Anonymizing but Deteriorating Location Databases

Tran Khanh Dang and Tuan Anh Truong

***Abstract*—The tremendous development of location-based services and mobile devices has led to an increase in location databases. Through the data mining process, valuable information can be discovered from such location databases. However, the malicious data miner or attackers may also extract private and sensitive information about the user, and this can create threats against the user location privacy. Therefore, location privacy protection becomes a key factor to the success in privacy protection for the users of location-based services. In this paper, we propose a novel approach as well as an algorithm to guarantee k-anonymity in a location database. The algorithm will maintain the association rules that have significance for the data mining process. Moreover, there may appear new significant association rules created after anonymization, they maybe affect the data mining result. Therefore, the algorithm also considers excluding new significant association rules that are created during the run of the algorithm. Theoretical analyses and experimental results with real-world datasets will confirm the practical value of our newly proposed approach.**

***Index Terms*—k-anonymity, location databases, data mining, privacy protection.**

I. Introduction

TODAY, advances in location technologies and wireless communication technologies enable the widespread development of location-based services (LBSs). When using services, the user may face with risks because the location information of the user can disclose some private information. Therefore, we need to protect the location information of the user from attacking of malefactors.

The user's location privacy should be protected in two stages. In the first stage, the location privacy should be protected at the time of using services. One popular method used in this stage is to obfuscate the user's exact location with respect to service providers in order to hide the user's location information. These solutions focus on preventing the user's location from an illegal observation at the time of service calls. We also proposed some approaches to obfuscate the user's location in [2, 3, 11, 22–25]. In the second stage, the location privacy of the user should be protected as the user's location information is stored in the database for data mining purposes. In this stage, the location information of the user will be anonymized before these data are published to other organizations or companies.

In this paper, we will focus on protecting the user's location at the second stage when the location data is stored in the database for data mining purposes. We assume that when the user uses services, he/she will provide his/her true location to service servers and the service servers will save all information about the location of the user. Then, many organizations, companies or individuals may collect these location data.

Through the data mining process, some valuable information can be obtained. However, these location data maybe disclose some private information of the user. For example, the attacker queries the database and receives results, but he also has some knowledge about the service and links the knowledge with the results to obtain some sensitive information. Therefore, we should protect these location data before they are collected by organizations, companies or individual. Fortunately, we have some techniques to protect user data before publishing these data as randomization, k-anonymity, etc. Among them, k-anonymity is an important method for privacy de-identification. The motivating factor behind the k-anonymity technique is that many attributes in the data can often be considered pseudo-identifiers which can be used in conjunction with public records in order to uniquely identify the records [1, 5].

This paper will improve the approach which was proposed in [4] and will use this improved approach to anonymize the location database to achieve an effective k-anonymous version. This approach does not use two popular techniques (generalization and suppression) because data after anonymizing by these techniques may not be significant to the data mining processes. The approach will use a technique which is called *Migrate Member technique* to anonymize the database [4]. The approach also considers the result of data mining process by maintaining association rules that are significant to the data mining process.

The rest of this paper is organized as follows: in Section II, we briefly summarize related works. Section III introduces definitions and calculating methods of crucial values for the algorithm. Next, Section IV presents the proposed algorithm to guarantee k-anonymity in location databases. Experimental results are shown in Section V. Finally, Section VI presents concluding remarks as well as future works of our approach.

Manuscript received March 14, 2012. Manuscript accepted for publication December 12, 2012.

T.K. Dang is with the Faculty of Computer Science & Engineering, HCMC University of Technology, VNUHCM, Ho Chi Minh City, Vietnam (e-mail: khanh@cse.hcmut.edu.vn).

T.A. Truong is with the Department of Information Engineering and Computer Science, University of Trento, Italy (e-mail: anhtt@cse.hcmut.edu.vn).

II. RELATED WORK

A. k-anonymity

The notion of k-anonymity, proposed by Samarati [7], is an approach to protect data from individual identification. k-anonymity is a property that models the protection of released data against possible re-identification of the respondents to which the data refer. Intuitively, k-anonymity states that each release of data must be such that every combination of values of released attributes, which are also externally available and therefore exploitable for linking, can be indistinctly matched to at least k respondents.

k-anonymous data mining has been recently introduced as an approach to ensuring privacy-preservation when releasing data mining results [7]. With this approach, the author defined the set of attributes whose values may be used, possibly together with external information, to re-identify the user's data. These attributes are called *Quasi-Identifiers* (QI). For example, even if data about the ZIP code, date of birth and sex do not explicitly identify an individual, they may be linked to external information (for example: public voter lists) to obtain name, address. Intuitively, the greater the value of k, the better the protection of privacy. However, if value of k is too great, data quality for the data mining process is not good. Therefore, how to keep the balance between data privacy and data quality is an important factor in privacy preserving in data mining. In this paper, we propose an algorithm not only to anonymize the location database but also to consider the result of data mining processes.

B. k-Anonymity Techniques, M3AR algorithm and problems

Today, we have some algorithms which guarantee k-anonymity in a database. These algorithms usually use one of two techniques: Generalization or Suppression. In the method of generalization, attribute values are generalized to a range in order to reduce the granularity of representation [10]. For example, date of birth could be generalized to a range such as year of birth, so as to reduce the risk of identification. In the method of suppression, the value of an attribute could be removed completely to guarantee k-anonymity. Clearly, these methods reduce the risk of identification with the use of public records and also reduce the accuracy of data mining applications on the transformed data. They only concentrate on guaranteeing k-anonymity for the database and do not consider data mining processes.

