

支持多约束的 K -匿名化方法*

杨晓春⁺, 刘向宇, 王 斌, 于 戈

(东北大学 信息科学与工程学院, 辽宁 沈阳 110004)

K -Anonymization Approaches for Supporting Multiple Constraints

YANG Xiao-Chun⁺, LIU Xiang-Yu, WANG Bin, YU Ge

(School of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

+ Corresponding author: Phn: +86-24-83687776, E-mail: yangxc@mail.neu.edu.cn

Yang XC, Liu XY, Wang B, Yu G. K -Anonymization approaches for supporting multiple constraints. *Journal of Software*, 2006,17(5):1222-1231. <http://www.jos.org.cn/1000-9825/17/1222.htm>

Abstract: K -Anonymization is an important approach to protect data privacy in data publishing scenario. Existing approaches mainly consider data processing with single constraint. There exist multiple constraints cases in the real applications, which makes the K -anonymization more complex. Simply applying the approaches with single constraint to the problem of multiple constraints may cause high information loss and low efficiency. Based on the idea of Classfly, a family of multiple constraints supported K -anonymization approaches named Classfly⁺ are proposed according to the features of multiple constraints. Three K -anonymization approaches are proposed, which are naïve approach, complete IndepCSet, and partial IndepCSet Classfly⁺ approaches. Experimental results show that Classfly⁺ can decrease the information loss and improve efficiency of k -anonymization.

Key words: k -anonymization; data privacy; generalization; multiple constraints; information loss

摘 要: K -匿名化(K -anonymization)是数据发布环境下保护数据隐私的一种重要方法。目前, K -匿名化方法主要针对单一约束条件进行处理,而实际应用中涉及到大量的多约束条件,使 K -匿名化问题更加复杂。如果简单地将单一约束 K -匿名化方法应用到多约束情况,会造成大量的信息损失及过低的处理效率。根据多约束之间的关系,通过继承 Classfly 算法的元组概括过滤思想,提出多约束 K -匿名化方法 Classfly⁺及相应的 3 种算法,包括朴素算法、完全 IndepCSet 算法和部分 IndepCSet 的 Classfly⁺算法。实验结果显示,Classfly⁺能够很好地降低多约束 K -匿名化的信息损失,改善匿名化处理的效率。

关键词: K -匿名化;数据隐私;概括;多约束;信息损失

中图法分类号: TP309 文献标识码: A

* 本文为 2005 年中国计算机大会推荐优秀论文。Supported by the National Natural Science Foundation of China under Grant Nos.60503036, 60573090 (国家自然科学基金); the University Key Teacher Award Program for Outstanding Young Teachers in High Education Institute of the Ministry of Education of China (教育部高等学校优秀青年教师教学科研奖励计划基金); the Natural Science Foundation for Doctoral Career of Liaoning Province of China under Grant No.20041016 (辽宁省博士科研启动项目); the National Research Foundation for the Doctoral Program of the Ministry of Education under Grant Nos.20030145029 (教育部博士点基金)

链接攻击(linking attack)^[1]是数据发布中非法获取隐私数据的常见方法,用户通过对发布的数据和其他渠道获得的数据进行连接处理,推演出隐私数据,从而造成隐私泄露.利用推演来标识个体信息的一组属性称作准标识符(quasi identifier)^[1],例如属性组 Race, Birth, Gender, ZIP 就是一个准标识符.如果发布的多个个体数据具有相同的准标识符,则可以防止链接攻击导致的隐私泄露.即对于一个发表的一条记录 r ,至少有 $K-1$ 条记录与 r 在准标识符上的投影值相等,称这样的元组符合 K -匿名约束.如果表中任何记录都符合 K -匿名约束,则称该表符合 K -匿名约束.产生符合 K -匿名约束的数据的处理过程称为 K -匿名化. K -匿名化会导致信息损失,不同的 K -匿名化方法造成的信息损失不同.信息损失越大, K -匿名化后数据的实用性越小.

目前, K -匿名化方法普遍针对单一约束对发布数据进行 K -匿名化处理,实际应用中存在大量多匿名约束的情况.如果一个表符合每个约束,则认为该表是符合 K -匿名约束的表.如果简单地将单一约束 K -匿名化方法应用到多约束情况,则会造成大量的信息损失及过低的处理效率,需要考虑多约束条件的特点进行多约束 K -匿名化.

本文在以前工作的基础上,对单一约束 K -匿名化方法 Classfly^[2]进行扩展,根据多约束和 Classfly 算法的特点,提出多约束 K -匿名化方法——Classfly⁺.Classfly⁺具有如下特点:(1) 继承 Classfly 的元组概括过滤思想,针对多约束特点提出最大匿名元组子集概括过滤机制,使得符合多约束的元组不参与以后的概括操作,减少信息损失,保证 K -匿名化的精度;(2) 根据多约束特点,提出独立约束子集等相关概念,给出一种约束集划分策略,将基于约束集 K -匿名化数据表问题转化为对划分的独立约束子集进行 K -匿名化处理的问题.根据不同元组概括过滤机制,提出 3 种 Classfly⁺算法:朴素 Classfly⁺算法、完全 IndepCSet Classfly⁺算法和部分 IndepCSet Classfly⁺算法.通过大量实验,对不同 K -匿名化方法的精度和执行时间进行了对比分析.

本文第 1 节介绍相关工作.第 2 节定义多约束 K -匿名化.第 3 节具体描述 3 种 Classfly⁺算法.第 4 节分析和评价实验结果.第 5 节总结全文.

1 相关工作

Sweeney 提出 K -匿名模型用于阻止链接攻击导致的隐私泄露^[1].由于 K -匿名化数据会造成信息损失,如何在数据满足 K -匿名约束的同时使数据精度最高,成为研究的热点问题.研究证明,获得最高精度的 K -匿名化数据是 NP 难问题^[3,4].文献[5]采用不完全随机搜索方法,提出 K -匿名中的数据挖掘分类问题,并给出一种基因算法以解决此问题;文献[6]给出视图中的 K -匿名验证问题;文献[7]采用启发式自底向上的概括方法,这种方法信息损失大,降低了数据的精度,并且执行效率不高,不能完全防止隐私泄露;文献[8]提出一种高效率全域 K -匿名化方法,提高了单一约束全域 K -匿名化的效率,减少了执行时间,但没有针对信息损失给出有效的解决方案.

