On Guaranteeing k-Anonymity in Location Databases

Anh Tuan Truong[1], Tran Khanh Dang[1,*] and Josef Küng[2]

[1] Faculty of Computer Science and Engineering, HCMC University of Technology, VNUHCM, Ho Chi Minh City, Vietnam
{anhtt,khanh}@cse.hcmut.edu.vn
[2] Institute for Application Oriented Knowledge Processing, Johannes Kepler University of Linz, Austria, Europe
josef.kueng@faw.jku.at

Abstract. The development of location-based services and mobile devices has lead to an increase in the location data. Through the data mining process, some valuable information can be discovered from location data. However, the attackers may also extract some private (sensitive) information of the user and this can make threats against the user location privacy. Therefore, location privacy protection becomes a key factor to the success in privacy preserving in location-based services. In this paper, we propose a new approach as well as an algorithm to guarantee k-anonymity in a location database. The algorithm will maintain the association rules which have significance for the data mining process. Moreover, the algorithm also considers excluding new significant association rules created during the run of the algorithm.

Keywords: k-anonymity, location databases, data mining, privacy protection.

1 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. In this paper, we will focus on protecting the user's location at time when the location data is stored in the database for data mining purposes. This paper will improve the approach which was proposed in [2, 9] 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 [2]. The approach also considers the result of data mining process by maintaining association rules that are significant to the data mining process.

[*] Part of this research had been done as the author was at FAW institute, JKU, Linz, Austria as a visiting researcher.

2 k-Anonymity Techniques, M3AR Algorithm and Problems

In [2], the authors proposed the *Migrate Member technique* (and a variant in [9]) 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, they 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.

3 Definitions and Values Calculation for Proposed Algorithm

As in [3], *Quasi-Identifiers* (QI) is the set of attributes whose values may be used, possibly together with external information, to re-identify the user's data. For the location database, we will consider the *QI* (refer [3] for details) 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.

3.1 Definitions

This section will give some definitions which 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.

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 (*t_s* and *t_c*) will be provided to specify whether an association rule is significant or not. 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→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→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).*

3.2 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. From [2], 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→B*, *s* is the support value and *c* is the confident value of this rule, *total* is the number of tuples in the database. We have two cases:

- A is changed, we have:

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

- B is changed, we have:

$$n = MIN\left(total * (s - t\_s), \left\lfloor \frac{s * total * (c - t\_c)}{c} \right\rfloor\right) \tag{2}$$

We also have the case: both *A* and *B* will be changed. However, we notice that this case is similar to the case 1: A is changed (see [2] for details). 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 add to a rule without generating new significant association rules. We will consider following cases: *A* will be added; *B* will be added and both *A* and *B* will be added. Only the last case may be make an insignificant rule become a significant one. Therefore, we have:

$$n = MIN\left(total * (t\_s - s), \left\lfloor \frac{s * total * (t\_c - c)}{c * (1 - t\_c)} \right\rfloor\right) \tag{3}$$

Where *n* is the maximal number of tuples, which can be added. *total* is the number of tuples in the database. The algorithm will use these maximal numbers to calculate cost for each change. The cost of a change will be mentioned in next sections.

4 Proposed 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.

The algorithm should guarantee that the goals will not be violated when the changes are performed. 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 goals. 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: (i) Consider two-way for the changes between two groups; (ii) Choose the changes which have the least effect on the goal; (iii) A group can receive or distribute tuples more than one time; (iv) A group can receive tuples from different groups; (v) Prioritize the combination of two unsatisfied groups when we have some combinations that have same cost; (vi) 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. In the previous papers [2], 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. In the 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:

- Prioritize unsatisfied groups in which the number of tuples is closer to k: 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.

In this paper, we also apply the grid based solution [1] to the location attribute of the location database to reduce the number of maintained association rules. As a result, the algorithm will run more quickly. 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.

The algorithm can be described as following pseudo code:

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'*

```
1.   Create a grid and anonymize all location values into this grid.
2.   Construct a set S(satisfied groups), a set US(unsatisfied groups
3.   Sort the set US by above criteria.
4.   Calculate the values as in section 3.2 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 proUS from US by the priority degree
7.      US = US \ proUS
        While (proUS is still an 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
10.        if (can == null){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 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()}
```

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*. The algorithm will try to solve this set at the final step. Clearly, the most important function is *find_best_can_group()*, which will try to find the best changes to transform current unsatisfied group into the satisfied one. As discussed above, the algorithm will try to maintain the significant association rules. 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.

