计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
50(2): 248-259, 2013

# 概念格的属性渐减原理与算法研究

张 磊[1,2] 张宏莉[1] 殷丽华[3] 韩道军[2]
[1](哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)
[2](河南大学数据与知识工程研究所 河南开封 475004)
[3](中国科学院信息工程研究所 北京 100093)
(zhanglei@henu.edu.cn)

# Theory and Algorithms of Attribute Decrement for Concept Lattice

Zhang Lei[1,2], Zhang Hongli[1], Yin Lihua[3], and Han Daojun[2]
[1](*School of Computer Science and Technology, Harbin Institute of Technology, Harbin* 150001)
[2](*Institute of Data and Knowledge Engineering, Henan University, Kaifeng, Henan* 475004)
[3](*Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093)

**Abstract** Incremental algorithms for the construction of concept lattices are of key importance. But most of them focus on the case of the addition of objects or attributes in formal context. When the attributes in formal context are deleted, reconstructing concept lattice by these algorithms is needed. It is very time consuming. The theory and algorithms of incrementally obtaining new concept lattice by updating the old one after attributes being deleted are investigated in this paper. At first, the mapping relation between concepts of the old and new concept-lattice is explained and the changes of edges from the old concept lattice to the new one are analyzed. Based on this, two decremental algorithms called top-down and bottom-up algorithms are proposed, by which the original concept lattice can be directly modified to obtain the new one, and reconstructing the whole structure from scratch is avoided. Relying on the structure of concept lattice, the algorithms only explore limited parts of the lattice for modifying. Thus, its time complexity is reduced to $O(\|L\|\cdot\|G\|\cdot\|M\|)$. Experimental results show that the algorithms presented can save considerable time compared with the traditional algorithms.

**Key words** formal concept analysis (FCA); concept lattice; attribute; decremental algorithm; construction

**摘 要** 渐进式算法是概念格构造的一类重要算法,但大多关注于形式背景中对象或属性增加的情况.而当形式背景的属性减少时,已有的算法则需要重新构造概念格,较为费时.针对这一情况,研究了属性消减后从原概念格渐进式产生新概念格的理论和算法,并且算法时间复杂度较低.首先分析了原概念格和新概念格中节点间的映射关系以及从原概念格到新概念格中边(节点间的前驱-后继关系)的变化规律.在此基础上,提出了自顶向下和自底向上两种渐进式的概念格属性渐减算法.算法能够对原有概念格直接进行修改来得到新的概念格,避免了从形式背景重新构造概念格,时间复杂度降低为 $O(\|L\|\cdot\|G\|\cdot\|M\|)$.实验及分析表明,当属性减少时,能比传统算法节省大量的运行时间.

**关键词** 形式概念分析;概念格;属性;渐减算法;构造

**中图法分类号** TP18

**收稿日期**:2010-10-28;**修回日期**:2012-05-02

**基金项目**:国家"九七三"重点基础研究发展计划基金项目(2007CB311101,2011CB302605);国家"八六三"高技术研究发展计划基金项目(2010AA012504,2011AA010705);国家自然科学基金项目(61070186,6110018,61173144)

概念格作为形式概念分析[1-2]的核心数据结构,已经引起了研究人员的广泛关注,并且在信息检索、数据挖掘、社会网络分析、机器学习等诸多领域得到了应用[3-6].由于概念格的时间和空间复杂度是随着形式背景的增加而呈非线性的增大,概念格的构造一直是形式概念分析研究的重点问题之一.现在已经提出了多种构造概念格的算法[7],大致可以分为批处理构造算法[8]和渐进式构造算法[9-12].

在两种构造算法中,由于渐进式构造算法能够在原有的概念格上根据形式背景的变化作相应的调整,避免了从形式背景重新构造概念格,节省了大量的时间.因此一直是概念格构造算法的研究热点.目前,国内外研究人员主要关注于在原概念格上新增对象或者属性情况下的渐增式算法.例如:Godin等人在文献[9]中提出了一种在新增对象的情况下,能够同时对概念格的概念集合和 Hasse 图进行更新的渐进式概念构造算法,该算法是经典的渐进式算法之一.Merwe 等人提出的 AddIntent 算法利用了节点间的父子关系,能够快速地找到标准产生子节点和更新节点,获得了更好的时间性能[10].刘宗田等人在文献[11]中提出了一种模糊概念格模型,并给出了该模型的渐进式构造算法.曲立平等人提出了一种基于属性的渐进式构造算法(FIA_A 算法),该算法采用树结构对节点进行组织,以约束更新节点和产生子格节点的搜索范围,节省了概念格的构造时间[12].Andrews 提出的 In-Close 算法被认为是近年最快的概念格构造算法之一,该算法利用增量闭包和矩阵搜索,快速计算出所有形式概念[13].

这些算法关注于在对象或属性增加的情况下如何快速构造概念格,这适用于一些常见的应用场景.例如,概念格在数据挖掘的应用中,常常将数据库里表的记录(行)作为形式背景的对象,而字段(列)作为形式背景的属性.由于数据库中的记录往往随时间不断增长,与之对应的形式背景的对象也在不断增加.相应地,渐增式的概念格构造算法在此类背景中得到了较为广泛的应用.

随着概念格应用领域的不断增加,越来越需要研究另外一类算法——渐减式算法,即在原概念格基础上减去某些对象或者属性的渐进式算法.尽管这类算法尚未引起研究人员的注意,但是一些实际问题却为这类算法提供了应用场景.例如,在 Web 信息检索中,常常将 Web 文档或资源作为形式背景的对象,用户关注的关键词作为形式背景的属性.随着时间和规则的变化,某些热点关键词可能不再被关注.此时若继续将其保留在概念格中,长期积累下来将会使概念格越来越庞大且无效概念越来越多,使得信息检索的效率下降.因此需要不断将失效的关键词从概念格中删除,进而压缩概念格的规模,提高信息检索的速度.

针对以上这种情况,本文关注于当形式背景的某些属性消减之后,如何快速有效地在原有的概念格上进行调整,得到新形式背景的概念格,而不是传统方式下的重新构造.本文在分析属性消减后原概念格与新概念格之间节点的映射关系和边(节点的前驱-后继关系)的变化规律的基础上,研究在原概念格基础上减去某些属性的渐减式算法.根据对概念格的遍历方式的不同,提出了自上而下和自底向上两种渐进式算法.本文的算法能够同时对概念格的节点和 Hasse 图进行渐进式的调整,具有良好的时间性能.

需要指出的是,概念格的属性约简理论[14]与本文所研究的属性消减都是关注属性的减少,但是侧重点不同.前者的主要工作是寻找最小的属性子集,使之能够完全确定原始形式背景上的概念及其层次结构,约简前后的概念格保持同构关系.而本文主要研究属性消减之后概念格的变化,属性消减前后的概念格不一定保持同构关系.本文提出的概念格属性渐减算法能够结合实际应用需求,渐进式地删除概念格中的选定属性,并快速得到删除选定属性之后的新概念格.

## 1　概念格的基本概念

在形式概念分析[1-2]中,形式背景是一个三元组 $K=(G,M,I)$,如表 1 所示,其中 $G$ 是对象集合,$M$ 为属性集合,$I$ 是 $G$ 和 $M$ 间的二元关系.对于一个对象 $x\in G$,属性 $m\in M$,那么 $xIm$ 就表示对象 $x$ 具有属性 $m$.

**Table 1　An Example of Formal Context**

**表 1　形式背景示例**

| Object | Attribute | | | | |
|---|---|---|---|---|---|
| | $a$ | $b$ | $c$ | $d$ | $e$ |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 0 | 0 | 1 |

在形式背景的对象集 $A\subseteq G$ 和属性集 $B\subseteq M$ 之间定义如下两个映射:

$$f(A)=\{m\in M\mid \forall x\in A, xIm\};$$
$$g(B)=\{x\in G\mid \forall m\in B, xIm\}.$$

如果一个二元组 $C=(A,B)$ 满足 $A=g(B)$, $B=f(A)$,则二元组$(A,B)$被称为一个形式概念(简称概念).其中 $A$ 称为概念 $C$ 的外延,记作 $Ext(C)$, $B$ 是称为概念 $C$ 的内涵,记作 $Int(C)$.