文献[9]采用自上而下的特化方法,提出面向分类的 K -匿名化策略,在发布数据满足 K -匿名的情况下,保证分类结果的高准确率,并且自上而下特化能简单地解决多约束 K -匿名化问题.但是,该方法并未注意 K -匿名化造成的信息损失.具有代表性的 Datafly^[7]算法在对表数据进行 K -匿名化处理时,主要以属性为单位对所有元组进行 K -匿名化处理.其缺点是,尽管某些元组已经满足了 K -匿名约束要求,但还要继续参与 K -匿名化处理,从而造成了过量的信息损失.针对多约束,Datafly 基于概括数据的全域特点,只将单一约束条件下的 K -匿名化算法进行了简单修改和移植.文献[2]提出了基于特征类的 K -匿名化方法 Classfly,该方法采用单一约束条件下的元组概括过滤策略实现对发布数据的 K -匿名化,但是该方法并未给出多约束 K -匿名化的解决策略.本文继承了 Classfly 的单一约束条件下元组概括过滤思想,提出多约束 K -匿名化方法 Classfly⁺,给出几种多约束的最大匿名元组子集概括过滤策略及算法.通过实验,对每种 Classfly⁺方法的信息损失程度和执行时间进行了测试与分析.

2 多约束 K -匿名化问题

本文采用文献[10]中定义的概括、域概括层次和值概括层次对发布数据进行匿名化处理,从而使发布数据符合 K -匿名要求.本文中,匿名约束 C 以 (QI, K) 的形式来表示,其中: $QI = \{Attr_m, \dots, Attr_n\}$, QI 为约束 C 的准标识符,由一组属性 $Attr_i (m \leq i \leq n)$ 构成; K 表示 C 的匿名度,即有 K 条记录在 QI 上的取值相等.

定义 1(约束集). 一个约束集被表示为 $CSet = \{C_1, \dots, C_N\}$, 其中 $C_i = \langle QI_i, K_i \rangle, 1 \leq i \leq N, N$ 为约束集 $CSet$ 中的约束数目, 记作 $|CSet|$. 如果元组集 T 符合 $CSet$ 中的每个约束, 则称元组集 T 符合 $CSet$.

定义 2(约束相交). 对于约束 $C_i = \langle QI_i, K_i \rangle$ 和 $C_j = \langle QI_j, K_j \rangle$, 如果 $QI_i \cap QI_j$ 非空, 则称约束 C_i 和 C_j 相交.

定义 3(约束包含). 对于约束 $C_i = \langle QI_i, K_i \rangle$ 和 $C_j = \langle QI_j, K_j \rangle$, 称 C_j 包含 C_i , 记作 $C_i \preceq C_j$, 当且仅当 $QI_i \subseteq QI_j$ 并且 $K_i \leq K_j$ 时, 称 C_i 是 C_j 的子约束, C_j 是 C_i 的父约束.

定理 1. 当 $C_i \preceq C_j$ 时, 如果元组集 T 符合约束 C_j , 则元组集 T 同样符合 C_i .

证明: 令 $count_T(t[QI])$ 表示在元组集 T 中, 与元组 t 在准标识符 QI 上具有相同投影值的元组数目. 由 $C_i \preceq C_j$ 知, $QI_i \subseteq QI_j$, 如果 T 符合约束 C_j , 则 $\forall t \in T$ 有 $K_j \leq count_T(t[QI_j])$, 即存在 $count_T(t[QI_j]) - 1$ 个元组和 t 在 QI_j 上的投影相等, 而 $QI_i \subseteq QI_j$, 因此, 至少有 $count_T(t[QI_j]) - 1$ 个元组和 t 在 QI_i 上投影相等, 即 $count_T(t[QI_i]) \leq count_T(t[QI_j])$. 因此, $K_j \leq count_T(t[QI_j])$. 另外, 由 $C_i \preceq C_j$ 可知 $K_i \leq K_j$, 因此, $K_i \leq count_T(t[QI_i])$. 所以, T 符合约束 C_i .

定义 4(独立约束子集). 已知约束集 $CSet$ 和 $SubCSet$ 满足 $SubCSet \subseteq CSet$, 如果下面两个条件之一得到满足, 则称 $SubCSet$ 是 $CSet$ 的独立约束子集:

- (1) $SubCSet$ 只包含一个约束 C , 而 C 与 $CSet - \{C\}$ (“-”表示集合的差操作) 中的任何一个约束不相交;
- (2) $SubCSet$ 包含多个约束, $\forall C \in SubCSet, SubCSet$ 中至少有 1 个约束和 C 相交, 并且 $CSet - SubCSet$ 中的任何一个约束和 C 不相交.

定理 2(约束集划分唯一性). 将约束集 $CSet$ 划分成 M 个独立约束子集 $IndepCSet_i (i=1, \dots, M)$, 则 $CSet$ 中的任意约束 C 唯一属于某个独立约束子集, 并且划分是唯一的, 即 $CSet$ 如果还可划分成 M' 个独立约束子集 $OtherIndepCSet_j (j=1, \dots, M')$, 则满足 $\{IndepCSet_1, \dots, IndepCSet_M\} = \{OtherIndepCSet_1, \dots, OtherIndepCSet_{M'}\}$.

定义 5(联合约束). 已知约束集 $CSet = \{C_1, \dots, C_N\}$, 其中 $C_i = \langle QI_i, K_i \rangle, 1 \leq i \leq N$, 则 $CSet$ 的联合约束记作 $\left\langle \bigcup_{i=1}^N QI_i, K = \max\{K_1, \dots, K_N\} \right\rangle$.

定理 3. 已知约束集 $CSet = \{C_1, \dots, C_N\}$, 其中 $C_i = \langle QI_i, K_i \rangle, 1 \leq i \leq N$. 如果元组集 T 符合 $CSet$ 的联合约束, 则 T 一定符合 $CSet$; 反之不成立.

