概念格的对象渐减更新算法

王静宇 郑雪岩
(内蒙古科技大学信息工程学院 内蒙古 包头 014010)

摘 要 为解决访问控制中的删除某个对象后重新构造概念格耗时的问题,对概念进行了分类,深入研究了概念格中删除对象后各个概念以及边的变化,分析了概念之间及边之间的联系和规则,在此基础上提出一种概念格的对象渐减更新算法。该算法采用渐进式构造方法,不需要重新构造概念格,而且是在原概念格的基础上采用广度优先遍历的顺序对概念格进行调整,进而可根据部分父概念的类型来直接判断子概念的类型,无须判断所有概念的类型。实验表明,需调整的概念数量占概念总数的比例较小,该算法减少了概念格构造的时间。

关键词 访问控制 分类 渐进式 概念格 删除对象

中图分类号 TP309 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2022.03.037

AN OBJECT GRADUAL UPDATING ALGORITHM BASED ON CONCEPT LATTICE

Wang Jingyu Zheng Xueyan
(*School of Information Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, Inner Mongolia, China*)

Abstract In order to solve the problem that it takes time to reconstruct the concept lattice after deleting an object in access control, the concepts are classified, the changes of various concepts and edges after deleting objects in the concept lattice are studied in depth, and the relationship and rules between concepts and edges are analyzed. On this basis, an object gradual updating algorithm of concept lattice is proposed. This algorithm used a progressive construction method. It did not need to reconstruct the concept lattice. It used breadth-first traversal based on the original concept lattice. The concept lattice was adjusted to directly determine the type of the child concept according to the type of some parent concepts, without the need to determine the types of all concepts. The experiments show that the number of concepts to be adjusted accounts for a small proportion of the total number of concepts, and this algorithm reduces the time to construct the concept lattice.

Keywords Access control Classification Progressive Concept lattice Delete object

0 引 言

形式概念分析是 Wille[1] 于 1982 年提出的,它的核心数据结构是概念格,是根据描述现实世界的形式背景建立起来的,描述了对象和属性之间的关系。在现实生活中,对象和属性不断发生变化,概念格也随之改变,因此研究概念格的维护是十分必要的。Godin 等[2]提出了一种渐进式的概念格构造算法;概念格的应用涉及到了多个领域,例如信息检索、知识管理、软件工程[3-4]等;吴刚等[5]提出了扩展概念格的维护;谢霖铨等[6]提出了粗糙概念格构造的算法;智慧来等[7]提出了概念格的第一类维护和第二类维护;刘保相等[8]提出了一种区间概念格结构;智慧来等[9]研究了以关系为更新粒度的概念格增量维护;张春英等[10]提出了基于属性幂集的渐进式生成算法;文献[11-12]提出了增加一个对象或属性时的概念格维护;崔芳婷等[13]提出了基于约束的模糊概念格构造算法;王春月等[14]提出了基于二元关系消减的概念格维护算法。

删除对象后概念格会发生改变,为了防止重新构造概念格耗费大量时间,本文借鉴了张磊[15]的构造概念格的方法,对删除对象后概念格的结构变化以及概念之间的联系进行改进,提出一种概念格的对象渐减更新算法。

收稿日期:2019-12-30。国家自然科学基金项目(61662056)。**王静宇**,教授,主研领域:大数据安全。**郑雪岩**,硕士生。

1 相关概念

形式背景是一个矩阵图,表示对象和属性之间的二元关系,矩阵的行表示属性 m、列表示对象 g,形式背景记为 $O=(G,M,R)$,其中:G 是对象的集合;M 是属性的集合;R 表示 G 和 M 之间的关系。若 g 行与 m 列的交叉处为 1,即 $gRm=1$,则表示对象 g 具有属性 m,相应地,$gRm=0$ 表示对象 g 不具有属性 m。形式背景如表 1 所示。

表 1 形式背景示例

用户	权限			
	a	b	c	d
1	0	0	1	1
2	1	1	0	0
3	1	1	1	0
4	0	1	1	1
5	0	0	1	0

定义 1 若 X 是 G 的子集,Y 是 M 的子集,令:

(1) $h(X)=\{m\in M\mid g\in X,gRm\}$。

(2) $h(Y)=\{g\in G\mid m\in Y,gRm\}$。

如果 X、Y 满足 $h(X)=Y$、$h(Y)=X$,则称(X,Y)是形式背景的概念 Node,简称 N,$X(x_1,x_2,\cdots,x_n)$是概念(X,Y)的外延,记为 $Extension(N)$,简称 $E(N)$,$Y(y_1,y_2,\cdots,y_n)$是概念(X,Y)的内涵,记为 $Intension(N)$,简称 $I(N)$,概念的集合记为 $NS(O)$。

定义 2 设(X_1,Y_1)和(X_2,Y_2)是形式背景的两个概念,如果 $X_1\subseteq X_2$,且不存在 X_3,使得 $X_1\subseteq X_3\subseteq X_2$,则称$(X_1,Y_1)$是$(X_2,Y_2)$的子概念,$(X_2,Y_2)$是$(X_1,Y_1)$的父概念,记为$(X_1,Y_1)\leqslant(X_2,Y_2)$。$(X_1,Y_1)$和$(X_2,Y_2)$是一对父子概念对,所有父子概念对连在一起构成了概念格的 Hasse 图,在 Hasse 图中,把连接两概念之间的线称为边。表 1 的形式背景所对应的概念格的 Hasse 图如图 1 所示。

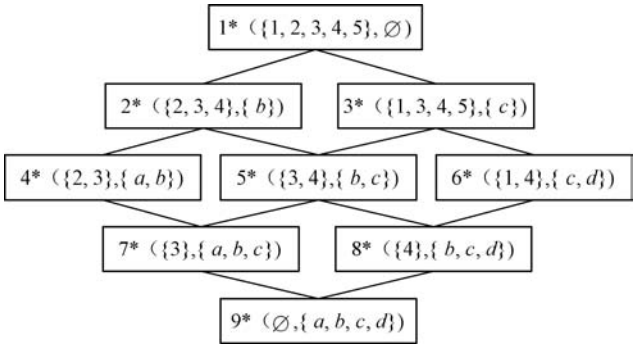

图 1 表 1 的形式背景对应的概念格的 Hasse 图

定义 3 对于概念(X_1,Y_1)和概念(X_2,Y_2),$X_1\subseteq X_2$,如果(X_x,Y_x)是(X_1,Y_1)到(X_2,Y_2)之间的路径上唯一的一个概念,则称(X_x,Y_x)是(X_1,Y_1)和(X_2,Y_2)的枢纽概念。

2 概念格

2.1 概念的分类

记删除对象 x 后的概念格为 $L(O\mid^{-\{x\}})$,原概念格为 $L(O)$,$L(O)$中的概念由如下三种类型组成:

(1) 不变概念:指外延中不含删除对象 x 的概念,即 $x\notin Extension(N)$ 的概念。这一类概念的集合用 $FS^{-\{x\}}(O)$来表示。

(2) 更新概念:若 N_1 为 N 的子概念,则更新概念是指外延包含 x,$\nexists\, Extension(N_1)=E(N)-\{x\}$ 的概念。这一类概念的集合用 $VS^{-\{x\}}(O)$来表示。

(3) 删除概念:指外延包含 x,$\exists\, Extension(N_1)=E(N)-\{x\}$ 的概念。这一类概念的集合用 $DS^{-\{x\}}(O)$来表示。

2.2 概念的分析

定理 1 删除对象 x 后,原概念格中的不变概念不发生任何变化,直接保留到新的概念格中,即 $FS^{-\{x\}}(O)\subseteq NS(O\mid^{-\{x\}})$。

定理 2 删除对象 x 后,更新概念的外延减去 x 即可成为新概念格中的概念,即 $VS^{-\{x\}}(O)\subseteq NS(O\mid^{-\{x\}})$。