形式背景 $K$ 的所有形式概念的集合被标记为 $CS(K)$.设有概念 $C_1=(A_1,B_1)$ 和 $C_2=(A_2,B_2)$,若满足 $A_1\subseteq A_2$(等价于 $B_1\supseteq B_2$),则称 $C_1$ 为 $C_2$ 的亚概念,$C_2$ 为 $C_1$ 的超概念,并记为$(A_1,B_1)\leqslant(A_2,B_2)$.由偏序关系$\leqslant$诱导的格被称为形式背景 $K$ 的概念格,记为 $L(K)$.特别地,若不存在 $C_3=(A_3,B_3)$,使得$(A_1,B_1)\leqslant(A_3,B_3)\leqslant(A_2,B_2)$,则称$(A_1,B_1)$为子概念,$(A_2,B_2)$为父概念,并记为$(A_1,B_1)\prec(A_2,B_2)$.

形式背景 $K=(G,M,I)$上的概念有以下基本性质($\forall A,A_1,A_2\subseteq G,\forall B,B_1,B_2\subseteq M$)[2]:

**性质 1.** $A_1\subseteq A_2\Rightarrow f(A_1)\supseteq f(A_2)$, $B_1\subseteq B_2\Rightarrow g(B_1)\supseteq g(B_2)$.

**性质 2.** $A\subseteq g(f(A))$, $B\subseteq f(g(B))$.

**性质 3.** $f(g(f(A)))=f(A)$, $g(f(g(B)))=g(B)$.

**性质 4.** $(g(f(A)),f(A))$和$(g(B),f(g(B)))$均为 $L(K)$上的概念.

概念格的 Hasse 图是按照父概念在上、子概念在下的原则,把父、子概念用线段连接而形成.图 1 所示为表 1 中形式背景诱导的概念格的 Hasse 图,图中的节点表示一个概念.记 $Parent(C)$为 $C$ 的所有父节点的集合,$Child(C)$为 $C$ 的所有子节点的集合.

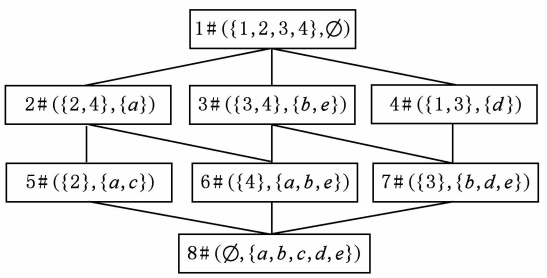


Fig. 1 Concept lattice of the formal context in table 1.

图 1 表 1 的形式背景所诱导的概念格

## 2 属性渐减的主要理论依据

### 2.1 属性渐减的基本定义

概念格的属性约简[14]有这样的理论基础:在形式背景$(G,M,I)$上,$\forall D\subseteq M$,记 $I_D=I\cap(G\times D)$,则$(G,D,I_D)$也是一个形式背景,记$(G,D,I_D)$所诱导的概念格为 $L(G,D,I_D)$.那么可以证明,$\forall(A,B')\in L(G,D,I_D)$, $\exists(A,B)\in L(G,M,I)$,满足 $B'\subseteq B$.该结论说明,形式背景的属性减少后,所诱导的概念格中的概念和原概念格中的概念具有一定的关系.例如,将表 1 中的属性 $a$ 减去后,某些节点会被删除掉或内涵被修改,而节点之间的边也随之作了相应的调整.如图 2 所示:

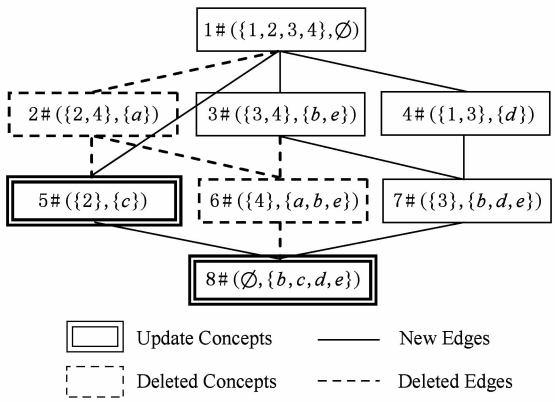


Fig. 2 Concept lattice of the formal context with deleting attribute 1.

图 2 表 1 的形式背景减去属性 1 后所诱导的概念格

为了方便研究这种变化的规律,我们作以下定义:

**定义 1.** 给定形式背景 $K=(G,M,I)$, $K'=(G,M',I')$,其中,$M'=M-\{m\}$, $m\in M$, $I'=I\cap G\times M'$.我们称 $m$ 为 $K$ 到 $K'$的消减属性,$K'$为 $K$ 的减属性 $m$ 背景,记为 $K'=K|_{-\{m\}}$, $L(K')$为 $L(K)$的减属性 $m$ 概念格,记为 $L(K')=L(K|_{-\{m\}})$.

形式背景 $K'$上的两个映射函数分别用 $f'$和 $g'$表示.显然,有如下关系成立:

**性质 5.** 当 $B\subseteq M'$时, $g'(B)=g(B)$.

**性质 6.** $f'(A)=f(A)-\{m\}$ 且当 $m\notin f(A)$时, $f'(A)=f(A)$.

为了研究概念格消减属性后改变的规律,我们根据原始概念格 $L(K)$中每个概念节点 $C$ 的内涵和消减属性 $m$ 之间的关系,定义它们的不同类型:

**定义 2.** 如果 $m\notin Int(C)$,则 $C$ 被称为是一个保留节点.形式背景 $K$ 上所有保留节点的集合记为 $RS_{-\{m\}}(K)$.

**定义 3.** 如果 $m\in Int(C)$且对于 $\forall C_1\in CS(K)$,都有 $Int(C_1)\neq Int(C)-\{m\}$;则 $C$ 被称为是一个更新节点.所有更新节点的集合记为 $MS_{-\{m\}}(K)$.

**定义 4.** 如果 $m\in Int(C)$且 $\exists C_1\in CS(K)$,使 $Int(C_1)=Int(C)-\{m\}$;则 $C$ 被称为是一个删除节点,$C_1$ 被称为节点 $C$ 的删除基.显然,删除基节点是

删除节点的父节点,一个删除节点只有一个删除基节点,且删除基节点为保留节点.所有删除节点的集合记为 $DS_{-\{m\}}(K)$.

由定义 2～4,显然有 $CS(K)=RS_{-\{m\}}(K)\cup MS_{-\{m\}}(K)\cup DS_{-\{m\}}(K)$.

## 2.2　概念格消减属性后节点的变化

本节通过研究 $RS_{-\{m\}}(K)$, $MS_{-\{m\}}(K)$, $DS_{-\{m\}}(K)$和 $CS(K|_{-\{m\}})$之间的关系,来研究概念格消减属性后的 $CS(K)$和 $CS(K|_{-\{m\}})$的关系.

**定理 1.** 设 $C\in CS(K)$,$K'=K|_{-\{m\}}$,如果 $C\in RS_{-\{m\}}(K)$,则必然有 $C\in CS(K')$.

证明. 设 $C=(A,B)$,由性质 4 可知,$(g'(B),f'(g'(B)))\in CS(K')$.因为 $C$ 为保留节点,故 $m\notin B\subseteq M'$,由性质 5 可知 $g'(B)=g(B)=A$.所以 $g'(B)=A$,下面证 $f'(g'(B))=B$.由性质 6 可知 $f'(g'(B))=f'(A)\subseteq f(A)=B$;由性质 2 可知 $B\subseteq f'(g'(B))$,所以 $f'(g'(B))=B$.所以$(g'(B),f'(g'(B)))=(A,B)=C\in CS(K')$.　　证毕.