证明: 由于 T 符合联合约束 $UnionC = \left\langle \bigcup_{i=1}^N QI_i, K = \max\{K_1, \dots, K_N\} \right\rangle$, 即 $\forall t \in T$ 有 $K \leq count_T \left(t \left[\bigcup_{i=1}^N QI_i \right] \right)$, 因此 $K \leq count_T(t[QI_i]) (1 \leq i \leq N)$, 又由 $K = \max\{K_1, \dots, K_N\}$ 可知, $K_i \leq count_T(t[QI_i]) (1 \leq i \leq N)$, 所以, T 符合约束集 $CSet$.

定义 6(匿名元组子集). 已知元组集 T 和约束集 $CSet$, T' 是 T 的一个元组子集, 如果 T' 符合 $CSet$, 则称 T' 是 T 在约束集 $CSet$ 上的一个匿名元组子集.

定义 7(最大匿名元组子集). 已知 AT 是 T 在约束集 $CSet$ 上的一个匿名元组子集, 如果 $\forall T' \subseteq T - AT, AT \cup T'$ 都不符合 $CSet$, 则称 AT 为 T 在约束集 $CSet$ 上的最大匿名元组子集.

定理 4. 元组集 T 在约束集 $CSet$ 上的最大匿名元组子集是唯一的.

证明: 已知元组集 T 和约束集 $CSet$, 假设存在两个元组子集 T_1 和 $T_2 (T_1 \neq T_2)$ 为 T 的最大匿名元组子集. 由于 T_1 和 T_2 均符合约束集 $CSet$, 因此 $T_1 \cup T_2$ 符合约束集 $CSet$. 由于 $T_1 \neq T_2$, 令 $T' = T_1 \cup T_2 - T_1$, 则 $T' \neq \emptyset$, 而 $T' \cap T_1 = \emptyset$ 并且 $T' \cup T_1 = T_1 \cup T_2$ 符合 $CSet$, 因此, T_1 不符合最大匿名元组子集的定义. 同理可证 T_2 不符合最大匿名元组子集的定义. 因此, 元组集 T 基于约束集 $CSet$ 的最大匿名元组子集是唯一的.

当采用约束集 $CSet$ 来 K -匿名化表 T 时, 已知 $CSet$ 唯一划分为若干独立约束子集 $IndepCSet_i (i=1, \dots, M)$, 由于独立约束子集之间无公共属性, 因此独立约束子集 $IndepCSet_i (i=1, \dots, M)$ 将表 T 划分为 M 个不相交的子表 $T_i (i=1, \dots, M)$, 其中每个子表 T_i 包含的属性是独立约束子集 $IndepCSet_i$ 所包含的属性. 由于子表之间不相交, 所以如果子表 $T_i (i=1, \dots, M)$ 符合对应独立约束子集 $IndepCSet_i$, 则表 T 一定符合约束集 $CSet$.

3 Classfly⁺:支持多约束的 K -匿名化方法

3.1 基于独立约束子集的 K -匿名化

当存在多约束集 $CSet$ 时,对表 T 进行 K -匿名化处理可以被转化为对一些子表 T_i 的 K -匿名化处理.这些子表根据独立约束子集 $IndepCSet_i$ 来划分,从而使表 T 符合约束集 $CSet$.当独立约束子集只包含单个约束时,可以基于该约束采用 Classfly 方法来处理相应子表.对于包含多个约束的独立约束子集,单一约束 K -匿名化方法 Classfly 的元组概括过滤策略不再适用.本文提出几种支持多约束 K -匿名化处理的策略.如图 1 所示,需要使用支持多约束的 K -匿名化方法使子表 T_1 符合 $IndepCSet_1$,子表 T_2 符合 $IndepCSet_2$,而对于子表 T_3 ,只需使用 Classfly 算法使 T_3 符合单约束 C_5 即可.

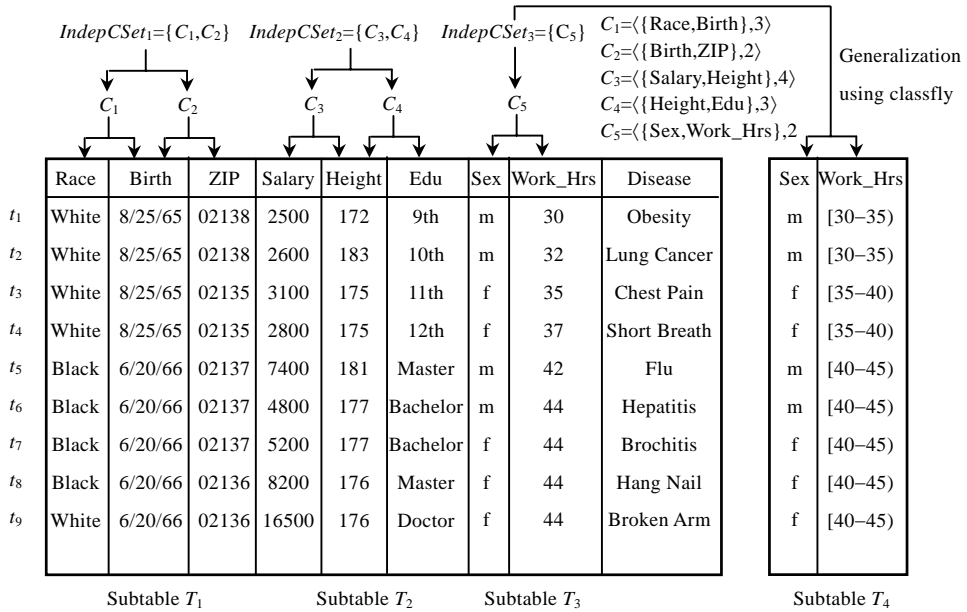


Fig.1 A table with multiple constraints

图 1 具有多约束的数据表

类似于 Classfly,基于独立约束子集 $IndepCSet$,采用概括方法对子表 T 进行 K -匿名化处理时,子表 T 中符合独立约束子集 $IndepCSet$ 的元组可以被过滤,不参与之后的概括操作.不同的过滤策略导致不同的处理效率以及不同的信息损失程度.因此,本文着重讨论不同的元组概括过滤策略.

