompute $r(S)$ for every candidate S in next iterations. In the next iteration, S'_8 has the minimum of the recomputed $r(S)$. Since all records in T are covered, we terminate. We can see that only closed frequent itemsets are selected in* GEN-COVER-FQ *with F_{FQ}. Since this cover of $\{S'_1, S'_8\}$ is also a partitioning of T,* CONVERT *has nothing to do and the table is 2-anonymized by this partitioning.* ∎

Trade-off between Time Complexity and Approximation Ratio: Since our approximate algorithms take $O(|F||T|)$ time, $|F|$ is is an important factor in running time. We next show how we can reduce $|F|$ with sacrificing the approximation ratio.

For a solution Π obtained by our approximation algorithms, consider an $S \in \Pi$ whose suppression length is $a(S)$. When a given table T is anonymized using Π, if all cells in S are suppressed instead of suppressing only the cells for the suppression attributes of S, the number of suppressed cells can be increased at most $m|S|$ where m is the total number of quasi-identifier attributes. If we let $a(S) = m/\beta$ with $\beta \geq 1$, the increasing ratio of suppressed cells is at most $m|S|/(a(S)|S|) = m/a(S) = m\beta/m = \beta$. Hence, if we restrict the elements of F to be the subsets whose suppression lengths is less than m/β, the approximation ratio of the solution can be increased at most β times. For example, consider the case when $\beta = 3$ and $m = 9$. Then, only the subsets whose suppression lengths are less than 3 are included in F. The number of such subsets would very small compared with the number of subsets initially in F since the sizes of most of frequent itemsets is typically 2 or 3 and thus the number of suppression attributes of them are larger than 3. Our experimental results confirm that with the reasonable values of β, the approximation algorithms speed up significantly sacrificing the quality of approximation ratio with small amount.

Procedure OPT-LB(T, F_{CF})
begin
// Π: a current partition
// D: currently covered tuples in T
1. $\Pi := \emptyset, D := \emptyset$
2. sort F_{CF} with counting sort in increasing order of $a(S)$
3. **while** $(D \neq T)$ **do** {
4. delete an S with the minimum $a(S)$ from F
5. $S' := S - D$
6. **if** $|S'| > 0$ {
7. $D := D \cup S'$
8. $\Pi := \Pi \cup \{S'\}$
9. }
10. }
11. **for each** $S' \in \Pi$
12. Insert the minimum number of *'s to the tuples in S'
13. s.t. all tuples in S' are identical
14. **return** Π
end

Figure 11: The *OPT-LB*

4.5 APPROX-DR: Direct Approximation

Traditional approximation algorithms and our proposed approximation algorithms use two phases such that they first find an approximate solution for a closely related problem of k-anonymity and then convert it to a solution for the k-anonymity problem. For example, the diameter sum problem was used as a closely relation problem in [12]. However, even an optimal solution for the diameter sum problem can be up to k times worse than the optimal solution for the k-anonymity problem. Meanwhile, our approximation algorithms proposed in this paper used suppression length sum problem.

Instead of using the two phase approaches, we will next take a different approach which directly computes an approximation solution for k-anonymity itself without using a solution for a closely related problem.

We modify *GEN-COVER* as follows: (1) The closed frequent itemsets F_{CF} is given as input instead of F. (2) We choose a subset S in lines 3-5 considering the value of the minimum suppression length only as long as there exists at least one uncovered record in the subset. (3) When we put

the currently chosen subset to the partition Π, we make the subset to have only newly covered records and put it to the partition Π found so far. We call this algorithm *OPT-LB* and presented it in Figure 11.

Let the cost of a partition Π produced by *OPT-LB* be the sum of the products of the suppression length and the size of subset $S' \in \Pi$. Each record contributes to the cost of the partition Π an amount of suppression length of the subset $S \in \Pi$ which covered the record first when selected.

Since the subsets of the partition Π consist of only newly covered records, a subset in the partition found may have less than k records and violates k-anonymity. However, notice that the partition Π provides a lower bound of the minimum number of suppressed cells for the k-anonymity problem as the following lemma illustrates.

LEMMA 4.16**:** *The number of suppressed cells in the partition produced by* OPT-LB *is at most the number of suppressed cells of any partition satisfying k-anonymity.*

Proof: Let Π be a partition satisfying k-anonymity and S be an element of Π. Then, there exists a subset S_{CF} in F_{CF} such that $S \subseteq S_{CF}$ and $a(S) = a(S_{CF})$ since $|S| \geq k$ and the representative of S is always a closed itemset as stated in the proof of Theorem 4.14. Because each record in T is suppressed by the suppression length of the subset to which it belongs and S is an arbitrary subset in Π, if we show that each record in S belongs to the subset with at most $a(S)$ as suppression length by *OPT-LB*, the proof is done.