由定理 1 可知,形式背景消减属性 $m$ 后,原概念格中的保留节点会保留到新的概念格中.即 $RS_{-\{m\}}(K)\subseteq CS(K|_{-\{m\}})$.

**引理 1.** 如果 $C=(A,B)\in MS_{-\{m\}}(K)$,则有 $g(B-\{m\})=g(B)$.

证明. 因为 $B-\{m\}\subseteq B$,根据性质 1 有 $g(B-\{m\})\supseteq g(B)$.再由性质 1 可得 $f(g(B-\{m\})\subseteq f(g(B))$.根据性质 2 有 $B-\{m\}\subseteq f(g(B-\{m\}))$,所以 $B-\{m\}\subseteq f(g(B-\{m\}))\subseteq f(g(B))=B$.则 $B-\{m\}=f(g(B-\{m\}))$和 $f(g(B-\{m\}))=f(g(B))$两个等式必然有且仅有一个成立.下面证 $B-\{m\}\neq f(g(B-\{m\}))$.根据性质 4 可知 $f(g(B-\{m\}))$为 $CS(K)$中某个概念的内涵.又因为 $C\in MS_{-\{m\}}(K)$,根据定义 3,所以格中不存在内涵为 $B-\{m\}$的节点.因此 $B-\{m\}\neq f(g(B-\{m\}))$,所以有 $f(g(B-\{m\}))=f(g(B))=B$.根据性质 3 又有 $g(B-\{m\})=g(f(g(B-\{m\})))=g(B)$.所以有 $g(B-\{m\})=g(B)$.　　证毕.

**定理 2.** 如果 $C=(A,B)\in MS_{-\{m\}}(K)$,$K'=K|_{-\{m\}}$,则必然有 $\exists C'=(A,B-\{m\})\in CS(K')$.

证明. 因为 $C=(A,B)\in MS_{-\{m\}}(K)$,故 $B=f(A)$,$m\in B$.设 $C'=(g'(f'(A)),f'(A))$,由性质 4 可知 $C'\in CS(K')$.由性质 6 可知 $f'(A)=f(A)-\{m\}=B-\{m\}$,由性质 5 和性质 6 及引理 1 可知 $g'(f'(A))=g'(f(A)-\{m\})=g(f(A)-\{m\})=g(B-\{m\})=g(B)=A$.所以有 $C'=(g'(f'(A)),f'(A))=(A,B-\{m\})\in CS(K')$.　　证毕.

由定理 2 可知,消减属性 $m$ 后,只需将原概念格的更新节点 $C$ 的内涵更新为 $Int(C)-\{m\}$就可以成为新概念格中的节点.

**引理 2.** 设 $K'=K|_{-\{m\}}$,对于 $\forall C'=(A',B')\in CS(K')$,都有 $\exists C=(A,B)\in CS(K)$,其中 $A'=A$,$B'=B-\{m\}$.

证明. 由于 $\forall C'=(A',B')\in CS(K')$,有 $f'(A')=B'$,$g'(B')=A'$.因为 $K'=K|_{-\{m\}}$,有 $A'\subseteq G$,$B'\subseteq M$.设 $A=g(f(A'))$,$B=f(A')$,根据性质 4 有$(A,B)\in CS(K)$.

先证 $B'=B-\{m\}$.由性质 6 可知 $f'(A')=f(A')-\{m\}\Rightarrow B'=B-\{m\}$.

再证 $A'=A$.根据性质 5、性质 6 和性质 1 有,$f(A')\supseteq f'(A')=B'\Rightarrow A=g(f(A'))\subseteq g(B')=g'(B')=A'$,即 $A\subseteq A'$.再由性质 2 可知 $A'\subseteq g(f(A'))=A$,所以有 $A'=A$.　　证毕.

**定理 3.** 设 $K'=K|_{-\{m\}}$,则集合 $RS_{-\{m\}}(K)\cup MS_{-\{m\}}(K)$中的元素和集合 $CS(K')$中的元素是一一对应的.

证明. 设映射 $u$: $RS_{-\{m\}}(K)\cup MS_{-\{m\}}(K)\rightarrow CS(K')$,映射条件为 $Ext(C)=Ext(C')$,其中 $C\in RS_{-\{m\}}(K)\cup MS_{-\{m\}}(K)$,$C'\in CS(K')$.

由定理 1 和定理 2 可知,在映射 $u$ 下,$C_1\neq C_2\Rightarrow u(C_1)\neq u(C_1)$,即集合 $RS_{-\{m\}}(K)$和 $MS_{-\{m\}}(K)$中的不同元素,在 $CS(K')$中有着不同的对应元素.所以映射 $u$ 是单射.

下面证明 $u$ 是满射.由引理 2 可知,映射 $u$ 是集合 $CS(K)$到 $CS(K')$的满射.由于 $CS(K)=RS_{-\{m\}}(K)\cup MS_{-\{m\}}(K)\cup DS_{-\{m\}}(K)$,因此只要证明在映射 $u$ 下,没有一个 $CS(K')$中的元素能在 $DS_{-\{m\}}(K)$中找到对应的元素即可.

对于节点 $C'=(A',B')\in CS(K')$,假设 $C'$ 在 $DS_{-\{m\}}(K)$中存在外延相同的对应的元素 $C_d=(A_d,B_d)$,则 $A_d=A'$.由引理 2 可知 $B'=B_d-\{m\}$.

由于 $C_d\in DS_{-\{m\}}(K)$,则由定义 3 可知 $m\in B_d$,设 $C=(A,B)$为 $C_d$ 的删除基节点,则有 $B=B_d-\{m\}$,$A\supset A_d$,$C\in CS(K')$.则保留节点 $C$ 和 $C'$ 内涵相同,外延不同,与格的定义矛盾,假设不成立.

所以映射 $u$ 是集合 $RS_{-\{m\}}(K)\cup MS_{-\{m\}}(K)$到 $CS(K')$的双射,集合 $RS_{-\{m\}}(K)\cup MS_{-\{m\}}(K)$和 $CS(K')$的元素是一一对应的.　　证毕.

由定理 3 可知,形式背景删除某个属性后诱导的概念格不会产生新的概念(外延没有变化),其中

的概念节点可以由原概念格中的保留节点和更新节点映射过来.因此只需要针对 $CS(K)$ 中的所有概念逐个判断节点的类型再进行相应的处理即可:根据定理 1,如果是保留节点则不作任何改变;根据定理 2,如果是更新节点,则将节点的内涵减去属性 $m$ 进行更新;而删除节点则直接从 $CS(K)$ 中删除掉即可.

为了能够在产生 $CS(K|_{-\{m\}})$ 集的同时建立相应的 Hasse 图,还需要对节点之间的边进行处理.下面我们进一步研究从 $L(K)$ 到 $L(K|_{-\{m\}})$ 的边的变化.

### 2.3 概念格减属性后边的变化

Hasse 图中的边仅仅存在于父节点和子节点之间,因此可以根据父子节点的变化来确定边的变化.根据节点的分类,仅需考虑 9 种情况,如表 2 所示:

**Table 2 Changes of All Types of Edges between Parent-Child**

**表 2 各类型父子节点间边的变化**

| Index | Parent | Child | Changes of Edges |
|---|---|---|---|
| 1 | remained | remained | no changes |
| 2 | remained | modified | no changes |
| 3 | remained | deleted | remove |
| 4 | modified | remained | impossible |
| 5 | modified | modified | impossible |
| 6 | modified | deleted | impossible |
| 7 | deleted | remained | impossible |
| 8 | deleted | modified | remove |
| 9 | deleted | deleted | remove |

**定理 4.** 如果 $C\in RS_{-\{m\}}(K)$,则 $\forall C'\in Parent(C)$,都有 $C'\in RS_{-\{m\}}(K)$.

证明. 因为 $C'\in Parent(C)$,故 $Int(C')\subseteq Int(C)$.又因为 $C\in RS_{-\{m\}}(K)$,故 $m\notin Int(C)$.所以 $m\notin Int(C')$,由定义 2 可知 $C'\in RS_{-\{m\}}(K)$. 证毕.