3.2 Classfly⁺核心算法

本节对 Classfly⁺算法的核心部分进行描述.由于多约束下的元组概括过滤策略在不同的 Classfly⁺方法中互不相同,后面对各种 Classfly⁺方法的多约束元组概括过滤方法进行详细描述.

算法 1(Classfly⁺核心算法).

输入:发布数据表 PT ,约束集 $CSet = \{C_1, C_2, \dots, C_N\}$;

输出:符合 $CSet$ 的数据表 GPT .

步骤:

- (1) 读入表 PT 以及约束集 $CSet$;
- (2) 预处理约束集 $CSet$,将 $CSet$ 划分为 M 个独立约束子集 $IndepCSet_i(i=1,2,\dots,M)$;
- (3) 将独立约束子集中的约束按照匿名度低优先原则排序,基于每个独立约束子集将表 PT 划分成 M 个不相交子表 $T_i(i=1,2,\dots,M)$;

- (4) 对于包含一个约束的独立约束子集及其对应子表,基于该独立约束子集对子表直接采用 Classfly 算法进行匿名化处理;
- (5) 对于包含多个约束的独立约束子集及其对应子表,基于该独立约束子集对子表采用多约束元组概括过滤策略进行匿名化处理,使得每个子表符合该独立约束子集;
- (6) 输出表 GPT ;

3.3 朴素 Classfly⁺算法

基于独立约束子集 $IndepCSet$ 对子表 T 进行 K -匿名化处理时,根据定理 3 可知,符合 $IndepCSet$ 联合约束的元组集必定符合 $IndepCSet$,因此,概括过程中符合 $IndepCSet$ 的联合约束的元组集可以不参与之后的概括处理.称该方法为朴素 Classfly⁺算法(naive Classfly⁺).

算法 2(Naive Classfly⁺算法中多约束元组概括过滤策略).

输入:独立约束子集 $IndepCSet=\{C_1,\dots,C_H\}$ 及其对应子表 T ;

输出:符合 $IndepCSet$ 的子表 GT .

步骤:

- (1) 初始子表 GT 的所有元组标记为 $to_be_suppressed$; // $to_be_suppressed$ 表示要被隐匿(suppress)
- (2) 当所有标记 $to_be_suppressed$ 的元组数目之和 $\geq\max\{K_1,\dots,K_H\}$ 时循环执行以下操作:
 - (a) 遍历标记 $to_be_suppressed$ 的元组,如果某个元组 t 符合 $IndepCSet$ 的联合约束,则将所有和 t 在联合约束准标识符上数值相等的元组标记为 $candidate$; // $candidate$ 表示候选
 - (b) 在标记为 $to_be_suppressed$ 的元组中选择不同数值最多的属性进行概括;
- (3) 隐匿标记为 $to_be_suppressed$ 的元组.

如图 1 所示,已知表 T 和约束集 $CSet=\{C_1,C_2,C_3,C_4,C_5\}$,根据朴素 Classfly⁺算法,基于约束 $\langle\{Race,Birth,ZIP\},3\rangle$ 采用 Classfly 算法对子表 T_1 进行 K -匿名化处理,结果如图 2 中的表 T_a 所示.可以看出,子表 T_a 符合 $IndepCSet_1$ 的联合约束 $\langle\{Race,Birth,ZIP\},3\rangle$,同时符合独立约束子集 $IndepCSet_1=\{C_1,C_2\}$.

	Table T_a		Table T_b																																																																																
$C_1=\langle\{Race,Birth\},3\rangle$ $C_2=\langle\{Birth,ZIP\},2\rangle$	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th></th> <th>Race</th> <th>Birth</th> <th>ZIP</th> </tr> </thead> <tbody> <tr><td>t_1</td><td>White</td><td>8/25/65</td><td>0213*</td></tr> <tr><td>t_2</td><td>White</td><td>8/25/65</td><td>0213*</td></tr> <tr><td>t_3</td><td>White</td><td>8/25/65</td><td>0213*</td></tr> <tr><td>t_4</td><td>White</td><td>8/25/65</td><td>0213*</td></tr> <tr><td>t_5</td><td>Black</td><td>6/20/66</td><td>02137</td></tr> <tr><td>t_6</td><td>Black</td><td>6/20/66</td><td>02137</td></tr> <tr><td>t_7</td><td>Black</td><td>6/20/66</td><td>02137</td></tr> <tr><td>t_8</td><td>*****</td><td>**/**/**</td><td>**/**/**</td></tr> <tr><td>t_9</td><td>*****</td><td>**/**/**</td><td>**/**/**</td></tr> </tbody> </table>		Race	Birth	ZIP	t_1	White	8/25/65	0213*	t_2	White	8/25/65	0213*	t_3	White	8/25/65	0213*	t_4	White	8/25/65	0213*	t_5	Black	6/20/66	02137	t_6	Black	6/20/66	02137	t_7	Black	6/20/66	02137	t_8	*****	**/**/**	**/**/**	t_9	*****	**/**/**	**/**/**		<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th></th> <th>Race</th> <th>Birth</th> <th>ZIP</th> </tr> </thead> <tbody> <tr><td>t_1</td><td>White</td><td>8/25/65</td><td>02138</td></tr> <tr><td>t_2</td><td>White</td><td>8/25/65</td><td>02138</td></tr> <tr><td>t_3</td><td>White</td><td>8/25/65</td><td>02135</td></tr> <tr><td>t_4</td><td>White</td><td>8/25/65</td><td>02135</td></tr> <tr><td>t_5</td><td>Black</td><td>6/20/66</td><td>02137</td></tr> <tr><td>t_6</td><td>Black</td><td>6/20/66</td><td>02137</td></tr> <tr><td>t_7</td><td>Black</td><td>6/20/66</td><td>02137</td></tr> <tr><td>t_8</td><td>*****</td><td>**/**/**</td><td>**/**/**</td></tr> <tr><td>t_9</td><td>*****</td><td>**/**/**</td><td>**/**/**</td></tr> </tbody> </table>		Race	Birth	ZIP	t_1	White	8/25/65	02138	t_2	White	8/25/65	02138	t_3	White	8/25/65	02135	t_4	White	8/25/65	02135	t_5	Black	6/20/66	02137	t_6	Black	6/20/66	02137	t_7	Black	6/20/66	02137	t_8	*****	**/**/**	**/**/**	t_9	*****	**/**/**	**/**/**
	Race	Birth	ZIP																																																																																
t_1	White	8/25/65	0213*																																																																																
t_2	White	8/25/65	0213*																																																																																
t_3	White	8/25/65	0213*																																																																																
t_4	White	8/25/65	0213*																																																																																
t_5	Black	6/20/66	02137																																																																																
t_6	Black	6/20/66	02137																																																																																
t_7	Black	6/20/66	02137																																																																																
t_8	*****	**/**/**	**/**/**																																																																																
t_9	*****	**/**/**	**/**/**																																																																																
	Race	Birth	ZIP																																																																																
t_1	White	8/25/65	02138																																																																																
t_2	White	8/25/65	02138																																																																																
t_3	White	8/25/65	02135																																																																																
t_4	White	8/25/65	02135																																																																																
t_5	Black	6/20/66	02137																																																																																
t_6	Black	6/20/66	02137																																																																																
t_7	Black	6/20/66	02137																																																																																
t_8	*****	**/**/**	**/**/**																																																																																
t_9	*****	**/**/**	**/**/**																																																																																