Since for each record *OPT-LB* chooses a subset with the minimum suppression length among all subsets containing the record in F_{CF}, if the records in S do not belong to S_{CF} by *OPT-LB*, then the records must belong to subsets with less suppression length than S_{CF}. Hence, the number of suppressed cells by *OPT-LB* is at most that by Π. ∎

Since our objective is to find a partition which k-anonymizes a table, we will propose a new algorithm *APPROX-DR* by exploiting *OPT-LB*. To get *APPROX-DR*, we replace only the condition of $|S'| > 0$ in line 6 of *OPT-LB* with the condition $|S'| \geq k$. *APPROX-DR* is a simple greedy algorithm which chooses a subset S with the minimum $a(S)$, of which the number of the newly covered records is at least k, iteratively until all records are covered. Although unfortunately we can not provide any guarantee of approximation ratio for *APPROX-DR*, our experimental results show that it is very effective for approximating the k-anonymity problem.

The time complexity of *APPROX-DR* is $O(|F_{CF}||T|)$. In order to find S with the minimum $a(S)$ in constant time in each iteration, *APPROX-DR* sorts the subsets in F_{CF} in order of increasing $a(S)$ with the counting sort in [6] in line 2. Since $a(S)$ is an integer between 0 and the number of quasi-identifier attributes (i.e. small number), we can use a counting sort which takes $O(|F_{CF}|)$ time. The lines 3-10 in *APPROX-DR* choose a subset S with the minimum $a(S)$ repeatedly in constant time due to the above sorting until all records are covered. Since all subsets in F_{CF} are chosen at most once, while-loop iterates at most $O(|F_{CF}|)$ times. In line 5, we make S' be the set of the newly covered records by S, and in line 6-9 we test whether $|S'|$ is at least k or not. These two steps take $O(|T|)$ time and thus the while-loop takes overall $O(|F_{CF}||T|)$ time. Finally, we can easily see that the steps in line 11-13 takes $O(|T|)$. Therefore, the overall time complexity is $O(|F_{CF}||T|)$.

5. EXPERIMENTS

We empirically compared the performance of our approximate algorithms with the approximate algorithms for local recoding method in [3, 12] as representatives of existing algorithms for k-anonymity.

All experiments reported in this section were performed on a Pentium-4 2.8 GHz machine with 1GB of main memory, running Linux operating system. All the methods were implemented using GCC compiler of Version 2.95.3.