定理 3 删除对象 x 后,新概念格中的概念是由 $FS^{-\{x\}}(O)$和 $VS^{-\{x\}}(O)$组成的。

由上文可知,新概念格由不变概念和更新概念组成,删除对象后,不变概念不发生任何变化,直接保留到 $L(O\mid^{-\{x\}})$中,更新概念的外延减去 x,即可成为 $L(O\mid^{-\{x\}})$中的概念,删除概念需删掉。

定理 4 包含所有对象的概念是头概念,头概念必定是更新概念,因此构造 $L(O\mid^{-\{x\}})$时不需要判断它的类型,直接更新外延即可。

定理 5 包含所有属性的概念是末概念。末概念的外延不包含任何对象,所以末概念必定是不变概念,构造 $L(O\mid^{-\{x\}})$时不需要判断它的类型。

2.3 边的分析

概念格是由概念和概念之间的边组成的,因此删除某一个对象后,不仅要考虑新概念格中概念的组成,还要考虑概念之间的边的变化。边都是存在于父概念和子概念之间的,所以我们来根据父概念和子概念之

间的关系去判断边的变化。

定理 6　不变概念的子概念还是不变概念。

由定理 6 可知,以下两种情况不存在:

(1) 父概念为不变概念,子概念为删除概念。

(2) 父概念为不变概念,子概念为更新概念。

定理 7　如果父概念和子概念中没有一个是删除概念,则该父概念和子概念之间的边不需要改变,保留到新的概念格中。

由定理 6 和定理 7 可知,以下三种情况不需要调整边:

(1) 父概念是更新概念,子概念是不变概念。

(2) 父概念是更新概念,子概念是更新概念。

(3) 父概念是不变概念,子概念是不变概念。

定理 8　删除对象 x 后,若 $Extension(N)-\{x\}=\varnothing$,则在删除概念的父概念与其子概念之间增加一条边;若 $Extension(N)-\{x\}!=\varnothing$,且概念是其父概念与子概念之间的枢纽概念,则在删除概念的父概念与其子概念之间增加一条边。

定理 9　一个删除概念的子概念中只有一个不变概念,其他都是删除概念。

由定理 9 可知,父子概念分别为删除概念和更新概念的情况不存在。

综上所述,删除对象后边的关系如表 2 所示。

表 2　删除对象后各父概念与子概念之间边的变化

父概念	子概念	边的变化
更新概念	删除概念	删除,根据定理 3 判断是否在更新概念和删除概念的子概念之间增加边
删除概念	不变概念	删除,根据定理 3 判断是否在删除概念的父概念和不变概念之间增加边
删除概念	删除概念	删除
更新概念	不变概念	不变
更新概念	更新概念	不变
不变概念	不变概念	不变

根据前文中对删除对象后概念和边的分析可知:删除对象后,父概念与子概念的类型有一定的联系,父概念与子概念之间的边也存在一定的变化规则。据此提出一种渐进式的概念格构造算法,即对象渐减更新算法。

3　算法描述

3.1　算法思想

根据前文所述可知,不变概念的子概念依然是不变概念,删除概念的非不变子概念是更新概念,不需要判断这两种概念的类型。算法主要解决的是删除一个对象后,如何得到一个新的概念格,该算法采用广度优先遍历的顺序,以头概念为顶点,依次对概念进行处理,因此访问某个概念的时候,它的父概念已经被访问过,进而该算法可根据父概念的类型来判断子概念的类型。

该算法不需要判断不变概念的子概念,所以只需要判断更新概念和删除概念的子概念即可。在该算法中设未被访问过的概念的 *visited* 的值为 0,被访问过的概念的 *visited* 的值为 1。概念的 *visited* 的值为 0 时算法向下执行,所有概念的 *visited* 的值为 1 时算法结束。

3.2　对象渐减更新算法

算法 1 的相关术语:*Child*(*FS*)用来表示不变概念的子概念;*Child*(*VS*)用来表示更新概念的子概念;*Child*(*DS*)用来表示删除概念的子概念。