Fig.2 K -anonymized table achieved using different generalization strategies

图 2 不同概括策略得到的 K -匿名化表

朴素 Classfly⁺算法信息损失较大,因为子表中某些元组虽然符合独立约束子集 $IndepCSet$,但由于不符合 $IndepCSet$ 的联合约束,进而参与之后的概括操作.例如,图 1 中元组集 $\{t_1,t_2,t_3,t_4\}$ 符合 $IndepCSet=\{C_1,C_2\}$,但由于不符合其联合约束而被概括成如图 2 所示的表 T_a .如果将不符合约束集的元组作为候选元组集,根据定理 4,候选元组集基于约束集的最大匿名元组子集是唯一的,与 Classfly 的元组概括过滤策略类似,符合约束集的最大匿名元组子集不必参与之后的概括操作.下面给出两种从候选元组集过滤出最大匿名元组子集的策略.

3.4 完整 $\text{IndepCSet Classfly}^+$ 算法

当针对独立约束子集 IndepCSet 采用概括对子表 T 进行 K -匿名化处理时,将当前不符合 IndepCSet 的元组集作为候选元组集,然后不断地从候选元组集里将不符合 IndepCSet 的元组淘汰,直到候选元组集为空或者不再有候选元组被淘汰为止.这种方法从候选元组集中获得匿名元组子集,由于任意被淘汰的元组与该匿名元组子集的并集都不符合约束集,因此,所得的匿名元组子集是符合 IndepCSet 的最大匿名元组子集,不必参与之后的概括操作,而淘汰元组被概括后作为候选元组集重复上面的概括过滤操作.这种方法称为完整 $\text{IndepCSet Classfly}^+$ (complete $\text{IndepCSet Classfly}^+$).

算法 3. Complete $\text{IndepCSet Classfly}^+$ 算法中多约束元组概括过滤策略.

输入:独立约束子集 $\text{IndepCSet}=\{C_1,\dots,C_H\}$ 及其对应子表 T ;

输出:符合 IndepCSet 的子表 GT .

步骤:

- (1) 初始子表 GT 中每个元组标记设为 $to_be_suppressed$; // $to_be_suppressed$ 表示要被隐匿;
- (2) 当所有标记为 $to_be_suppressed$ 的元组数目之和 $\geq \max\{K_1,\dots,K_H\}$ 时,循环执行以下操作:
 - (a) 将标记为 $to_be_suppressed$ 的元组的标记设为 $candidate$; // $candidate$ 表示候选;
 - (b) 循环执行以下操作,直到没有标记为 $candidate$ 的元组的标记被设为 $to_be_suppressed$ 时结束:
对于约束 $C_i(1 \leq i \leq H)$:遍历标记为 $candidate$ 的元组,若某个元组 t 不符合 C_i ,则将所有与 t 在 QI_i 上具有相同投影的元组标记为 $to_be_suppressed$;
 - (c) 将标记为 $candidate$ 的元组标记为 $satisfy_IndepCSet$; // $satisfy_IndepCSet$ 表示符合 IndepCSet ;
 - (d) 在标记为 $to_be_suppressed$ 的元组中选择具有最少相同取值的属性进行概括;
- (3) 隐匿标记为 $to_be_suppressed$ 的元组.

3.5 部分 $\text{IndepCSet Classfly}^+$ 算法

与完整 $\text{IndepCSet Classfly}^+$ 算法相似,从候选元组集里过滤独立约束子集 IndepCSet 的最大匿名元组子集时,可以先从候选元组集里淘汰不符合 $\{C_1\}$ 的元组,然后再淘汰不符合 $\{C_1,C_2\}$ 的元组,以此类推,直到从候选元组集里淘汰不符合 IndepCSet 的元组为止.与完整 $\text{IndepCSet Classfly}^+$ 算法的原理相同,候选元组集为初始候选元组集基于 IndepCSet 的最大匿名元组子集,不参与之后的概括操作.这种基于 IndepCSet 通过不断扩大约束条件从候选元组集过滤出符合 IndepCSet 的最大匿名元组子集而使其不参与概括的方法,称为部分 $\text{IndepCSet Classfly}^+$ (partial $\text{IndepCSet Classfly}^+$).

算法 4. Partial $\text{IndepCSet Classfly}^+$ 算法中多约束元组概括过滤策略.

输入:独立约束子集 $\text{IndepCSet}=\{C_1,\dots,C_H\}$ 及其对应子表 T .

输出:符合 IndepCSet 的子表 GT .