Normally, after data is collected, they will be analyzed by data mining applications to enucleate some value information. Therefore, if input data is not correct, the result of data mining applications may be invaluable. With these methods, transformed data is generalized and suppressed too much. Consequently, the results, which are received after mining, may not bring some value information. Moreover, most data mining applications use association rule mining as their main technique to enucleate value information from the input data. Therefore, association rules, which are supported in the data, should be maintained. However, it is difficult to maintain all association rules because the number of association rules may be big. Moreover, only association rules, which are significant to the data mining process, may enucleate some value information. Therefore, we should only maintain the significant association rules to reduce the number of rule maintained and also to reduce the complexity of work.

In [4, 21], the authors proposed the *Migrate Member technique* to anonymize the database to achieve a k-anonymous version. The technique first groups tuples of original data into separate groups by the similarity in quasi-identifier values. Then, the groups, which have less than k tuples, will be transformed into the satisfied ones by performing some Migrate Member operations. A satisfied group will have at least k tuples in it. The database achieves a k-anonymity version if all groups must be satisfied after the processing. The authors also proposed an algorithm called M3AR (Migrate Member Maintenance Association Rules) to concretize the approach.

With M3AR, we guarantee k-anonymity for the database while still maintaining the significant association rules. However, it remains many unsatisfied groups, which the algorithm can not transform them into the satisfied ones, after processing. Therefore, the algorithm may need more time and pay the "cost" to anonymize these unsatisfied groups. The cause of this is that M3AR selects a random unsatisfied group for each process step and thus this group may not be good for this step. As a result, this group can receive more tuples than its need, thus we may have no tuples to anonymize other groups. Moreover, M3AR did not also consider reducing new significant association rules that are generated during the process. Because these new significant association rules can interfere in the input data of the data mining process, it can make the result of the data mining process less valuable.

In next sections, we will propose some solutions to solve the problems of the algorithm M3AR. We also propose a new algorithm to anonymize the location database to achieve an effective k-anonymous version. Moreover, the algorithm also reduces new significant association rules generated during the run of the algorithm.

III. DEFINITIONS AND VALUES

As discussed above, a database satisfies k-anonymity if any tuple in this database can be indistinctly matched to at least k respondents. Moreover, this approach also defined a set of attributes whose values may disclose some sensitive information. For the location database, we will consider the *QI* will include a location attribute and a time attribute. For simplification, we will only consider the location attribute in this paper. The time attribute will leave as future works. In this section, we will give some definitions and calculate the values that are used for the proposed algorithm.

A. Definitions

This section will give essential definitions that will be used in the algorithm:

Definition: A group is a set of tuples. Moreover, all tuples in a group must have the same *QI* values. A group satisfies k-anonymity if it has at least k tuples or has no tuples in it. Otherwise, we call this group as an unsatisfied group.

We will consider the following example. The *QI* attributes are: **Sex**, **ZIP code**, **Salary** and **Status**. We will have groups: Group *A* includes tuples: *1, 7, 10* and *13*. Group *B* includes tuple: *2*. Group *C* includes tuples: *3, 9*. Group *D* includes tuples: *4, 8* and so on. If we assume that *k* is equal to 4, group *A* will satisfy 4-anonymity while group *B*, *C*, *D* will be unsatisfied groups.

TABLE 1: AN EXAMPLE TABLE

No.	ID	Sex	ZIP code	Salary	Status	Data
1	u1	Male	70000	2000	Married	…
2	u2	Male	10000	1500	Single	…
3	u1	Female	48000	1000	Married	…
4	u5	Male	48000	2000	Married	…
5	u7	Female	70000	1500	Single	…
6	u8	Female	10000	1000	Single	…
7	u6	Male	70000	2000	Married	…
8	u4	Male	48000	2000	Married	…
9	u5	Female	48000	1000	Married	…
10	u3	Male	70000	2000	Married	…
11	u9	Male	25000	1500	Single	…
12	u11	Male	54000	1500	Married	…
13	u10	Male	70000	2000	Married	…

As discussed in the previous section, the algorithm will try to retain the association rules while guaranteeing k-anonymity. However, it is difficult to retain all association rules because the number of the association rules may be very big. Normally, the data mining process will consider association rules which occur frequently in the database. Therefore, the algorithm should try to retain these rules. We call these rules as significant rules. In the algorithm, two thresholds will be provided to specify whether an association rule is significant or not. We call them as *t_s* and *t_c*. An association rule is significant if its support value is greater than *t_s* and its confidence value is also greater than *t_c*. Conversely, the association rule is insignificant.

Definition: A change between two groups $a \rightarrow b$, where *a* and *b* are groups, will change all *QI* values of some tuples in *a* to the correlative values in *b*. For example, group *a* has two tuples with *QI* is *(x1, y1, t1)* and group *b* has three tuples with *QI* is *(x2, y2, t2)*, the change $a \rightarrow b$ will form group *b* which has five tuples. The additional tuples are from group *a* and their *QI* attributes are changed to *(x2, y2, t2)*.

Definition: a change $a \rightarrow b$ is total if all tuples in group *a* are transferred to group *b*. Conversely, if several of them are transferred, the change will be partial.