定理 4 告诉我们,保留节点的父节点必然还是保留节点.因此表 2 中序号 4,7 所示情况不存在.

**定理 5.** 设 $K'=K|_{-\{m\}}$,$C_1,C_2\in RS_{-\{m\}}(K)\cup MS_{-\{m\}}(K)$,如果在 $L(K)$ 中,$C_1\prec C_2$,则在 $L(K')$ 中也有 $C_1\prec C_2$.

证明. 设 $C_1=(A_1,g(A_1))$,$C_2=(A_2,g(A_2))$,在 $L(K)$ 中,因为 $C_1\prec C_2$,所以 $A_1\subseteq A_2$ 且不存在节点的外延 $A_3$ 使得 $A_1\subseteq A_3\subseteq A_2$.由定理 3 可知,$CS(K')$ 中没有新增外延.所以在 $L(K')$ 中也不存在节点 $C_3=(A_3,g(A_3))$,使得 $A_1\subseteq A_3\subseteq A_2$.因此在 $L(K')$ 中,$C_1\prec C_2$. 证毕.

由定理 5 可知,如果父子节点都不是删除节点,则父子节点间的边会保留到新的概念格中.因此表 2 中序号 1,2,5 所示情况边不发生变化.

**定理 6.** 设 $C_d=(A_d,B_d)\in DS_{-\{m\}}(K)$,如果 $C=(A,B)$,使得 $C_d\prec C$ 成立,则 $C$ 必为下列两种情形之一:

1) $C\in RS_{-\{m\}}(K)$,且为 $C_d$ 的删除基节点;

2) $C\in DS_{-\{m\}}(K)$.

证明. 设 $C_1=(A_1,B_1)$ 为 $C_d$ 的删除基节点,则 $C_d\prec C_1$,$B_1=B_d-\{m\}$.

1)根据定义 4 可知 $C_1\in RS_{-\{m\}}(K)$.因此仅需证明当 $C\in RS_{-\{m\}}(K)$ 时,若 $C\neq C_1$,则 $C_d\prec C$ 不成立即可.因为 $C_d\prec C$ 且 $C\in RS_{-\{m\}}(K)$,则有 $B\subseteq B_d$,且 $m\notin B$.所以 $B\subseteq B_d-\{m\}=B_1\subseteq B_d$.若 $C\neq C_1$,有 $C_d\prec C_1\leqslant C$,故 $C_d\prec C$ 不成立.

2)由 $CS(K)=RS_{-\{m\}}(K)\cup MS_{-\{m\}}(K)\cup DS_{-\{m\}}(K)$ 及 1),仅需证明 $C\notin MS_{-\{m\}}(K)$ 即可.假设 $C\in MS_{-\{m\}}(K)$.由 $C_d\prec C$,$C_d\prec C_1$,可知在 $L(K)$ 中,$C$ 和 $C_1$ 不存在$\leqslant$关系(由格的偏序定义可知,等价于 $A$ 和 $A_1$ 不存在$\subseteq$关系).再由 $C_d\prec C$,有 $B\subseteq B_d\Rightarrow B-\{m\}\subseteq B_d-\{m\}=B_1$,因此节点更新后,在 $L(K')$ 中有 $C\leqslant C_1$(等价于 $A\subseteq A_1$).由定理 3 可知,$L(K')$ 和 $L(K)$ 中的概念外延部分并不改变,矛盾.假设不成立. 证毕.

由定理 6 可知,删除节点的父节点只有两种情形:唯一地保留节点(删除基节点)或删除节点,即表 2 中序号 6 所示情况不存在.

对于表 2 中序号 3,8,9 所示情况,可以发现,删除节点存在于更新节点和保留节点之间.当删除节点被移除时,其父节点和子节点的边也相应地被删除,但此时需要考虑删除节点的父节点(仅需考虑为删除基节点时的情况)和子节点之间有没有直接前驱后继关系:当删除节点的子节点的所有父节点都和删除基节点不存在$\leqslant$关系时,删除基节点和删除节点的子节点需要新增边.定理 7 可以使我们在判断删除节点的子节点的所有父节点是否和删除基节点有$\leqslant$关系时,减少不必要的比较.

**定理 7.** 设 $C_d=(A_d,B_d)\in DS_{-\{m\}}(K)$,$C_1=(A_1,B_1)$ 为 $C_d$ 的删除基节点,若存在 $C_b=(A_b,B_b)\in MS_{-\{m\}}(K)\cup DS_{-\{m\}}(K)$ 且 $C_b\in Parent(Child(C_d))$,则 $C_b\leqslant C_1$ 不成立.

证明. 用反证法.设 $C_b\leqslant C_1$ 成立,则有 $B_1\subseteq B_b$.因为 $C_b\in MS_{-\{m\}}(K)\cup DS_{-\{m\}}(K)$,有 $m\in B_b$.因为 $C_1$ 为 $C_d$ 的删除基节点,有 $B_d=B_1\cup\{m\}$,则

$B_1 \subseteq B_b \Rightarrow B_1 \cup \{m\} \subseteq B_b \cup \{m\} = B_b \Rightarrow B_d \subseteq B_b$,所以有 $C_b \leqslant C_d$,与 $C_b \in Parent(Child(C_d))$矛盾.

证毕.

定理7告诉我们,删除节点 $C_d$ 的某个子节点的父节点 $C_b$ 如果是更新节点或删除节点,则 $C_b$ 和删除基节点 $C_1$ 间不存在偏序关系$\leqslant$.

## 3　概念格的属性渐减算法

由遍历方式的不同,属性渐减时可以对格按自底向上和自顶向下两种方式进行调整.

### 3.1　自底向上算法的主要思想

自底向上从 $L(K)$更新为 $L(K|_{-\{m\}})$算法的主要思想是:从格的最大下界节点 inf($L(K)$)出发,按照节点间的父子关系,自底向上分别判断每个节点是否是更新节点和删除节点,并进行相应的节点更新、节点删除及边的调整,如算法1所示.

**算法1.** BUAD(bottom-up attribute decremental)算法.

输入:原始概念格 $L(K)$;删除属性 $m$;

输出:删除属性 $m$ 后的格 $L(K|_{-\{m\}})$.