步骤:

- (1) 初始子表 GT 中每个元组标记设为 $to_be_suppressed$; // $to_be_suppressed$ 表示要被隐匿
- (2) 当所有标记为 $to_be_suppressed$ 的元组数目之和 $\geq \max\{K_1,\dots,K_H\}$ 时循环执行以下操作:
 - (a) 将标记为 $to_be_suppressed$ 的所有元组的标记设为 $candidate$; // $candidate$ 表示候选;
 - (b) $i=1$,当 $i \leq H$ 时执行如下循环操作:
 $i++$;执行下面循环操作,直到没有元组的标记由 $candidate$ 变成 $to_be_suppressed$:
遍历标记为 $candidate$ 的元组,若某个元组 t 不符合 $C_j(1 \leq j \leq i)$,则将标记为 $candidate$ 的元组中所有与 t 在 QI_j 上投影相等的元组标记为 $to_be_suppressed$;
- (3) 将所有标记为 $candidate$ 的元组的标记设为 $satisfy_IndepCSet$; // $satisfy_IndepCSet$ 表示符合 IndepCSet ;
- (4) 在标记为 $to_be_suppressed$ 的元组中选择具有最少相同取值的属性进行概括;

(5) 隐匿标记为 *to_be_suppressed* 的元组.

通过分析完整和部分 *IndepCSet Classfly⁺* 算法的元组概括过滤策略可知,这两种方法都获得了候选元组集基于对应独立约束子集的最大匿名元组子集.根据定理 4 可知,对于相同的元组集 T ,两种算法对子表 T 进行匿名化处理后的结果相同,即信息损失相同,只是由于最大匿名元组子集过滤策略的不同导致算法的效率有所差异.

4 性能分析与评价

本节给出了 *Datafly*, *Naive Classfly⁺*, 完整 *IndepCSet Classfly⁺* 和部分 *IndepCSet Classfly⁺* 的 K -匿名化算法(后面分别简记为 D, N, C, P)的信息损失以及性能的测试结果,并进行分析比较.从以下两方面比较其性质:(1) 变化数据集大小;(2) 变化约束集.表 1 给出性能测试所使用的数据集描述,包括属性以及每种属性的数据类型,表 2 和表 3 给出了实验中设定的约束以及各种测试情况下约束集的描述.

使用具有 9 个属性的表作为数据集.数据集中的各个特征值是均匀分布的,实验数据由一个时间函数随机产生.选定的数据集大小从 1 024 变化到 10 240(表示为 1K~10K),采用包含 2~8 个约束的约束集.所有测试使用的软硬件环境如下:(1) 硬件环境: Intel Pentium4 2.4GHz CPU, 512MB SDRAM 内存;(2) 操作系统平台: Microsoft Windows 2000 Server;(3) 编程环境: Microsoft Visual C++ 编译器.

Table 1 Dataset for the experimental test

表 1 实验测试中的数据集

Attribute	Race	Birth	Sex	ZIP	Height	Weight	Work_Hrs	Salary	Edu
Type	Category	Date	Category	Number	Number	Number	Number	Number	Category

Table 2 Constraints for the experimental test

表 2 实验测试中采用的约束

Constraints	Expressions	Constraints	Expressions
C_1	$\langle\{Race, Birth, Sex\}, 5\rangle$	C_5	$\langle\{Weight, Work_Hrs, Salary\}, 6\rangle$
C_2	$\langle\{Birth, Sex, ZIP\}, 3\rangle$	C_6	$\langle\{Work_Hrs, Salary, Edu\}, 3\rangle$
C_3	$\langle\{Sex, ZIP\}, 5\rangle$	C_7	$\langle\{Race, Birth, Edu\}, 4\rangle$
C_4	$\langle\{ZIP, Height\}, 3\rangle$	C_8	$\langle\{ZIP, Weight\}, 8\rangle$

Table 3 Constraint sets for the experimental test

表 3 实验测试中采用的约束集

Number of constraints	Constraint sets
$ CSet =2$	$CSet=\{C_1, C_2\}$
$ CSet =3$	$CSet=\{C_1, C_2, C_3\}$
$ CSet =4$	$CSet=\{C_1, C_2, C_3, C_4\}$
$ CSet =5$	$CSet=\{C_1, C_2, C_3, C_4, C_5\}$
$ CSet =6$	$CSet=\{C_1, C_2, C_3, C_4, C_5, C_6\}$
$ CSet =7$	$CSet=\{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$
$ CSet =8$	$CSet=\{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$

4.1 K -匿名化信息损失程度分析

本文采用精度^[10]描述一个表的信息损失程度.由于完整 *IndepCSet Classfly⁺* 和部分 *IndepCSet Classfly⁺* 都是从候选元组集过滤出最大匿名元组子集,根据定理 4 可知,这两种算法 K -匿名化的数据精度相同.

图 3(a)和图 3(b)给出了当约束集分别为 $\{C_1, C_2\}$ 和 $\{C_1, C_2, C_3, C_4\}$ 时,不同 K -匿名化方法在数据量从 1K~10K 变化时的精度比较.