B. Values Calculation

With our algorithm, we will try to transform unsatisfied groups into satisfied ones. To do this, the algorithm will find the changes which will be performed to transform these unsatisfied groups to satisfied groups. Moreover, the algorithm also maintains significant association rules of the database. Thus, the algorithm should find the suitable changes in order that when performing these changes, these significant association rules will not be lost. In this section, we will calculate some values which will be used in the algorithm to find these changes.

Assume that we have a significant association rule $A \rightarrow B$ that needs to be maintained. It means that the support and confidence values of this rule are greater than thresholds. When we perform the changes, they maybe alter some values of QI attributes of tuples supporting this significant association rule. The result is that this tuple may no longer support the association rule. Clearly, if we alter too more tuples, the association rule $A \rightarrow B$ may not be significant. Therefore, for each significant association rule, we should calculate the maximal number of tuples which we can alter so that the significant association rule is still significant. Moreover, when performing the changes, an insignificant association rule may become a significant one. As discussed above, the algorithm also guarantees that no new significant rule will be generated because the new significant rules may affect the result of the data mining process. Therefore, we also calculate the maximum number of tuples which we can alter without generating new significant association rules. The algorithm will use these maximal numbers to calculate cost for each change. The cost of a change will be mentioned in next sections. From the costs, the algorithm will find the best changes that will be used to transform an unsatisfied group into a satisfied one.

In following parts, we will calculate the maximal number of tuples which we can alter so that the association rule is still significant: We have a significant association rule $A \rightarrow B$, *s* is the support value and *c* is the confident value of this rule. We have: $s >= t\_s$ and $c >= t\_c$. When we perform a change $a \rightarrow b$, some tuples in *a* will be altered. These tuples may support the association rule $A \rightarrow B$. Therefore, when they are altered, the rule may be affected. We will consider the following cases:

Case 1: *A* will be changed:

We call *n* is the number of tuples which are anonymized, *s'* is the support value and *c'* is the confident value of the rule after performing the change. The rule is significant, therefore, we must have $s' >= t\_s$ and $c' >= t\_c$:

$$c' = \frac{number(A \rightarrow B) - n}{number(A) - n} \quad (1)$$

$$s' = \frac{number(A \rightarrow B) - n}{total} \quad (2)$$

where *number*($A \rightarrow B$) is the number of tuples which have both *A* and *B*, *number(A)* is the number of tuples which only have *A*, *total* is the number of tuples in the database. Besides, we also have:

$$s = \frac{number(A \rightarrow B)}{total} \quad (3)$$

$$c = \frac{number(A \rightarrow B)}{number(A)} \quad (4)$$

The maximal number of tuples, which can be altered, is:

$$n = MIN\left(total * (s - t\_s), \left\lfloor \frac{s * total * (c - t\_c)}{c * (1 - t\_c)} \right\rfloor\right) \quad (5)$$

Case 2: *B* will be changed:

Similarly, we have:

$$c' = \frac{number(A \rightarrow B) - n}{number(A)} \quad (6)$$

$$s' = \frac{number(A \rightarrow B) - n}{total} \quad (7)$$

Moreover, we also have:

$$s = \frac{number(A \rightarrow B)}{total} \quad (8)$$

$$c = \frac{number(A \rightarrow B)}{number(A)} \quad (9)$$

The condition are *s'>=t_s* and *c'>=t_c*. Therefore, we have:

$$n = MIN\left(total \times (s - t\_s), \left\lfloor \frac{s * total * (c - t\_c)}{c} \right\rfloor\right) \quad (10)$$

Case 3: Both *A* and *B* will be changed: We notice that this case is similar to the case 1. Therefore, we have:

$$n = MIN\left(total \times (s - t\_s), \left\lfloor \frac{s * total * (c - t\_c)}{c * (1 - t\_c)} \right\rfloor\right)$$

In short, when A is changed, we have:

$$n = MIN\left(total \times (s - t\_s), \left\lfloor \frac{s * total * (c - t\_c)}{c * (1 - t\_c)} \right\rfloor\right),$$

and when B is changed, we have:

$$n = MIN\left(total \times (s - t\_s), \left\lfloor \frac{s * total * (c - t\_c)}{c} \right\rfloor\right).$$

For insignificant association rule $A \rightarrow B$, the algorithm must guarantee that this rule will not become a significant one when performing any changes. When a tuple is altered, the value of its attributes may be changed to other values. These values may be *A* or *B*. Therefore, the support value and confidence value of the rule may be affected and this rule can become a significant rule. In this part, we will calculate the maximal number of tuples which can be altered so that the rule does not become a significant one. *s* and *c* are the support value and the confident value of this rule before performing the change, *s'* and *c'* are the corresponding values after performing the change. *n* is the number of additional tuples. We will consider following cases:

Case 1: *A* will be added:

In this case, we always have:

$$c' = \frac{number(A \rightarrow B)}{number(A) + n} \quad (11)$$

$$c = \frac{number(A \rightarrow B)}{number(A)} \quad (12)$$

where *number*($A \rightarrow B$) is the number of tuples which have both *A* and *B*, *number(A)* is the number of tuples which only have *A*. Clearly, *c'* is smaller than *c*. We also have:

$$s = \frac{number(A \rightarrow B)}{total} = s' \quad (13)$$

where *total* is the number of tuples in the database. Therefore, if only value A will be added, this rule can not become a significant rule.