5.1 Algorithms Implemented

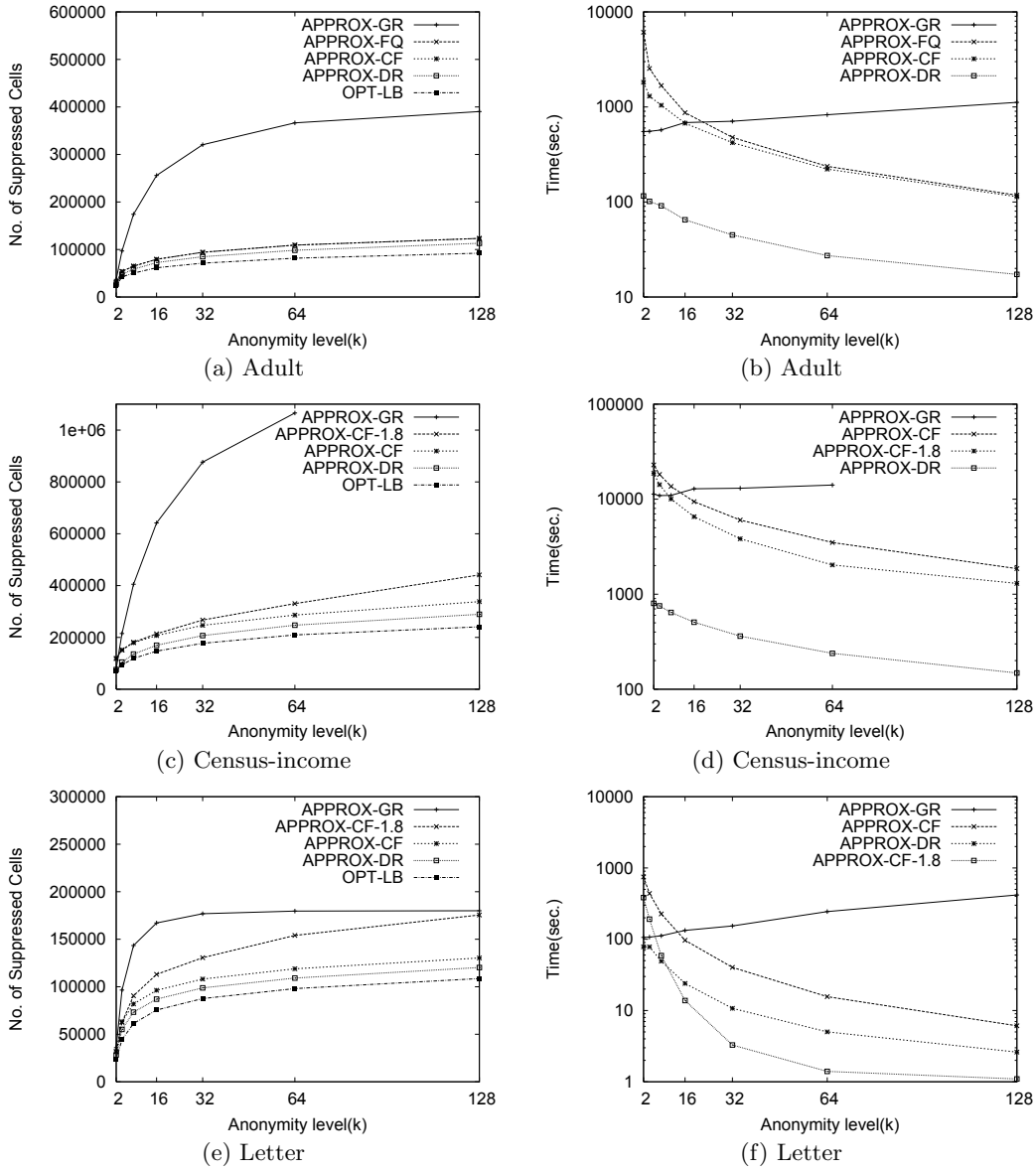
- **APPROX-DS:** This is our implementation of an $O(k \log m)$ approximation algorithm for k-anonymity in [12]. The $O(k \log k)$ approximation algorithm is not implemented because of its prohibitive time complexity.
- **APPROX-GR:** It represents the implementation of the $O(k)$ approximation algorithm in [3].
- **APPROX-NAIVE:** It is the implementation of our $O(\log k)$-approximation algorithm *APPROX-NAIVE* using F introduced in Section 4.2.
- **APPROX-FQ:** It is our $O(\log k)$-approximation algorithm *APPROX-FQ* using F_{FQ} provided in Section 4.3.
- **APPROX-CF:** It is our $O(\log k)$-approximation algorithm *APPROX-CF* using F_{CF} illustrated in Section 4.4.
- **APPROX-DR:** This is the implementation of our approximate algorithm *APPROX-DR* shown in Section 4.5.
- **OPT-LB:** It represents our implementation of *OPT-LB* presented in Section 4.5 which is not an algorithm for the k-anonymity problem, but can show the lower bound on the number of suppressed cells for the optimal solution for the k-anonymity problem.

Since all of our algorithms need to compute frequent itemsets, we added the computing time to the running time of the results in the paper.

5.2 Real-life Data Sets

We used several data sets from the UCI Machine Learning Repository[14]. They include Adult, Census-income, COIL2000, Nursery and Letter data sets. Due to the lack of space, we show the result of the following 3 data sets only.

- **Adult:** The Adult data contains 45,222 tuples from US Census data after the tuples with missing values are removed.
- **Census-income:** Census-income data contains 196,130 tuples after deleting tuples with missing values. Since Census-income database is similar to Adult data, we chose the set of the same quasi-identifier attributes with Adult as quasi-identifier.
- **Letter:** Letter recognition data is based on various fonts. It consists of 20,000 tuples and 17 attributes, and we selected the first 9 attributes as quasi-identifier.



(a) Adult (b) Adult

(c) Census-income (d) Census-income

(e) Letter (f) Letter

Figure 12: Varying k

5.3 Performance Results

To study how the approximate algorithms perform, we ran the implementations of the approximate algorithms. However, *APPROX-DS* could not finish within a day and *APPROX-NAIVE* could not run due to out of memory. Thus we do not report the result of both algorithms. Furthermore, since *OPT-LB* is not an algorithm for k-anonymity, but can show the lower bound on the number of suppressed cells for optimal k-anonymity, we plot the number of suppressed cells only and do not present the execution times.