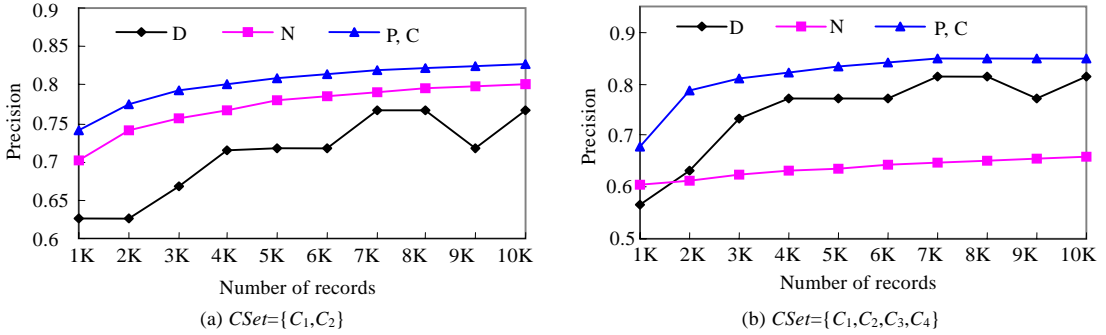


Fig.3 Comparison of precision using different K -anonymization approaches

图 3 不同 K -匿名化算法的精度对比

从图 3 可以看出,当数据量和约束集相同时,部分 IndepCSet Classfly+方法、完整 IndepCSet Classfly+方法(简称 P,C 方法) K -匿名化的精度最高,平均情况下,P,C 方法 K -匿名化的精度比 Datafly 高 10%~20%.Datafly 方法采用概括来 K -匿名化数据时,是以整个数值域为单位进行概括的,精度必然比 P,C 方法的精度要低.Naive Classfly+基于独立约束子集的联合约束采用 Classfly 方法对相应子表进行 K -匿名化处理.根据定理 3,符合约束集的元组不一定符合其联合约束,因此,符合约束集的某些元组仍然参与后面的概括,所以 Naive Classfly+方法的精度比 P,C 方法的精度要低.当约束集是 $\{C_1, C_2\}$ 时,Naive Classfly+ K -匿名化的精度比 Datafly 要高,这是因为此时 Datafly K -匿名化后,每个属性处于较高的概括层次,导致 Datafly 的信息损失比 Naive Classfly+的要大.当约束集是 $\{C_1, C_2, C_3, C_4\}$ 时,Naive Classfly+方法 K -匿名化的精度比 Datafly 的要低,这是因为约束集 $\{C_1, C_2, C_3, C_4\}$ 的联合约束的 K 值比每个约束的 K 值大得多,导致 Naive Classfly+比 Datafly 的信息损失要大.

从图 4 可以看出,当约束集相同时,经过 Naive Classfly+方法、完整 IndepCSet Classfly+方法和部分 IndepCSet Classfly+方法进行 K -匿名化处理的数据精度随数据量增大而增加.其原因在于,数据量越大,表中元组数值差异性越大,元组集里符合约束集条件的最大匿名元组子集中包含的元组数目越多,于是,参与进一步概括操作的元组所占比例减小,从而导致 K -匿名化的精度提高.

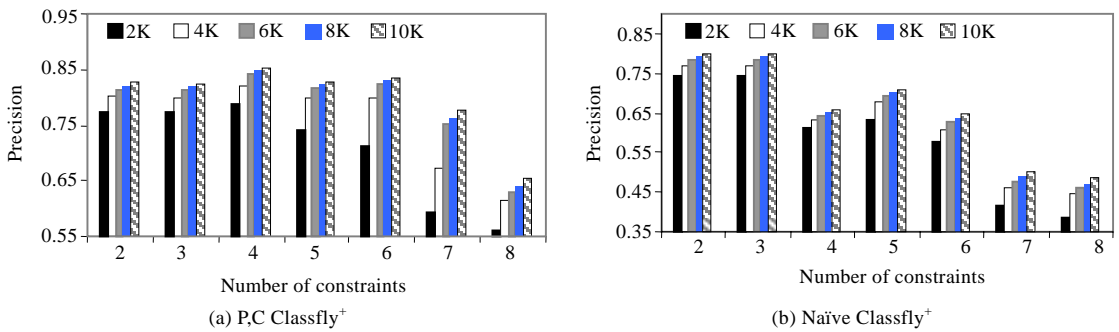


Fig.4 Precisions of algorithms under fixed constraint set and varied database size

图 4 固定约束集情况下不同 K -匿名化算法的精度随数据量的变化

4.2 执行时间分析

Datafly 方法、Naive Classfly+方法、完整 IndepCSet Classfly+方法和部分 IndepCSet Classfly+方法 K -匿名化数据时所采用的元组概括过滤策略不同,导致每种 K -匿名化方法所需的执行时间不同.

图 5 比较了在不同数据量(2K~10K)、不同约束条件下、不同 K -匿名化方法的执行时间.

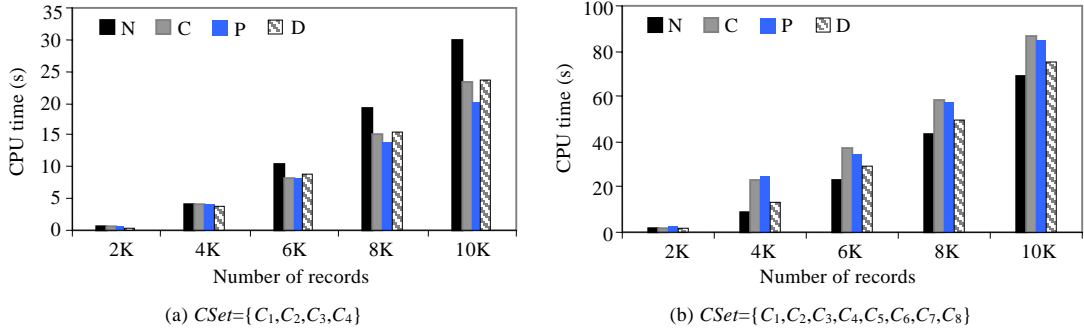


Fig.5 Comparison of CPU time using different K-Anonymization algorithms

图 5 不同 K-匿名化算法的 CPU 时间对比

从图中可以看出,完整 IndepCSet Classfly⁺和部分 IndepCSet Classfly⁺的执行时间相差不多.对于图 5(a), Naïve Classfly⁺方法的执行时间比其他 K-匿名化方法的执行时间要多很多,原因在于,对约束集 $CSet = \{C_1, C_2, C_3, C_4\}$ 进行划分之后只有一个独立约束子集,其联合约束为 $C = (\{\text{民族, 出生日期, 性别, 邮编, 身高}\}, 5)$, Naïve Classfly⁺方法基于 C 采用 Classfly 算法 K-匿名化数据,虽然约束数目由 4 个转化为 1 个,减少了检验元组是否符合约束集的时间,但根据定理 3 可知, C 比 CSet 的要求高,很多符合 CSet 的元组由于不符合 C 而参与概括,相应地增加了执行时间,导致此种约束集下 Naïve Classfly⁺方法的执行时间比其他 K-匿名化方法的执行时间要大.因此,Naïve Classfly⁺方法的执行时间与约束集相关.在图 5(b)中,在 $|CSet|=8$ 的约束集下, Datafly 算法的执行时间比其他 K-匿名化方法的执行时间相对少一些,因为 Datafly 算法在 $|CSet|$ 较大时,其全域概括的性质使得表中元组更快地趋于符合约束集,而约束数目的增加,使得其他 K-匿名化方法在检验元组是否符合约束集方面所需时间相应地有所增加,导致 Datafly 算法的执行时间相对少一些.不过, Datafly 在执行时间方面的微弱优势是以大量信息损失为代价的.