Case 2: *B* will be added: Similar to the case 1, the rule can not be a significant rule.

Case 3: Both *A* and *B* will be added:

Similarly, we will have:

$$c' = \frac{number(A \rightarrow B) + n}{number(A) + n} \quad (14)$$

$$s' = \frac{number(A \rightarrow B) + n}{total} \quad (15)$$

Moreover, we also have:

$$s = \frac{number(A \rightarrow B)}{total} \quad (16)$$

$$c = \frac{number(A \rightarrow B)}{number(A)} \quad (17)$$

The condition are *s'<=t_s* and *c'<=t_c*. Therefore, we have:

$$n = MIN\left(total \times (t\_s - s), \left\lfloor \frac{s * total * (t\_c - c)}{c * (1 - t\_c)} \right\rfloor \right) \quad (18)$$

In summary, when A and B are added, the maximum number of tuples which we can alter without generating new significant association rules:

$$n = MIN\left(total \times (t\_s - s), \left\lfloor \frac{s * total * (t\_c - c)}{c * (1 - t\_c)} \right\rfloor \right).$$

IV. ALGORITHM

Clearly, the objectives of the proposed algorithm are to perform the changes to transform unsatisfied groups into satisfied ones, and to maintain the significant association rules. Moreover, the algorithm should also reduce the number of new significant association rules that are created while running the algorithm. We call the maintaining significant association rules and reducing the number of new significant association rules as proposed algorithm's goals. During the anonymization, a group can be in two statuses, receiving tuples or distributing tuples. When we perform a change $a \rightarrow b$ between two groups *a* and *b*, we consider *a* is the group that distributes tuples and *b* is the group that receives tuples. Furthermore, the algorithm should guarantee that the goals will not be violated when the changes are performed.

The algorithm will perform some changes to transform each unsatisfied group into the satisfied one. Thus, for each unsatisfied group, the algorithm will choose a/some group(s), which is the other unsatisfied group or the satisfied group, to form the changes. However, the algorithm does not choose these groups randomly; it will choose the best "compatible" groups so that when performing the changes between the unsatisfied group and these "compatible" groups, they have the least effect on the algorithm's goal. To do this, the algorithm will calculate "cost" for each change. Then it will choose the changes which have the least cost. While seeking these best "compatible" groups, the algorithm should concern the following issues:

- Consider two-way for the changes between two groups. It means the algorithm will consider the changes $a \rightarrow b$ and $b \rightarrow a$ and then choose the best one when considering the changes between group *a* and *b*.
- For each unsatisfied group, the algorithm will choose the changes which have the least effect on the association rules when performing it.
- A group can receive or distribute tuples more than one time.
- A group can receive tuples from different groups.
- Prioritize the combination of two unsatisfied groups when we have some combinations that have same cost.
- For unsatisfied groups, prioritize the receipt of tuples from satisfied groups and the distribution of tuples to another unsatisfied groups.

Moreover, as discussed above, the algorithm should assign a priority degree for each unsatisfied group in order to determine which groups will be processed first. First of all, the algorithm will try to transform unsatisfied groups which have higher priority degree. Then, it will work with the lower ones. In the previous papers [4, 21], their algorithm chose the current transformed unsatisfied group randomly. Therefore, this group may receive all of tuples that are available for distribution and we will not have enough tuples for other unsatisfied groups. As a result, we may get more unsatisfied groups after finishing the algorithm.

For example, we have three unsatisfied groups: first group has 1 tuple, the second group has three tuples and the third one has four tuples. We also have k = 5 and the number of tuples, which are available for distributing, is 3. If the first one is processed first, it will receive all of these tuples and when the other groups are processed, we have no tuple for them. Therefore, we still have three unsatisfied groups after the processing. Conversely, if we process the third group first, it will receive one tuple to guarantee k-anonymity; the second one will be processed then and receives two tuples. Finally, we have two satisfied groups and one unsatisfied group. The second result is better. In this algorithm, we will try to transform many more unsatisfied groups into the satisfied ones by assigning a priority degree for each unsatisfied group. To assign the priority degree for unsatisfied groups, the algorithm will base on criteria:

Criteria:

- Prioritize unsatisfied groups in which the number of tuples is closer to k: because the algorithm will try to receive satisfied groups as many as possible, it will give priority to the unsatisfied groups which are closer to gain the satisfied ones. Clearly, unsatisfied groups, which the number of its tuples is closer to k, will be transformmed to the satisfied ones more easily.
- Prioritize unsatisfied groups which can not distribute tuples.

The algorithm will try to finish the anonymization of current unsatisfied group before working with next unsatisfied groups. An unsatisfied group can be transformed into a satisfied one if one of two following cases can be performed without affecting the goals: (i) all its tuples are distributed to other groups; (ii) it adds some tuples from other groups so that the number of its tuples is greater than k. In the second case, if a great number of tuples can be added to current unsatisfied group without affecting the goals, the group should only add enough tuples. It means that the number of group's tuples after processing should be equal to k. The remaining tuples will be left for other unsatisfied groups which are processed later.