BEGIN

① $Vset := \varnothing$;$CSet :=\{\inf(L(K))\}$;

② WHILE $CSet \neq \varnothing$ DO

③　移出 $CSet[0]$到 $C$;$Int(C) := Int(C) - \{m\}$;

④　$C_{db} := \varnothing$;

⑤　FOR each $C_p \in Parent(C)$

⑥　　IF ($C_{db} = \varnothing$) AND ($Int(C) = Int(C_p)$) THEN

⑦　　$C_{db} := C_p$;

⑧　　IF $C = \inf(L(K))$ THEN $\inf(L(K)) := C_p$;END IF

⑨　　ELSE

⑩　　IF ($C_p$ 没有被置 $vs$ 标记) AND (($C_{db} \neq \varnothing$) OR ($m \in Int(C_p)$)) THEN

⑪　　添加 $C_p$ 至 $CSet$ 尾;给 $C_p$ 置 $cs$ 标记;

⑫　　END IF

⑬　　添加 $C_p$ 至 $VSet$ 尾;给 $C_p$ 置 $vs$ 标记;

⑭　　END IF

⑮　END FOR

⑯　IF $C_{db} \neq \varnothing$ THEN

⑰　　FOR each $C_c \in Child(C)$

⑱　　$needEage$:= True;

⑲　　FOR each $C_{cp} \in Parent(C_c)$

⑳　　IF ($C_{cp}$ 没有被置 $cs$ 标记) AND ($C_{cp} \neq C$) AND ($Ext(C_{cp}) \subseteq Ext(C_{db})$) THEN

㉑　　$needEage$:= False;BREAK;

㉒　　END IF

㉓　　END FOR

㉔　　IF $needEage$ THEN 新增边 $C_{db} \rightarrow C_c$;END IF

㉕　END FOR

㉖　FOR each $C_p \in Parent(C)$删除边 $C_p \rightarrow C$;END FOR

㉗　FOR each $C_c \in Child(C)$删除边 $C \rightarrow C_c$;END FOR

㉘　从 $L(K)$中移除并销毁节点 $C$;

㉙　ELSE

㉚　取消 $C$ 的 $cs$ 标记;

㉛　END IF

㉜ END WHILE

㉝ 取消集合 $VSet$ 中节点的 $vs$ 标记;

END

在算法1中,$CSet$ 用于保存待处理的删除和更新节点的集合,$VSet$ 为保持访问过的节点集合,$C_{db}$ 用于记录当前节点 $C$ 的删除基节点,$needEage$ 用于标志是否在删除基节点和删除节点的子节点间新增边.为了加快节点的判断速度,每个节点有2个标志域:$vs$ 用于标志该节点是否已判断过,$cs$ 用于标志该节点是否为删除或更新节点.

算法1第①行对 $CSet$ 和 $VSet$ 进行初始化,将 inf($L(K)$)放入 $CSet$,②～㉜行是算法的主循环.算法在③～⑮行对节点类型的判断和处理,并将更新、删除节点加入 $CSet$ 中.其中,⑥～⑨行判断待处理节点 $C$ 是否为删除节点,并记录删除节点的删除基节点.由于删除基节点只有一个,因此当找到删除基节点后($C_{db} \neq \varnothing$)便不再判断.⑧行是判断是否需要修改 inf($L(K)$),如果 inf($L(K)$)为删除节点,则需要将 inf($L(K)$)的删除基节点作为格 $L(K|_{-\{m\}})$的 inf($L(K|_{-\{m\}})$)节点.⑩～⑫行将包含消减属性 $m$ 的节点 $C$ 的父节点放入 $CSet$ 中.在⑩行的判断语句中,第1个判断条件($C_p$ 没有被置 $vs$ 标记)是为了避免重复判断,第2个判断条件($C_{db} \neq \varnothing$)是因为此时 $C$ 是删除节点,其父节点必然是删除节点(表2中的另一种情况,删除节点的父节点是保留

节点——删除基节点的情况已在⑥～⑨行处理). 由表 2 可知,保留节点的父节点不可能是删除或更新节点,因此,除 inf($L(K)$)外,$CSet$ 中的节点均为删除节点和更新节点. 同时概念格 $L(K)$中的删除节点和更新节点均被放入 $CSet$ 中.

算法在⑯～㉙行对删除节点的边进行调整,并将删除节点从格中删除. 其中,⑳行的判断语句中,判断条件($C_{cp}$没有被置 $cs$ 标记)是利用定理 7 的结论减少判断次数. 算法在㉚行和㉝行将保留节点、更新节点的 $vs$ 和 $cs$ 标志复位,以便继续进行其他属性的删除.

算法的时间复杂度主要取决于主 WHILE 循环和⑰～㉕行的双层嵌套 FOR 循环的执行次数. 算法的时间复杂度可以看作两者的乘积. 假设形式背景 $K$ 上的对象数为 $\|G\|$,属性数为 $\|M\|$,概念格 $L(K)$的节点数量为 $\|L\|$,某节点 $C=(A,B)$的对象数为 $\|A\|$,属性数为 $\|B\|$.

首先分析主 WHILE 循环的执行次数. 根据循环终止条件,WHILE 循环依赖于所有被放入 $CSet$ 集合的节点总个数. 由算法 1 的伪代码可知,被放入 $CSet$ 集合的节点是所有的删除节点和更新节点,且仅放入一次. 而算法⑯～㉛行是对删除节点进行处理. 根据定义 4 可知,概念格的保留节点的数量必然不少于删除节点. 因此,概念格 $L(K)$中的删除节点最多有 $\|L\|/2$ 个.

下面分析⑰～㉕行的双层嵌套 FOR 循环的执行次数. 该循环主要判断是否应在删除节点的子节点和删除基节点之间增加边,需要遍历删除节点的所有子节点的所有父节点. 而算法的全局最坏时间复杂度的情况是,对于给定的 $\|G\|$ 和 $\|M\|$,概念格中的节点数目达到最多. 此时,$M$ 的任何一个子集都能成为一个概念的内涵,或者 $G$ 的任何一个子集都能成为一个概念的外延. 也即,任何一对父子概的内涵仅相差一个属性,或者外延仅相差一个对象. 因此,对格中的每一个概念 $C$ 而言,最多有 $\|B\|$ 个父概念,$\|A\|$ 个子概念. 令 $\|B\|$ 和 $\|A\|$ 分别取上界 $\|G\|$ 和 $\|M\|$,则在全局最坏情况下,双层嵌套 FOR 循环最大循环次数不超过 $\|G\|\cdot\|M\|$.

综上所述,WHILE 循环遍历删除节点不超过 $\|L\|/2$ 次,⑰～㉕行的双层嵌套循环遍历删除节点的所有子节点的所有父节点不超过 $\|G\|\cdot\|M\|$ 次. 故本文的渐减式算法 1 的最坏时间复杂度为 $O(\|L\|\cdot\|G\|\cdot\|M\|)$,优于文献[10]渐增式算法的时间复杂度 $O(\|L\|\cdot\|G\|^2\cdot\|M\|)$.

### 3.2 自顶向下算法的主要思想

观察表 2 可以发现:更新节点的子节点必然为更新节点,删除节点的子节点必然为删除或更新节点. 则如果找到某个节点含有 $m$ 属性,则该节点所有的子孙节点都含有 $m$ 属性. 如果自顶向下判断,则几乎不用去考虑保留节点(仅仅在需要判断是否为删除节点时才需要),这样会减少判断次数.

自顶向下算法的主要思想是:从概念格的最小上界出发,找到最顶层的含有消减属性 $m$ 的节点;然后遍历含有消减属性 $m$ 的节点的所有子孙节点,逐个判断是更新节点还是删除节点;对更新节点,删除内涵中的 $m$ 属性;对删除节点,判断删除基节点和子节点之间是否需要新增边,将删除节点和父子节点间的边移除,销毁删除节点. 算法的关键是寻找最顶层的含有消减属性 $m$ 的节点.

**定义 5.** 形式背景 $K=(G,M,I)$中,$m\in M$,如果 $\|Ext(C)\|=\|g(\{m\})\|$,则称 $C$ 为属性 $m$ 的根节点.

**定理 8.** 设形式背景 $K=(G,M,I)$,$m\in M$,节点 $C\in CS(K)$,如果 $m\in Int(C)$,则 $\max(\|Ext(C)\|)=\|g(\{m\})\|$.

证明. 由性质 4 可知,$(g(\{m\}),f(g(\{m\})))$为 $L(K)$的概念,所以必然存在节点 $C\in CS(K)$,使得 $\|Ext(C)\|=\|g(\{m\})\|$. 下面证 $\|Ext(C)\|>\|g(\{m\})\|$是不可能的. 设 $C'=(A',B')\in CS(K)$且 $m\in B'$,$\|A'\|>\|g(\{m\})\|$,$g(B')=A'$. 根据性质 1 有 $\{m\}\subseteq B'\Rightarrow g(\{m\})\supseteq g(B')=A'$,与 $\|A'\|>\|g(\{m\})\|$ 矛盾. 所以不存在节点 $C'\in CS(K)$且 $m\in B'$,使得 $\|Int(C')\|>\|f(\{m\})\|$.