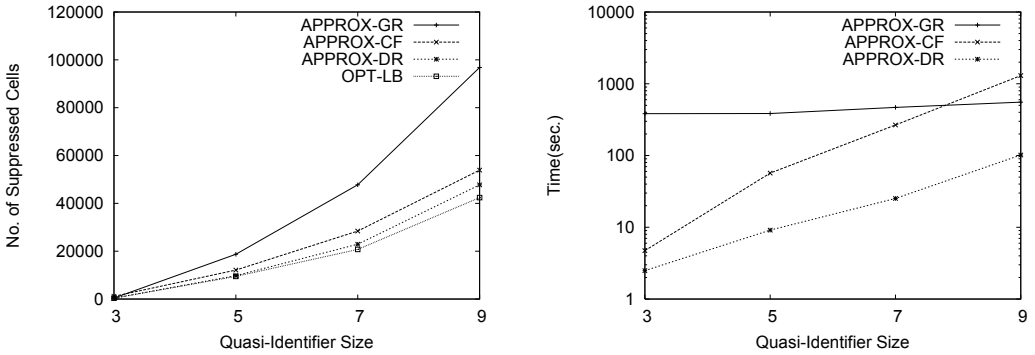
Varying k: We varied k from 2 to 128 and presented the result in Figure 12. For Adult data set, the numbers of suppressed cells by *APPROX-FQ* and *APPROX-CF* are very close as expected, but *APPROX-CF* runs faster than *APPROX-FQ*. Thus, we do not plot the results for *APPROX-FQ* for other data sets.

APPROX-DR and *APPROX-GR* are the best and worst performers respectively in terms of the number of suppressed cells. The number of suppressed cells by *APPROX-GR* was about 3 times more than those of *APPROX-CF* and *APPROX-DR*. The figure also confirms that the approximation ratios of *APPROX-FQ*, *APPROX-CF* and *APPROX-DR* are very close to the optimal solution.

Regarding the execution times, *APPROX-DR* is about 8 times faster than *APPROX-CF* and *APPROX-CF* is 10 times faster than *APPROX-GR* for large k. When k is small, *APPROX-GR* performs faster than *APPROX-CF*.

For Census-income data, the number of suppressed cells by *APPROX-GR* was three times larger than those of the other algorithms. Furthermore, due to out of memory problem, we could not run *APPROX-GR* for $k = 128$. The figure for the execution times are similar to the one for Adult data set.

We varied the value β to show the effect of β introduced



(a) Adult (b) Adult

Figure 13: Varying number of quasi-identifier attributes

in Section 4.4. We plot the graphs for *APPROX-CF-1.8* which is *APPROX-CF* with $\beta = 1.8$. The number of suppressed cells by *APPROX-CF-1.8* increases with the factor of 1.8, but the execution time becomes about 40% faster than *APPROX-CF*.

We also present the result with Letter data. As we increase k, the number of suppressed cells by *APPROX-CF-1.8* becomes worse and closer to that of *APPROX-GR*, but *APPROX-CF-1.8* runs much faster than *APPROX-DR*.

Varying Quasi-identifier size: We varied the number of quasi-identifier attributes from 3 to 9 and ran the approximate algorithms on Adult data with $k = 4$. We present the result in Figure 13. As we increase the number of quasi-identifier attributes, the number of suppressed cells becomes larger. The number of suppressed cells by *APPROX-GR* was about twice that by *APPROX-CF*

6. CONCLUSION

The k-anonymity model was introduced in order to preserve privacy against linking attack. In this model, it is required that each record in the table to be indistinguishable with k-1 other records. The problem of k-anonymizing a table with minimizing the number of suppressed cells is known to be NP-Hard. The $O(k \log k)$-approximation and $O(k)$-approximation algorithms were proposed previously.

In this paper, we proposed several approximation algorithms that guarantee $O(\log k)$-approximation ratio and outperform the traditional algorithms significantly. We also developed $O(\beta \log k)$-approximate algorithms which gracefully adjust running time according to the tolerance $\beta(\geq 1)$ of the approximation ratios. Through our experiments, we showed that our approximation algorithms perform significantly better than traditional approximation algorithms.

7. REFERENCES

[1] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *VLDB*, pages 901–909, 2005.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. In *VLDB*, 2006.

[3] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *ICDT*, pages 246–258, 2005.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.

[5] R. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *ICDE*, pages 217–228, 2005.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (Second Edition)*. McGraw Hill and MIT Press, 2001.

[7] D.S.Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[8] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD Conference*, pages 1–12, 2000.

[9] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *SIGMOD*, pages 49–60, 2005.

[10] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, page 25, 2006.

[11] A. Machanavajjhala, J. Gehrke, and D. Kifer. l-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

[12] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proc. of PODS*, 2004.

[13] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416, 1999.

[14] U.C.Irvine Machine Learning Repository. http://www.ics.uci.edu/∼mlearn/mlsummary.html.

[15] P. Samarati and L. Sweeny. Generalizing data to provide anonymity when disclosing information(abstract). In *In Proc. of ACM Symposium on Principles of Database Systems*, page 188, 1998.

[16] L. Sweeny. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowedge-based Systems*, 10(5):557–570, 2002.

[17] L. Willenborg and T. deWaal. Elements of statistical disclosure control. Springer Verlog Lecture Notes in Statistics, 2000.