Clearly, the greater the number of unsatisfied groups is, the more slowly the algorithm may run. Therefore, the algorithm should first reduce the number of unsatisfied groups. Moreover, the number of significant association rules is also affect the run of the algorithm because the algorithm always

considers these rules during the transformation of unsatisfied groups. In this paper, we also propose the grid based solution to apply to the location attribute of the location database. Normally, when mining the location data, data mining applications usually try to find some valuable values in an area rather than at an exact location. Therefore, the idea of this solution is that the exact location values will be anonymized into grid cells. With this solution, the algorithm will create a grid which covers the space containing the locations of the users in the database. After that, the locations of the users will be anonymized into this grid's cell. We will consider the following example: we have 11 location values which will be anonymized into a grid. The grid, which covers the space containing the locations of all users, is in the following figure (*S* is a starting point):

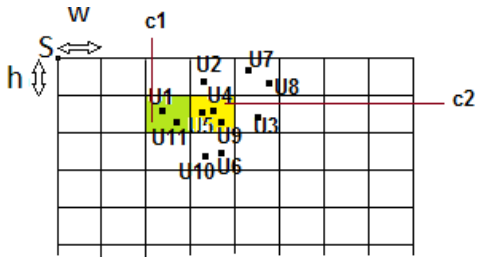

Fig. 1. An example of grid based solution.

The values *U1* and *U11* will be anonymized into cell *c1*, the values *U4, U5, U9* will be anonymized into cell *c2* and so on. Clearly, if we have two association rules *A→U1* and *A→U11*, they may become an association rule *A→c1*. Thus, the number of significant association rules, which need to maintain, can be reduced. Moreover, the number of unsatisfied groups may also be reduced because the number of tuples in each group may be increased. As a result, the algorithm will run more quickly.

The proposed algorithm can be described as the following pseudocode:

Name: *k_anoymization()*

Input: Set *R* includes the significant association rules which need to maintain, *k*, original table *T*, *QI*, the grid cell size.

Output: anonymous version table *T'*

Method:

```
1. Create a grid and anonymize all
   location values into this grid.
2. Construct a set S which contains
   satisfied groups and a set US which
   contains unsatisfied groups.
3. Sort the set US by above criteria.
4. Calculate the number of tuples which
   can be moved for each rule in R
5. a set cannotProcess=∅, it contains
   groups that can not be transformed into
   a satisfied one.
   While (US is not empty) {
6. Select an unstaisfied group proUS from
   US by its priority degree
7. US = US \ proUS
   While (proUS is unsatisfied group) {
8. Run find_best_can_group() function to
   find a best change to transform proUS.
   A candidate group can and a set of
   tuples W containing tuples, which can
   be anonymized without affecting the
   goals, will be returned by this
   function.
9. Exclude can from US or S
   if (can == null){
10.cannotProcess = cannotProcess U proUS
11.Give back all tuples, which are
   anonymized during the transformation of
   the current unsatisfied group, to their
   original groups.
12.Unmark all examined groups in S and US
13.break;
   } Else {
14.Perform the change.
15.Update the support and confidence
   values of each rule in R
16.Mark can as be examined
17.if(can is satisfied group) S = S U can
18.Else US = US U can
19.S = S U proUS
20.Unmark all examined groups in S and US
   } } }
   if (cannotProcess is not empty) {
21.final_process() }
```

The algorithm will try to transform each unsatisfied group into a satisfied one. For each unsatisfied group, the algorithm will try to finish the transformation for it before working with next unsatisfied groups. After the processing, if the algorithm can not transform this unsatisfied group into a satisfied one, all tuples, which are anonymized during the processing of this unsatisfied group, will be given back to their original groups and this unsatisfied group will be added to the set *cannotProcess*. The algorithm will try to solve this set at the final step.

During the transformation of an unsatisfied group *proUS*, the algorithm will try to find changes which will apply to this unsatisfied group to transform this group into a satisfied one. Each change will have its cost which reflects the effect of this change on the goals. The cost for each change will be calculated in the *find_best_can_group()* function. From the costs of these changes, this function will also find the best changes for current unsatisfied group. A candidate group *can* and a set of tuples *W* containing tuples, which can be anonymized without affecting the goals, will be returned by this function. The set *W* will contain tuples from *can* if we have the change *can->proUS*. Otherwise, *W* will contain tuples from *proUS*. After receiving results from the *find_best_can_group()* function, the algorithm will perform the change, which is in accord with the results, for current unsatisfied group. After performing each change, if the unsatisfied group is not still satisfied, the algorithm will try to find additional changes to transform this unsatisfied group into the satisfied one. If the algorithm can not find any additional changes to transform the group without affecting the goals, this group will be moved to the set *cannotProcess*.

Moreover, as discussed above, in the case we have too more tuples can be added to an unsatisfied group to transform it into the satisfied one, this unsatisfied group should add enough tuples to guarantee k-anonymity. The remains of tuples will be reserved for other unsatisfied groups. Therefore, when the algorithm performs a change for current unsatisfied group *proUS*, following cases will be considered:

- *proUS* receives tuples: if *(the number of tuples in W + the number of tuples in proUS)* is smaller than *k*, all tuples in *W* will be anonymized into *proUS*. Otherwise, the number of tuples in *W*, which will be anonymized into *proUS*, is *(k – number of tuples in proUS)*.
- *proUS* distributes tuples: if *can* is an unsatisfied group and *(the number of tuples in W + the number of tuples in can)* is greater than *k*, the number of tuples in *W*, which will be anonymized into *can*, is *(k – number of can)*. Otherwise, all tuples in *W* will be anonymized into *can*.