在算法 1 中,直接对头概念进行更新,然后算法向下执行。如果是更新概念的子概念,则判断其类型并进行相应的处理(4 – 17 行);如果是删除概念的非不变子概念,则该概念必然是删除概念,此时调用算法 2 对概念进行删除操作并修改相关的边的关系(18 – 22 行),直到所有概念的 *visited* 的值为 1。

算法 1　对象渐减更新算法

输入:原始概念格 $L(O)$,删除对象 x。

输出:删除对象 x 后的格 $L(O|^{-\{x\}})$。

```
1.  Extension(N) := Extension(N) - {x};          //更新头概念
2.  While N.visited := 0                  ///当概念未被访问过的时候
3.     For (N ∉ Child(FS))
4.        For (N ∈ Child(VS))
5.           If (x ∉ Extension(N))
6.              doNotChange;                        //不做改变
7.              N.visited := 1;
8.           Else If ( ∃Extension(N) := Extension(N) - {x})
9.           Delete(L(O), N);
                          //调用算法 2,删除概念并调整相应的边
10.          N.visited := 1;
11.       Else If
12.          Extension(N) := Extension(N) - {x};
                                          //更新概念的外延
13.          N.visited := 1;
14.       End If;
15.    End If;
16.       End If;
17.    End For;
18.    For (N ∈ Child(DS))
19.    If(N ∈ Child(DS) and 非 FN)
```

20. Delete($L(O)$, N);
21. $N.visited$: =1;
22. End For;
23. End For;
24. End While;
25. Return $L(O|^{-\{x\}})$;

算法 2 的相关术语:N_{Child}用来表示概念的子概念;N_{Parent}用来表示概念的父概念。

该算法用于将概念删除并调整其相关边的关系,在该算法中,符合下面两种情况的需要添加边:

(1) 对于外延不只由 x 组成的概念,如果它是任意两个概念之间的枢纽概念,则在这两个概念之间添加边。

(2) 对于外延只由 x 组成的概念,直接在其父概念与子概念之间添加边。

算法 2 删除算法

输入:概念格 $L(O)$,删除概念 N。

输出:删除概念 N 后的格 $L(O|^{-\{x\}})$。

1. If ($Extension(N)-\{x\}$! = $\varnothing$ and N 是 $N_{Parent}\rightarrow N_{Child}$的枢纽概念)
2. Then 删除与 N 相关的边,添加边 $N_{Parent}\rightarrow N_{Child}$,删除 N;
3. Else 删除与 N 相关的边,删除 N;
4. If ($Extension(N)-\{x\}=\varnothing$)
5. Then 删除与 N 相关的边,添加边 $N_{Parent}\rightarrow N_{Child}$,删除 N;
6. Else 删除与 N 相关的边,删除 N;
7. End

4 算法示例及实验验证

4.1 算法示例

以图 1 为例,删除对象 4,算法的维护过程如下:

(1) 头概念是更新概念,直接更新。

(2) 概念 2^* 的外延去除对象 4 后与概念 4^* 的外延相同,因此 2^* 是删除概念,删除 2^*,删除边(1^*,2^*),删除边(2^*,4^*),删除边(2^*,5^*),且 2^* 是 1^* 与 4^* 之间的枢纽概念,故添加边(1^*,4^*)。

(3) 3^* 的外延减去4 后与其任何一个子概念延都不相等,故 3^* 为更新概念,更新 3^* 的外延。

(4) 因为 4^* 和 5^* 都是 2^* 的子概念,且 4^* 是不变概念,故 5^* 为删除概念,删除 5^*,删除边(3^*,5^*),删除边(5^*,7^*),删除边(5^*,8^*),因为 5^* 是 3^* 和 7^* 之间的枢纽概念,故增加边(3^*,7^*)。

(5) 6^* 的外延减去 x 后与其任何一个子概念延都不相等,故 6^* 为更新概念,更新 6^* 的外延。

(6) 7^* 的外延不包含对象 4,故 7^* 为不变概念,不作任何改变。

(7) 8^* 的外延只包含对象 4,符合 $E(N)-\{x\}=\varnothing$,故删除概念 8^*,添加边(6^*,9^*),删除边(6^*,8^*),删除边(8^*,9^*)。