证毕.

由定理 8 可知,最顶层的含有消减属性 $m$ 的节点必然是外延大小为 $\|g(\{m\})\|$ 的节点,即属性 $m$ 的根节点. 因此我们可以从概念格的最小上界出发,先寻找内涵大小为 $\|g(\{m\})\|$ 的节点,然后判断该节点是否为含消减属性 $m$ 的节点.

**推论 1.** 设形式背景 $K=(G,M,I)$,$m\in M$,$C$ 为属性 $m$ 的根节点,如果不存在属性 $m'$,$m\neq m'$,使得 $g(\{m'\})\supseteq g(\{m\})$成立,则 $C\in Child(\sup(L(K)))$或者 $C=\sup(L(K))$.

证明. 由定理 8 可知,$\|Ext(C)\|=\|g(\{m\})\|$. 由性质 4 可知,$(g(\{m'\}),f(g(\{m'\})))$为 $L(K)$的概念. 由于不存在 $g(\{m'\})\supseteq g(\{m\})$,所以不存在概念 $C'$ 满足 $C\leqslant C'$,所以当 $C\neq\sup(L(K))$时有 $C\in Child(\sup(L(K)))$.

证毕.

由推论 1 可知,对于具有消减属性 $m$ 的所有对象的集合 $g(\{m\})$,如果形式背景中不存在其他属性 $m'$,使得 $g(\{m'\})\supseteq g(\{m\})$,则属性 $m$ 的根节点必然是 $\sup(L(K))$节点或 $\sup(L(K))$的子节点.

自底向上属性渐减算法如算法 2 所示.在算法 2 中,为了寻找属性 $m$ 的根节点,需要从 $\sup(L(K))$ 节点出发,往下检查格节点.*DSet* 和 *VSet* 是用于保存这些节点的集合.*CSet* 用于保存待处理的删除和更新节点的集合.和算法 1 相比,每个节点增加一个标志域 *ms*,用于标志该节点是否为更新节点.

**算法 2.** TDAD(top-down attribute decremental)算法.

输入:原始概念格 $L(K)$;删除属性 $m$;对象集 $g(\{m\})$;

输出:删除属性 $m$ 后的格 $L(K|_{-\{m\}})$.

```
BEGIN
① CSet :=∅;VSet :=∅;Dset :={sup(L(K))};
② WHILE Dset≠∅ DO
③   移出 Dset [Length(Dset)-1]到 C;
④   FOR each C_c∈Child(C)
⑤     IF (C_c 没有被置 vs 标记) AND
        (‖Ext(C_c)‖>‖g({m})‖) THEN
⑥       添加 C_c 至 Dset 尾和 Vset 尾;给 C_c
          置 vs 标记;
⑦     ELSE
⑧       IF (‖Ext(C_c)‖=‖g({m})‖)
          AND (m∈Int(C_c)) THEN
⑨         CSet :={C_c};给 C_c 置 cs 标记;
⑩         BREAK;
⑪       END IF
⑫     END IF
⑬   END FOR
⑭   IF CSet≠∅ THEN BREAK;END IF
⑮ END WHILE
⑯ 取消集合 VSet 中节点的 vs 标记;
⑰ WHILE CSet≠∅ DO
⑱   移出 Cset [0]到 C;Int(C)←Int(C)-
      {m};
⑲   C_db:=∅;
⑳   IF C 没有被置 ms 标记 THEN
㉑     FOR each C_p∈Parent(C)
㉒       IF (C_p 没有被置 cs 标记) AND
          (Int(C)=Int(C_p)) THEN
㉓         C_db:= C_p;BREAK;
㉔       END IF
㉕     END FOR
㉖   END IF
㉗   FOR each C_c∈Child(C)
㉘     IF C_c 没有被置 cs 标记 THEN
㉙       添加 C_c至 CSet 尾;给 C_c 置 cs 标记;
㉚     END IF
㉛     IF C_db=∅ THEN 给 C_c 置 ms 标记;
        END IF
㉜   END FOR
㉝   取消节点 C 的 ms 标记;
  ⋮
  算法 1 的⑯～㉛行
  ⋮
㉞ END WHILE
END
```

算法 2 的第①行对 *DSet*,*VSet* 和 *CSet* 进行初始化,将 $\sup(L(K))$放入 *DSet* 集合.②～⑮行的循环用于寻找属性 $m$ 的根节点.循环从 $\sup(L(K))$出发,寻找外延集的对象数和 $\|g(\{m\})\|$ 相等的节点,判断其内涵是否含有属性 $m$,如找到则退出循环.

⑰～㉞行的 WHILE 循环是对所有的删除或更新节点进行处理,并调整节点间的边.循环不断将 *CSet* 的节点的父节点加入到 *CSet* 中.由于 *CSet* 中的初始节点是属性 $m$ 的根节点,因此可知所有的删除或更新节点都将被加入到 *CSet* 中.在⑳行中,如果 $C$ 没有被置 *ms* 标记,因此可能是删除节点,则进一步在㉑～㉕行判断,若为删除节点记录其删除基节点,否则为更新节点.由表 2 可知,更新节点子节点必然为更新节点,故在㉛行中将子父节点标记为 *ms*.

⑯和㉝行恢复 *VSet* 中节点的 *vs* 和 *ms* 标记,以便继续对其他属性删除时使用.

如果当前节点为删除节点,需要进一步调整节点间的父子关系并将其从格中删除.这个过程与算法 1 中行⑯～㉛相同.

算法 2 的最坏时间复杂度也主要是体现在处理删除节点时对边进行调整的双层嵌套循环上,因此和算法 1 相同.

在实际应用中,形式背景的属性可能会不断变化,需要连续消减属性.此时可以预先将格的节点按照外延集中对象数的多少预先存入算法 2 的 *DSet* 之中,这样每次只需要根据对象集 $g(\{m\})$的元素个数,直接在 $DSet[\|g(\{m\})\|]$中寻找属性 $m$ 的根节点即可.

在某些特殊的应用中,可能只有概念格 $L(K)$ 而不知道其对应的形式背景 $K$. 因此当需要从概念格中消减属性 $m$ 时,并不知道属性 $m$ 在形式背景 $K$ 上的对象集 $g(\{m\})$. 这种情况下可以考虑采用广度优先按照节点的内涵数逐层遍历的方法来寻找属性 $m$ 的根节点.

### 3.3 渐增与渐减算法的节点类型关系分析

本文所提出的概念格属性渐减算法和已有的渐增式概念格构造算法都是渐进式构造算法. 两类算法都是在原有概念格基础上进行修改来得到新的概念格,是互逆的算法. 因此在节点类型的划分上具有很强的对应关系. 下面以最经典的 Godin 算法作为渐增式算法的代表,来分析两者的对应关系.

通过分析可知,本文所定义节点类型与 Godin 算法中的节点类型的对应关系为: 两者的保留节点(有文献也将 Godin 的保留节点译作不变节点)不需要经过任何改变就能留在新概念格中;更新节点均需要经过修改才能成为新概念格中的节点,其中,本文的更新节点的内涵需要删除待消减属性,Godin 算法中更新节点的外延需要加入待新增对象;而本文中的删除节点与 Godin 的新生节点处理起来正好相反,前者是从原概念格中删除,而后者是加入到原概念格中;本文的删除基节点与 Godin 算法的产生子节点都是保留节点,其中,本文的删除基节点是删除节点的父节点,而 Godin 算法的产生子节点是新生节点的子节点.