Clearly, the most important function in the algorithm is *find_best_can_group()*, which will try to find the best changes to transform current unsatisfied group into the satisfied one. In this function, we will provide 2 thresholds *t_s* and *t_c*. As discussed above, the algorithm will maintain the significant association rules which their support values are greater than *t_s* and their confident value are also greater than *t_c*. Moreover, the algorithm will not generate additional significant association rules, which their support values are greater than *t_s* and their confident value are also greater than *t_c*, during the running of it. This function will be described as the below pseudo code:

Name: *find_best_can_group()*
Input: unsatisfied group *proUS*, threshold *t_s*, threshold *t_c*
Output: a group *can* and a set *W* contains tuples that can be moved, the direction of the change (*proUS->can* or *can->proUS*)
Method:

```
1. A group can = null
   For each group temp from US U S
   (exclude proUS and examined groups) {
2. Calculate the cost for the changes
   proUS-> temp and temp-> proUS
3. Generate set W containning tuples,
   which will not affect the goals when
   anonymizing them.
   }
   If (exist the changes that do not
   violate the goals when performming
   them) {
4. Choose a best change so that: (i) when
   performing it, the goals are not
   violated and (ii) it has the lowest
   cost. The change will include a group
   temp, a set W and a direction which
   determines proUS->temp or temp->proUS
5. Assign can = temp.
   }
   Return can and W
```

With this function, it will calculate cost for each change at first step. Moreover, as mentioned above, we always consider two-way for the changes between two groups, thus, if we have two groups *proUS* and *temp*, the algorithm will consider two changes: *proUS->temp* and *temp->proUS*. The cost, which is calculated, will base on following criteria (Notice that upper criterion has higher priority):

- The number of significant association rules which will be insignificant after performing the change.
- The number of significant association which will be generated after performing this change.
- Danger degree of significant rules after performing the change: for example, a significant rules has *support=0.7* and *confidence=0.6*. Assume that after performing the change number 1, this rule will have *support=0.64* and *confidence=0.53* and after performing the change number 2, the corresponding values will be *support=0.67* and *confidence=0.59*. The change number 2 will be better because it make the rule less dangerous.
- The number of tuples in the set *W*: the algorithm prefers set *W* which has greater number of its tuples because the more the number of tuples in the set *W*, the more satisfied an unsatisfied group.

Intuitively, we will choose the change that has the lowest cost. Moreover, the function should return the set *W* containing the tuples which can be anonymized. As discussed above, if the algorithm chooses the change *proUS ->temp*, *W* will contain some tuples from *proUS*. Otherwise, it will contain tuples from *temp*.

After anonymization, there are some unsatisfied groups which the algorithm can not find the changes to transform these unsatisfied groups into satisfied ones. These groups will be added to the set *cannotProcess*. We also notice that before an unsatisfied group will be added to the set *cannotProcess*, all tuples, which are anonymized during the processing of this unsatisfied group, will be back to their original groups. It means that all groups will return the statuses which they had before transforming current unsatisfied group. In the case the set *cannotProcess* is not empty, the algorithm will run some additional steps to transform groups in this set into satisfied ones, these addition steps are in the *final_process()* function:

- At the first step, the algorithm will try to transform unsatisfied groups, which are in the set *cannotProcess*, into the better groups that are more satisfied than the original group. It also means that the number of tuples in each better group will be closer to *k* or 0. To do this step, the algorithm will choose the best changes, which will not affect the goals when performming them, to transform the unsatisfied group into a better one. The function *find_best_can_group()* can be used to find these best changes in this step.
- At the second step, the algorithm will try to transform these better unsatisfied groups into the satisfied ones. At this step, the goals may be violated.

In order to transform the better unsatisfied groups into the satisfied ones. The algorithm will find changes that have the least effect on the goals. After that, it will perform these changes to transform the better unsatisfied groups into the satisfied ones.

In contrast with the previous steps, the goals will be violated if changes, which are found in the second step, are performed. It means that some significant association rules may be no longer significant and/or new significant association rules may be generated after these changes are performed. This is "cost" which we must pay to guarantee k-anonymity for the database because with these unsatisfied groups, the algorithm can not find any changes to transform them without effect on the goals.

V. Evaluations

In this section, we show the evaluation we conducted in order to evaluate the effectiveness of our algorithms. We will verify the proposed algorithm with three other algorithms: M3AR [4], KACA [20], OKA [19] in both criteria: the percentage of lost significant association rules and the percentage of new significant association rules that are generated during the run of algorithms. Intuitively, the smaller two values, the more effective the algorithm. We call them as *p_s* and *p_n*:

$$p\_s = \frac{l\_r}{t\_r}, \tag{19}$$

where *l_r* is the number of significant association rules that are lost during the run of the algorithm and *t_r* is the total of significant association rules.

$$p\_n = \frac{n\_r}{t\_r}, \tag{20}$$

where *n_r* is the number of significant association rules that are generated during the run of the algorithm and *t_r* is the total of significant association rules.

The real database, which is used for the evaluation, will be extracted from GeoLife project [16, 17], which is collected in (Microsoft Research Asia) GeoLife project by 165 users in a period of over two years (from April 2007 to August 2009) and Adult database from the UC Irvine Machine Learning Repository [18]. This database will include 34827 records. The *QI* will include status, age, sex and location attribute. The grid cell size, which is used to anonymize the location attributes, is 500m*500m. For each value of *k*, we will execute each algorithm in five times; the achieved result is the average of five tests. The following figures show the result of the evaluation.