删除对象 4 后的概念格对应的 Hasse 图如图 2 所示。

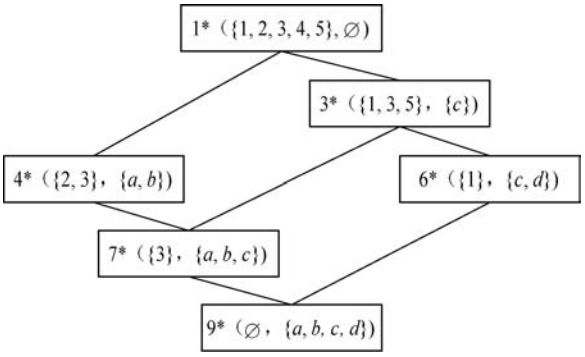

图 2 删除对象 4 后的概念格

4.2 实验验证

删除某一对象后,该渐进式维护算法不需要重新构造概念格,只需对更新概念、删除概念及其相关的边进行处理,因此在很大程度上减少了概念格的维护时间。该算法还包含以下几个优点:

(1) 不需要判断头概念的类型,可直接更新头概念。

(2) 该算法可根据父概念的类型来判断子概念的类型,所以可以减少概念类型的判断次数。

(3) 若概念的外延中只包含删除对象 x,则不必判断概念是否是两个概念之间的枢纽概念,可直接在概念的父概念和其子概念之间添加边。

实验一 验证需调整的概念占总概念的比例。

随机生成形式背景,属性个数固定为 20,对象数目从 10 到 100,每次增加 10 个对象来进行实验。实验结果如图3 所示,纵轴表示需要调整的概念占全体概念的比例;横轴表示对象的数量;两条折线的对象属性间存在关系的概率分别为 0.20 和 0.25。当删除一个对象时,需要调整的概念所占比例较小,而且随着对象数的增加,这个比例会更小,所以相对于重新构造概念格,本文这种渐进式的方式的效率会比较高。

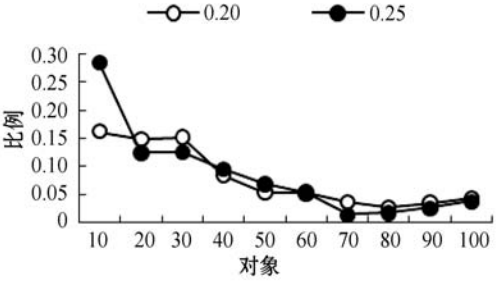

图 3 需要调整的概念占总概念的比例

实验二　主要从时间方面验证算法的有效性。

为了验证本文优化算法的结果,与 AddIntent[16] 算法和 In-Close[17] 算法进行对比。AddIntent 算法是最快的基于对象的概念格构造算法之一,In-Close 算法是近年出现的最快的概念格构造算法之一。我们随机产生了三组数据,三个算法的性能都是这三组数据的平均值,属性的数目都为 20,对象数目从 100 到 900,每次增加 50 个对象来测试算法的执行时间。相对于 In-Close 算法和本文算法,AddIntent 算法消耗的时间比较多,放在同一个图中对比不明显,因此分开来对比。图 4 是本文算法与 AddIntent 算法的对比,图 5 是本文算法与 In-Close 算法的对比,两个图的横轴都表示对象,纵轴都表示算法执行所用的时间。实验结果表明,随着对象数目的增加,三个算法所花费的时间都在增长,但本文算法所用时间明显低于 AddIntent 算法和 In-Close 算法。因此本文的算法明显减少了构造概念格所花费的时间。