图 6(a)~图 6(c)分别显示了 3 种 K-匿名化方法的执行时间随约束数目和数据量的变化情况.从图中可以看出,对于任何一种 K-匿名化方法,当数据量相同时, K-匿名化的执行时间随约束数目的增加而增大.当约束增加时,元组概括过滤需要更多时间搜索符合约束集的最大匿名元组子集,因此, K-匿名化执行时间增加.当约束集不变时,对于任何一种 K-匿名化方法, K-匿名化的执行时间随数据量的增大而增加,因为数据量增大时,元组概括过滤需要搜索更大的数据空间.

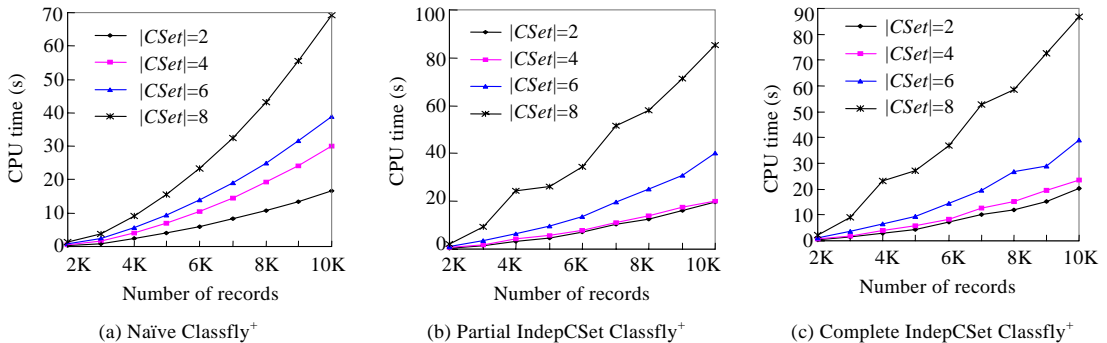


Fig.6 CPU time of algorithms under varied number of constraints and database sizes

图 6 CPU 时间随约束数目和数据量的变化

5 结论

支持多约束的 K-匿名化处理是 K-匿名研究中的一个重要问题,而 K-匿名化策略的选取直接影响 K-匿名化

的精度以及执行时间的大小.本文针对多约束的特点,提出了多约束的相关概念和定理,给出一种新的支持多约束 K -匿名化处理的约束集划分策略,并在此基础上,将独立约束子集的概念引入单一约束 K -匿名化方法 Classfly 中,提出了基于独立约束子集的多约束 K -匿名化方法——Classfly⁺.Classfly⁺引入Classfly的元组概括过滤思想,基于独立约束子集过滤相应候选元组集的最大匿名元组子集.给出了 3 种 Classfly⁺方法的性能测试结果.实验结果表明,完整和部分 IndepCSet Classfly⁺方法 K -匿名化的精度比 Datafly 提高了 10%~20%.

References:

- [1] Sweeney L. K -Anonymity: A model for protecting privacy. *Int'l Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, 2002,10(5):557–570.
- [2] Liu XY, Yang XC, Yu G. A representative classes based privacy preserving data publishing approach with high precision. *Computer Science*, 2005,32(9A):368–373 (in English with Chinese abstract).
- [3] Meyerson A, Williams R. On the complexity of optimal k -anonymity. In: Deutsch A, ed. *Proc. of the 23rd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS 2004)*. New York: ACM, 2004. 223–228.
- [4] Aggarwal G, Feder T, Kenthapadi K, Motwani R, Panigrahy R, Thomas D, Zhu A. Anonymizing tables. In: Eiter T, Libkin L, eds. *Proc. of the 10th Int'l Conf. on Database Theory (ICDT 2005)*. LNCS 3363, Springer-Verlag, 2005. 246–258.
- [5] Iyengar V. Transforming data to satisfy privacy constraints. In: Zaïane O, ed. *Proc. of the 8th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD 2002)*. New York: ACM, 2002. 279–288.
- [6] Yao C, Wang XS, Jajodia S. Checking for k -anonymity violation by views. In: Böhm K, Jensen CS, Hass LM, Kersten ML, Larson P, Ooi BC, eds. *Proc. of the 31st Int'l Conf. on Very Large Data Bases (VLDB 2005)*. Trondheim: ACM, 2005. 910–921.
- [7] Sweeney L. Guaranteeing anonymity when sharing medical data, the Datafly system. In: Masys DR, ed. *Proc. of the 1997 American Medical Informatics Association Annual Fall Symp. (AMIA'97)*. 1997. 51–55. <http://www.amia.org/pubs/symposia/D004462.pdf>
- [8] LeFevre K, DeWitt D, Ramakrishnan R. Incognito: Efficient full-domain k -anonymity. In: Ozcan F, ed. *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*. New York: ACM, 2005. 49–60.
- [9] Fung B, Wang K, Yu P. Top-Down specialization for information and privacy preservation. In: Toyama M, Sasaki S, eds. *Proc. of the 21st Int'l Conf. on Data Engineering (ICDE 2005)*. Tokyo: IEEE Computer Society, 2005. 205–216.
- [10] Sweeney L. Achieving k -anonymity privacy protection using generalization and suppression. *Int'l Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, 2002,10(5):571–588.



杨晓春(1973 -),女,辽宁沈阳人,博士,副教授,CCF 高级会员,主要研究领域为分布式数据管理,访问控制.



王斌(1972 -),男,博士生,讲师,主要研究领域为分布式数据处理,体系结构.



刘向宇(1981 -),男,硕士生,主要研究领域为数据隐私保护.



于戈(1962 -),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库理论与技术.