These results show that with our proposed algorithm, the percentage of significant association rules, which are lost during the run of the algorithm, is minimal. Similarly, the percentage of new significant association rules, which is generated during the processing, is also minimal. It also means that our algorithm will generate an effective k-anonymous version of the database. The reason of these results is that our algorithm tries to transform the unsatisfied groups with the changes that will cause least effect on the goals. Therefore, the result of data mining process may be more effective.

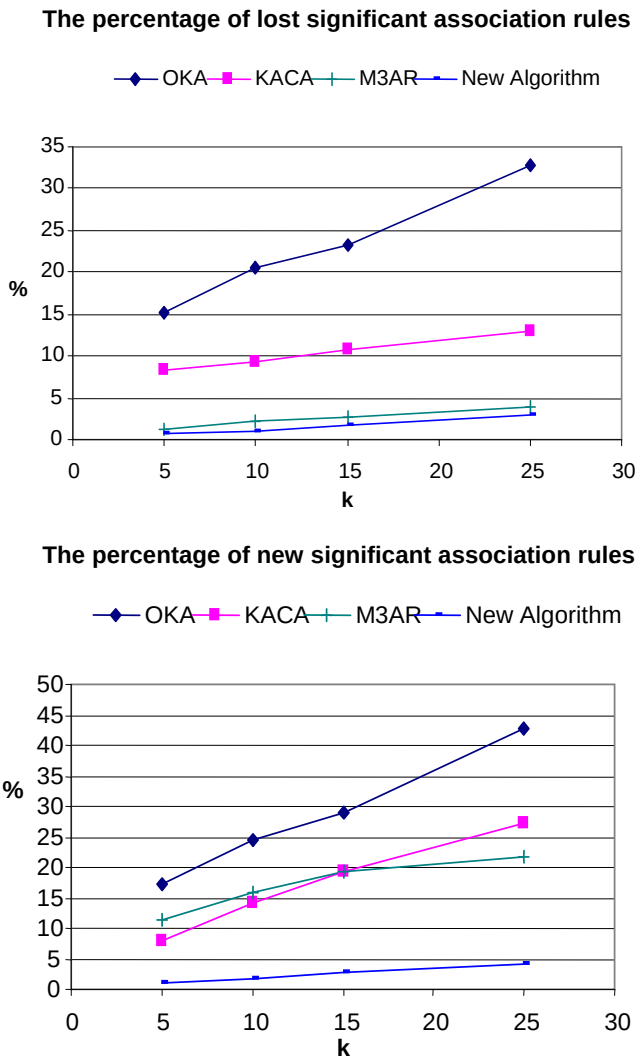

Fig. 2. The evaluation results.

VI. Conclusion and Future Works

In this paper, we proposed an algorithm that anonymizes the location database to an effective k-anonymous version. The algorithm solves some problems in the M3AR algorithm that was proposed before to guarantee k-anonymity for general databases. With the algorithm, the number of significant association rules, that are lost during the anonymization, is reduced. Thus, the results generated by the data mining process, which input data is the k-anonymous version of the database, are more valuable.

The paper also proposed the solution to reduce the significant association rules which are generated during the anonymization. Clearly, if new significant association rules are generated, they may interfere negatively in the results of the data mining process. With the newly proposed algorithm, the number of new significant association rules, which is

generated spuriously, is also reduced and hence the result of the data mining process is more effective.

In this paper, we applied the grid based solution to reduce the number of significant association rules and also reduce the number of unsatisfied groups. Thus, the algorithm is more efficient. In the future, we will focus on investigating additional solutions to improve the performance of the algorithm. On the other side, we should assign a priority degree for each unsatisfied group to determine which group will be processed first. The priority degree will be based on some criteria that are mentioned above. We can improve these criteria so that the algorithm can return a more effective k-anonymous version of the database. Moreover, the location of the user is usually accompanied with a time value. Therefore, the algorithm should also consider the time value when anonymizing the location database.

ACKNOWLEDGMENTS

This work was supported in part by D-STAR Lab (www.dstar.edu.vn). We appreciate the helpful supports from all members of D-STAR Lab (www.dstar.edu.vn) during this research.

REFERENCES

[1] Sergio, M., Claudio, B., Wang, S.X. and Sushil, J.: k-anonymity in Databases with Time Stamped Data. In: 13th International Symposium on Temporal Representation and Reasoning, pp. 177--186, IEEE Press, Budapest, Hungary (2006).