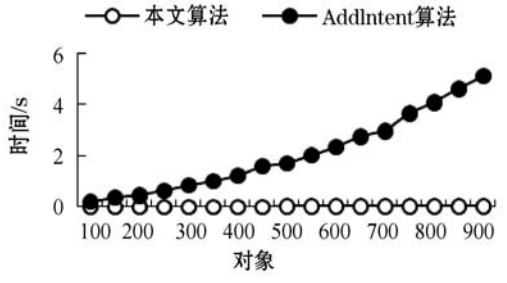

图 4　对象数目增大时本文算法与 AddIntent 算法的时间性能

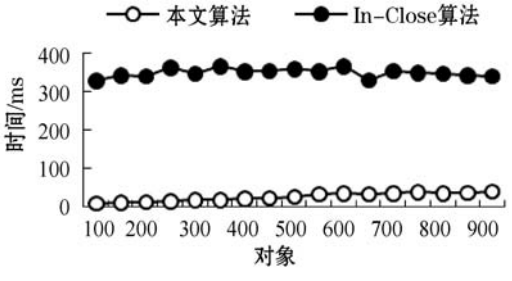

图 5　对象数目增大时本文算法与 In-Close 算法的时间性能

5　结　语

本文根据删除对象后,概念格中父子之间的边以及概念的变化,提出一种概念格渐进式更新算法,该算法减少了概念格构造的时间。

该算法对删除概念的子概念的访问是有序的,而删除概念的非不变子概念肯定是删除概念,如果先找出不变子概念,就不需要判断其余子概念的类型。对于同一个概念的子概念,如果删除概念在不变概念前的情况较多,则算法的执行速度可能会更快,下一步将研究这一方面的内容。

参　考　文　献

[1] Wille R. Restructuring lattice theory: An approach based on hierarchies of concepts[J]. NATO Advanced Study Series, 1982,83:445 – 470.

[2] Godin R, Missaoui R, Alaoui H. Incremental concept formation algorithms based on Galois (concept) lattices[J]. Computational Intelligence,1995,11(2):246 – 267.

[3] Cole R,Stumme G. CEM-A conceptual email manager[C]// 7th International Conference on Conceptual Structures,2000: 438 – 452.

[4] Oourie D G, ObiedOov S, Watson B W, et al. An incremental algorithm to construct a lattice of set intersections [J]. Science of Computer Programming,2009,74(3):128 – 142.

[5] 吴刚,简宋全,胡学钢,等. 扩展概念格的维护[J]. 计算机工程与应用,2002,38(4):76 – 78.

[6] 谢霖铨,付悦华,毛伊敏. 粗糙概念格构造的算法[J]. 计算机工程与设计,2015,36(3):674 – 678,709.

[7] 智慧来,智东杰. 概念格维护原理与算法[J]. 计算机工程与应用,2014,50(6):96 – 101.

[8] 刘保相,张春英. 一种新的概念格结构——区间概念格[J]. 计算机科学,2012,39(8):273 – 277.

[9] 智慧来,智东杰. 关系粒度的概念格增量维护与关联规则更新[J]. 计算机科学,2013,40(4):256 – 258.

[10] 张春英,王立亚. 基于属性集合幂集的区间概念格 L_α ~ β 的渐进式生成算法[J]. 计算机应用研究,2014(3):731 – 734.

[11] 屠莉,陈峻,李云. 一种基于属性的概念格生成及维护算法[J]. 计算机应用,2004,24(10):116 – 118.

[12] 李云. 概念格分布处理及其框架下的知识发现研究[D]. 上海:上海大学,2005.

[13] 崔芳婷,王黎明,张卓. 基于约束的模糊概念格构造算法[J]. 计算机科学,2015,42(8):288 – 293,318.

[14] 王春月,王黎明,张卓. 基于二元关系消减的概念格维护算法[J]. 计算机科学,2016,43(S2):35 – 41.

[15] 张磊. 基于概念格的角色工程相关算法研究[D]. 哈尔滨:哈尔滨工业大学,2015.

[16] Merwe D V D, Obiedkov S, Kourie D. AddIntent: A new incremental algorithm for constructing concept lattices[C]// International Conference on Formal Concept Analysis,2004: 372 – 385.

[17] Andrews S. In-Close, a fast algorithm for computing formal concepts[C]//17th International Conference on Conceptual Structures,2009.