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*).

```
1.   A group can = null
     For each temp from US U S (exclude proUS and examined groups) {
2.      Calculate the cost for changes: proUS-> temp and temp-> proUS
3.      Generate set W}
     If (exist the changes that do not violate the goals) {
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
```

This function will calculate cost for each change at the first step. Intuitively, we will choose the change that has the lowest cost. The cost calculated will be based on the following criteria: (i) The number of significant association rules which will be insignificant after performing the change; (ii) The number of significant association which will be generated after performing this change; (iii) The danger degree of significant rules after performing the change: for example, a significant rule 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 *0.67* and *0.59*. The change number 2 will be better because it make the rule less dangerous; (iv) 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.

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. 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. Different from the previous steps, the goals will be violated if these changes 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.

5 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 [2], KACA [7], OKA [6] 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*:

$$p\_s = \frac{l\_r}{t\_r} \tag{4}$$

$$p\_n = \frac{n\_r}{t\_r} \tag{5}$$

Where *l_r* is the number of significant association rules that are lost during the run of the algorithm, *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 [4], 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 [5]. 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.

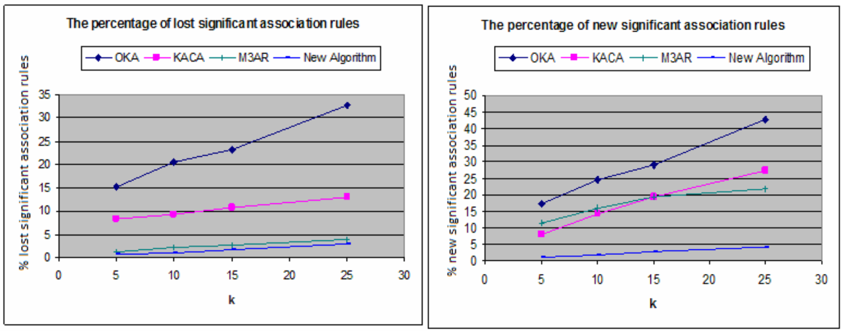

Fig. 1. The evaluation results

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.

6 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. Moreover, the number of significant association rules, which are generated during the anonymization, is also reduced. Thus, the results generated by the data mining process, which input data is the k-anonymous version of the database, are more effective and more valuable. We also 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 effective.

In the future, we will focus on investigating additional solutions to improve the performance of the algorithm. On the other side, we should improve the criteria which are applied to assign a priority degree for each unsatisfied group 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 database.

References

1. Truong, Q.C., Truong, T.A., Dang, T.K.: Memorizing Algorithm: Protecting User Privacy using Historical Information of Location-Based Services. IJMCMC 2(4), 65–86 (2010)
2. Dang, T.K., Küng, 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, France, pp. 109–118 (2011)
3. Sweeney, L.: k-Anonymity: A Model for Protecting Privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5), 557–570 (2002)
4. Zheng, Y., Li, Q., Chen, Y., Xie, X.: Understanding Mobility Based on GPS Data. In: ACM Conference on Ubiquitous Computing, Korea, pp. 312–321 (2008)
5. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases (1998), `http://www.ics.uci.edu/mlearn/MLRepository.html`
6. Jun, L.L., Meng, C.W.: An Efficient Clustering Method for k-anonymization. In: Int. Workshop on Privacy and Anonymity in Information Society, France, pp. 46–50 (2008)
7. Li, J., Wong, R.C.W., Fu, A.W.C., Pei, J.: Achieving k-Anonymity by Clustering in Attribute Hierarchical Structures. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 405–416. Springer, Heidelberg (2006)
8. To, Q.C., Dang, T.K., Küng, J.: B^{ob}-Tree: An Efficient B+-Tree Based Index Structure for Geographic-aware Obfuscation. In: Nguyen, N.T., Kim, C.-G., Janiak, A. (eds.) ACIIDS 2011, Part I. LNCS (LNAI), vol. 6591, pp. 109–118. Springer, Heidelberg (2011)
9. Van Quoc, P.H., Dang, T.K.: eM^2: An Efficient Member Migration Algorithm for Ensuring k-Anonymity and Mitigating Information Loss. In: Jonker, W., Petković, M. (eds.) SDM 2010. LNCS, vol. 6358, pp. 26–40. Springer, Heidelberg (2010)