[2] Truong, T.A., Truong, Q.C. and Dang, T.K.: An Adaptive Grid-based Approach to Location Privacy Preservation. In: 2nd Asian Conference on Intelligent Information and Database Systems, pp. 133--144, Springer Verlag, Vietnam (2010)

[3] Truong, Q.C., Truong, T.A. and Dang, T.K.: The Memorizing Algorithm: Protecting User Privacy in Location-Based Services using Historical Services Information. In: International Journal of Mobile Computing and Multimedia Comm., 2(4), pp. 65--86, IGI-Global (2010)

[4] Dang, T.K, Kueng, J., Huynh, V.Q.P: Protecting User Privacy while Discovering and Maintaining Association Rules. In: 4th IFIP International Conference on New Technologies, Mobility and Security, IEEE Computer Society, Paris, France (2011)

[5] Ciriani, V., De Capitani di Vimercati, S., Foresti, S. and Samarati, P.: k-Anonymous Data Mining: A Survey. In: Michael, G., Sushil, J. (eds.), Handbook of Database Security– Applications and Trends, pp. 105--136, Springer Science, LLC (2008)

[6] Sweeney, L.: Achieving k-anonymity Privacy Protection using Generalization and Suppression. In: International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, 10(5), pp. 571--588, World Scientific (2002)

[7] Samarati, P. and Sweeney, L.: Protecting Privacy When Disclosing Information: k-anonymity and its Enforcement through Generalization and Suppression. Technical Report SRI-CSL-98-04, Computer Science Laboratory, SRI International (1998)

[8] Gedik, B. and Ling, L.: Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. In: IEEE Trans. on Mobile Computing, 7(1), pp. 1--18 (2008)

[9] Gruteser, M. and Grunwald, D.: Anonymous Usage of Location-Based Services through Spatial and Temporal Cloaking. In: ACM International Conference Mobile Systems, Applications, and Services, pp 31--42, ACM New York, USA (2003)

[10] Bettini, C., Mascetti, S. and Wang, X.S.: Privacy Protection through Anonymity in Location-based Services. In: Michael, G., Sushil, J. (eds.), Handbook of Database Security – Applications and Trends, pp. 509--530, Springer Science, LLC (2008)

[11] To, Q.C., Dang, T.K., Kueng, J.: A Hilbert-based Framework for Preserving Privacy in Location-based Services. Intl. Journal of Intelligent Information and Database Systems (IJIIDS), Inderscience Publisher, ISSN 1751-5858 (2012) (*to appear*)

[12] Cuellar, J.R.: Location Information Privacy. In: B. Srikaya (Ed.), Geographic Location in the Internet, pp. 179--208, Kluwer Academic Publishers (2002)

[13] Gidófalvi, G., Huang, X. and Pedersen, T.B: Privacy-Preserving Data Mining on Moving Object Trajectories. In: 8th International Conference on Mobile Data Management, pp. 60--68, Mannheim, Germany (2007)

[14] Bettini, C., Wang, X. and Jajodia, S.: Protecting Privacy against Location-based Personal Identification. In: 2nd VLDB Workshop on Secure Data Management, pp. 185--199, Trondheim, Norway (2005)

[15] Ardagna, C.A., Cremonini, M., Vimercati, S.D.C. and Samarati, P.: Privacy-enhanced Location-based Access Control. In: Michael, G., Sushil, J. (eds.), Handbook of Database Security – Applications and Trends, pp. 531--552, Springer Science, LLC (2008)

[16] Zheng, Y., Li, Q., Chen, Y. and Xie, X.: Understanding Mobility Based on GPS Data. In: ACM conference on Ubiquitous Computing, pp. 312--321, ACM Press, Seoul, Korea (2008)

[17] Zheng, Y., Zhang, L., Xie, X. and Ma, W.Y.: Mining Interesting Locations and Travel Sequences from GPS Trajectories. In: International conference on World Wild Web, pp. 791--800, ACM Press, Madrid, Spain (2009)

[18] Newman, D.J., Hettich, S., Blake, C.L. and Merz, C.J.: UCI Repository of Machine Learning Databases, available at *www.ics.uci.edu/mlearn/MLRepository.html*, University of California, Irvine (1998)

[19] Jun, L.L. and Meng, C.W.: An Efficient Clustering Method for k-anonymization. In: the 2008 International Workshop on Privacy and Anonymity in Information Society, pp. 46--50, ACM New York, Nantes, France (2008)

[20] Li, J., Wong, R.C.W., Fu, A.W.C and Pei, J.: Achieving k-Anonymity by Clustering in Attribute Hierarchical Structures. In: Tjoa, A.M., Trujillo, J. (eds.) Data Warehousing and Knowledge Discovery, LNCS 4081, pp. 405--416, Springer Verlag, Heidelberg (2006)

[21] Huynh, V.Q.P. and Dang, T.K.: eM^2: An Efficient Member Migration Algorithm for Ensuring k-Anonymity and Mitigating Information Loss. In: 7th VLDB Workshop on Secure Data Management, pp. 26--40, Springer Verlag, Singapore (2010)

[22] To, Q.C., Dang, T.K., Kueng, J.: OST-tree: An Access Method for Obfuscating Spatio-Temporal Data in Location Based Services. In: 4th IFIP International Conference on New Technologies, Mobility and Security, IEEE Computer Society, Paris, France (2011)

[23] To, Q.C., Dang, T.K., Kueng, J.: B^{ob}-Tree: An Efficient B+-Tree Based Index Structure for Geographic-aware Obfuscation. In: 3rd Asian Conference on Intelligent Information and Database Systems, LNAI 6591, pp. 109--118, Springer Verlag, Korea (2011)

[24] To, Q.C., Dang, T.K., Kueng, J.: A Hilbert-based Framework for Preserving Privacy in Location-based Services. In: International Journal of Intelligent Information and Database Systems, Inderscience Publisher (2013, to appear)

[25] Le, T.B.T, Dang, T.K.: Semantic Bob-Tree: A New Obfuscation Technique for Location Privacy Protection. In: 10th International Conference on Advances in Mobile Computing & Multimedia, ACM, Bali, Indonesia (2012)