显然,节点类型的引入对算法的运行将会产生重要的影响. 两类算法都是通过对节点的类型划分来减少搜索空间,加快运算速度. 具体地,在算法的实现方面,由于 Godin 算法是根据原概念格中内涵的升序来归类格节点,可以避免对所有节点的比较,在一定程度上提高了访问速度;而本文算法则根据不同类型节点的前驱-后继关系来避免对所有节点的比较,进一步缩减了搜索空间,因此具有更快的运算速度.

## 4 实验与分析

为了验证算法的有效性,我们使用 Delphi7 编程实现了 Godin 算法[9]、AddIntent 算法[10]、FIA_A[12]算法、In-Close 算法[13]以及本文提出的算法 1(BUAD)和算法 2(TDAD). Godin 算法是最经典的概念格构造算法,AddIntent 算法是最快的基于对象的渐进式算法之一,FIA_A 算法是渐增属性的算法,In-Close 算法是近年出现最快的概念格构造算法之一. 因此选取这几种算法与本文所述的两种算法进行实验性能比较具有代表性的意义. 我们在 CPU 主频为 2.30 GB、内存为 3 GB、操作系统为 Windows XP 的计算机上进行了两个实验. 为了使实验结果更具有统计意义,对于每个实验所用到的形式背景及其诱导的概念格都随机产生 5 组数据. 实验中获得的算法所耗费的时间是在这 5 组数据上的平均值.

**实验 1.** 由形式背景重新构造概念格和概念格属性渐减两种方式的性能对比.

在实验所用到的每组数据中,形式背景的对象个数固定为 100,对象属性间存在关系的概率为 0.30,属性个数从 10 开始,每次递增 10 个,直至 150 为止,共包含 15 个形式背景. 首先分别使用 BUAD 和 TDAD 算法将所有概念格的第 1 个属性减去并记录算法时间. 再将所有形式背景的第 1 个属性减去,然后分别采用 Godin 算法、AddIntent 算法、In-Close 算法和 FIA_A 算法构造概念格并记录构造时间. 实验结果如图 3~5 所示:

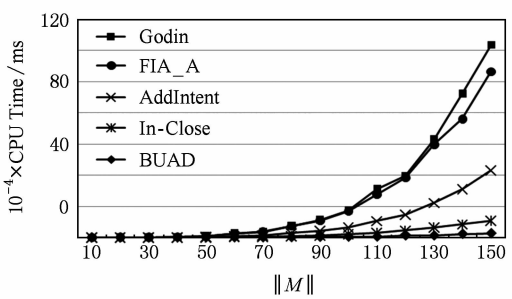


Fig. 3 Performances of Godin, FIA_A, AddIntent, In-Close and BUAD where $\|G\|=100$.

图 3 $\|G\|=100$ 时 Godin, FIA_A, AddIntent, In-Close 与 BUAD 算法的性能比较

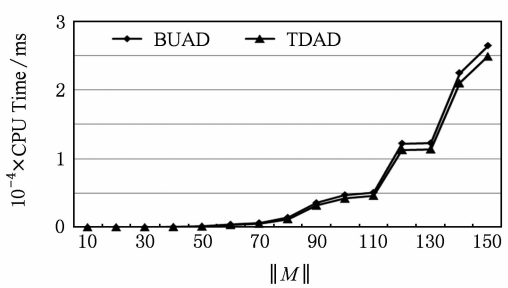


Fig. 4 Performances of TDAD and BUAD where $\|G\|=100$.

图 4 $\|G\|=100$ 时 TDAD 和 BUAD 算法的时间性能

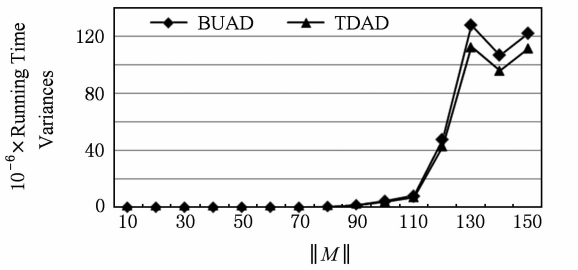


Fig. 5　Running time variances between TDAD and BUAD where $\|G\|=100$.

图 5　$\|G\|=100$ 时 TDAD 和 BUAD 算法在 5 组实验数据上的方差

从图 3 可以看出,随着属性个数的增加,直接采用 BUAD 算法在原有概念格中进行属性消减,要比在形式背景中消减属性之后再用 Godin,AddIntent,In-Close 或 FIA_A 算法节省大量的时间.

由于实验中 BUAD 和 TDAD 算法所消耗的时间差距并不明显,因此将两者的实验数据放在图 4 中单独对比.在图 4 中随着属性个数的增加,两种算法进行属性消减所需的时间逐渐增大,且 TDAD 算法具有略微的时间优势.图 5 中显示出了两种算法在 5 组实验数据上所耗费时间的方差.可以看出随着属性个数的增加,两种算法所耗费时间的方差逐渐增大,且 TDAD 算法的方差略小.说明对同规格的概念格进行属性消减时,算法取得的时间性能随形式背景属性规模增大而具有更大的随机性.

**实验 2.** 属性数较多时 BUAD 和 TDAD 算法的性能对比.

在实验的每组数据中,形式背景的对象个数固定为 20,对象属性间存在关系的概率为 0.30,属性个数从 100 开始,每次递增 50 个,直至 1 000 为止,共包含 19 个形式背景.分别使用 BUAD 和 TDAD 算法将所有概念格的第 1 个属性减去并记录算法时间.对于 TDAD 算法特别记录其寻找消减属性的根节点时的时间.实验结果如图 6～8 所示.

从图 6 可以看出,随着形式背景属性个数的增加,两种算法整体上所耗费时间增大,但是 TDAD 曲线波动幅度较大.说明 TDAD 算法的时间性能随机性比较大,可以在图 7 中得到验证.图 7 中 TDAD 算法在 5 组实验数据上的方差比 BUAD 要大,而且随着属性个数的增大,方差显著性的增加.这是由于消减属性的根节点位于格中的层次随机性很大,TDAD 算法在寻找该节点时耗费的时间比较多,如图 8 所示.

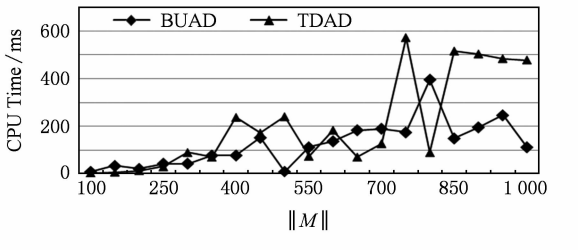


Fig. 6　Performances of TDAD and BUAD where $\|G\|=20$.

图 6　$\|G\|=20$ 且属性个数较多时 TDAD 和 BUAD 算法的时间性能

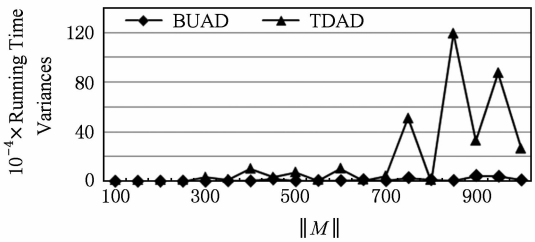


Fig. 7　Running time variances between TDAD and BUAD where $\|G\|=20$.

图 7　$\|G\|=20$ 且属性个数较多时 TDAD 和 BUAD 算法在 5 组实验数据上的方差

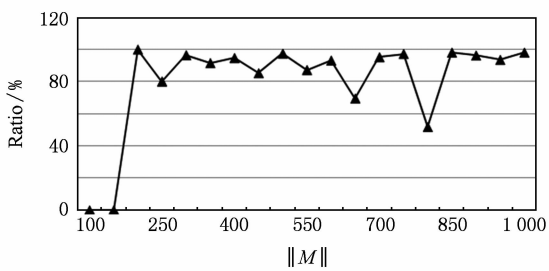


Fig. 8　The ratio between time for finding the root node of deleting attribute and the total completion time in TDAD algorithm.

图 8　TDAD 算法用于寻找属性的根节点所用时间占算法总时间的比值

对比图 4～7 可以发现,对于实验 1 和实验 2 中不同规格的实验数据,TDAD 和 BUAD 算法取得了相反的实验性能.这是由概念格中的消减属性的根节点所处的层次造成的.由推论 1 可知,在形式背景 $(G,M,I)$ 中,若不存在其他属性 $m'$,使得 $g(\{m'\})\supseteq g(\{m\})$,则属性 $m$ 的根节点必然是 $\sup(L(K))$ 节点或 $\sup(L(K))$ 的子节点.很容易进一步推知,若 $g(\{m'\})\supseteq g(\{m\})$ 成立,属性 $m$ 的根节点必然在概念格中隐藏较深,此时 TDAD 算法需要耗费大量

的时间去寻找属性 $m$ 的根节点.

在形式背景 $K=(G,M,I)$ 中,易知具有属性 $m$ 的对象集 $g(\{m\})$ 被具有其他属性 $m'$ 的对象集 $g(\{m'\})$ 包含的情况有 $2^{\|G\|-\|g(\{m\})\|}$ 种. 又因为对象集 $G$ 的子集有 $2^{\|G\|}$ 个,其中有 $\|M\|$ 个在集合 $\{g(\{m'\})\mid \forall m'\in M\}$ 中,即在形式背景中和属性间存在 $g$ 映射关系. 因此对象集 $G$ 的子集可能是 $g(\{m'\})$ 的概率为 $\|M\|/2^{\|G\|}$,所以属性 $m$ 具有对象 $g(\{m\})$ 被其他属性具有对象 $g(\{m'\})$ 包含的概率为 $\|M\|/2^{\|g(\{m\})\|}$. 又因为 $\|g(\{m\})\|$ 的取值范围为 $[0,\|G\|]$,所以当 $\|M\|$ 越大,而 $\|G\|$ 和 $\|g(\{m\})\|$ 越小时,属性 $m$ 的根节点隐藏的越深,TDAD 算法需要耗费更多的时间去寻找属性 $m$ 的根节点.

在实际应用中,大多数情况下(如数据挖掘、信息检索等情况)形式背景对象个数要远远大于属性个数(即实验 1 中的形式背景规格),因此通常情况下 TDAD 算法会取得比 BUAD 算法较好的时间性能. 如果属性数远大于对象数(如实验 2 中的特例情况下的形式背景规格)时,采用 BUAD 算法会产生较好的时间性能.

## 5 结论及进一步的工作

本文揭示了概念格属性消减后概念集合和 Hasse 图的变化规律,在此基础上给出了自底向上和自顶向下两种算法,实现了对概念格属性的渐进式消减. 本文的算法实现了较低的时间复杂度 $O(\|L\|\cdot\|G\|\cdot\|M\|)$. 实验与分析表明,本文提出的概念格属性渐减算法是有效的. 在实际的应用领域中,如果因形式背景的属性发生变化需要删除某些属性时,利用本文提出的算法从原有概念格中将属性直接消减掉,比重新从形式背景构造概念格所需要的时间要少得多.

在实际应用中,有可能会同时删除两个以上的属性,尽管这可以连续使用多次概念格属性渐减算法做到,但是研究同时删除 $n$ 个以上属性的概念格的属性渐减算法,无疑也是一件很有意义的事情. 在实际应用中,还可能会出现修改一个对象和一个属性间的关系这种情况,这可以使用属性渐减算法和 FIA_A 算法相结合的方式,先用属性渐减算法将该属性删除,然后再利用 FIA_A 算法将调整过对象-属性关系的属性重新加入到概念格中. 对能够一次做完对象和属性关系调整的算法的研究,也是需要进一步研究的工作.

## 参 考 文 献


[1] Wille R. Restructuring lattice theory: An approach based on hierarchies of concept [C] //Rival I ed. Ordered Sets. Dordecht- Boston: Reidel, 1982: 445-470

[2] Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations [M]. Berlin: Springer, 1999

[3] Peng Qiangqiang, Du Yajun, Hai Yufeng, et al. Topic-specific crawling on the Web with concept context graph based on FCA [C] //Proc of Int Conf on Management and Service Science. Piscataway, NJ: IEEE, 2009: 1-4

[4] Shyng J Y, Shieh H M, Tzeng G H. An integration method combining rough set theory with formal concept analysis for personal investment portfolios [J]. Knowledge-Based System, 2010, 23(6): 586-597

[5] Snasel V, Horak Z, Kocibova J, et al. A analyzing social networks using FCA: Complexity aspects [C] //Proc of 2009 IEEE/WIC/ACM Int Joint Conf on Web Intelligence (WI) And Intelligent Agent Technologies (IAT). Piscataway, NJ: IEEE, 2009: 38-41

[6] Li Yang, Yang Xu, Decision making with uncertainty information based on lattice-valued fuzzy concept lattice [J]. International Journal of General Systems, 2010, 39(3): 235-253

[7] Kuznetsov S O, Obiedkov S A. Comparing performance of algorithms for generating concept lattices [J]. Journal of Experimental & Theoretical Artificial Intelligence, 2002, 14(2): 189-216

[8] Nourine L, Raynaud O. A fast algorithm for building lattices [J]. Information Process Letter. 1999, 71(5/6): 199-204

[9] Godin R, Missaoui T, Alaui H. An incremental concept formation algorithm based on Galois (concept) lattices [J]. Computational Intelligence, 1995, 11(2): 246-267

[10] van der Merwe D, Obiedkov S, Kourie D. AddIntent: A new incremental algorithm for constructing concept lattices [G] // LNCS 2961: Proc of the 2nd Int Conf on Formal Concept Analysis. Berlin: Springer, 2004: 372-385

[11] Liu Zongtian, Qiang Yu, Zhou Wen, et al. A fuzzy concept lattice model and its incremental construction algorithm [J]. Chinese Journal of Computers, 2007, 30(2): 184-188 (in Chinese)
(刘宗田, 强宇, 周文, 等. 一种模糊概念格模型及其渐进式构造算法[J]. 计算机学报, 2007, 30(2): 184-188)

[12] Qu Liping, Liu Daxin, Yang Jing, et al. Attribute-based fast incremental algorithm for building concept lattice [J]. Journal of Computer Research and Development, 2007, 44(Suppl): 251-256 (in Chinese)

(曲立平，刘大昕，杨静，等. 基于属性的概念格快速渐进式构造算法[J]. 计算机研究与发展，2007，44(增刊)：251-256)

[13] Andrews S. In-Close，a fast algorithm for computing formal concepts[C] //Supplementary Proc of the 17th Int Conf on Conceptual Structures (ICCS). Moscow，Russia：CEUR-WS，2009：1-14

[14] Zhang Wenxiu，Wei Ling，Qi Jianjun. The theory and approach of attribute reduction in concept lattice[J]. Science in China Series E：Informational Science，2005，35(6)：628-639 (in Chinese)
(张文修，魏玲，祁建军. 概念格的属性约简理论与方法[J]. 中国科学 E 辑：信息科学，2005，35(6)：628-639)

**Zhang Lei**，born in 1981. PhD candidate in computer science in Harbin Institute of Technology since 2009. Lecturer of Henan University. His main research interests include formal concept analysis，data mining，and information security.

**Zhang Hongli**，born in 1973. PhD and professor of Harbin Institute of Technology. Her main research interests include network and information security，network measurement and modeling，network computing，and parallel processing(zhanghongli@hit.edu.cn).

**Yin Lihua**，born in 1973. PhD and associate researcher. Member of China Computer Federation. Her current research interests include information security and security properties evaluation (yinlihua@software.ict.ac.cn).

**Han Daojun**，born in 1979. PhD and associate professor. His main research interests include information security and formal concept analysis(handj@henu.edu.cn).