基于属性拓扑的可视化形式概念计算

张 涛[1],任宏雷[1],洪文学[2],李 慧[1]
(1.燕山大学信息科学与工程学院,河北秦皇岛 066004;2.燕山大学电气工程学院,河北秦皇岛 066004)

摘 要: 形式概念计算是形式概念分析的重要研究内容之一.本文提出一种利用属性拓扑进行概念计算的可视化算法.该算法将属性拓扑以顶层属性为核心分解为若干子拓扑,利用可视化全路径搜索得到当前子拓扑的所有概念,进而得到该背景下的全概念集.该算法使概念计算精准并易于实现.

关键词: 形式概念分析; 属性拓扑; 全路径搜索; 可视化
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112 (2014)05-0925-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.05.014

The Visualizing Calculation of Formal Concept That Based on the Attribute Topologies

ZHANG Tao[1], REN Hong-lei[1], HONG Wen-xue[2], LI Hui[1]
(1. *School of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei* 066004, *China*;
2. *School of Electrical Engineering, Yanshan University, Qinhuangdao, Hebei* 066004, *China*)

Abstract: The calculation of formal concepts is a very important part in the theory of formal concept analysis. This paper proposes a visualization algorithm of formal concept calculation that used the attribute topology of formal context. This algorithm puts the top attributes as the cores to decompose the topology into some sub-topologies, gets all the formal concepts by the visualization full path searching in present sub-topology, combine them to get a full set of concepts of the formal context. Through the analysis of the method, using the full path searching of the sub-topology to get formal concepts not only makes the calculation of formal concept simple and easy to operate, and will not result in lost or redundant computing formal concept.

Key words: formal concept analysis; attribute topology; full path searching; visualization

1 引言

形式概念分析[1]是对形式背景中属性、对象及其关系进行分析和研究的理论,其提供了一种与传统数据分析和知识表示完全不同的方法,由德国的 WILLE 教授于 1982 年根据概念的哲学思想所提出[2,3].目前形式概念分析的应用领域已经越来越广泛,如软件工程[4],信息处理[5],知识发现[6],数据挖掘[7]等.

形式背景是形式概念分析的研究对象,其表示方法除了经典的对象属性关系二元表和传统概念格外,还包括属性偏序图[8]、属性树[9]、属性拓扑[10]等等.其中,形式概念的计算是各种表示方法研究的热点问题.经过国内外学者多年研究,目前已经从不同角度提出了多种形式概念的计算方法.通常将概念格的计算方法分为批处理算法[11,12],增量算法[13,14],并行算法[15,16],快速渐进式构造算法[17]和基于属性拓扑的形式概念计算方法等.其中,批处理算法中较为典型的是 Bordat 的自顶向下算法[11].该方法采用了递归计算,可直观的表示概念格的构造,并且有利于并行计算的实现.但是由于过程中会产生节点的重复,反而增大了运算量;增量算法中比较典型的有 Godin 算法[13].该方法将格中的节点分为不变节点、更新节点和新增节点三类,当插入一个新对象时,只需要检查那些和新对象至少有一个相同属性的节点来判断其在格中应处的位置,实现了格的构造速度的提升.但是该算法需要对概念格的分支进行修整,相对增加了算法的步骤和时间;并行算法主要思想是将形式背景拆分为子形式背景构造子格后再进行相应的合并运算.智慧来在文献[15]中基于概念格合并时不改变原有形式概念的父子关系原理,对概念格的横向和纵向两个方面进行合并运算,效率要高于单个形式概念插入

收稿日期:2013-05-02;修回日期:2013-12-17;责任编辑:孙瑶
基金项目:国家自然科学基金(No.60904100);河北省自然科学基金(No.F2011203073)

的算法.谢志鹏[17]采用渐进式构造算法,提出了树结构对概念格节点生成的作用,有效的提高算法的效率.Tao Zhang[10]提出形式背景的全新表示方法,即属性拓扑表示法,并将属性拓扑分解为子拓扑,利用拓扑中属性对象的关联及关联强度来直接计算形式背景的所有形式概念.

然而当前研究的方法普遍存在一些问题:经典的概念格计算复杂度较高且不能直观表示各属性间的关联强度;属性偏序图揭示了数据的知识结构,在大形式背景下可视性较好,但目前尚没有利用其对形式概念进行计算的研究;属性树表示法只能表示属性间的关系而不便于形式概念的计算;属性拓扑能够直观的表示形式背景中各属性间的关联性及关联强度,实现了形式背景表达与形式概念计算的统一.但当前属性拓扑对概念的计算方法逻辑性较差且不适用大数据形式背景的形式概念计算.

本文在属性拓扑思想的基础上,综合考虑形式背景的表示和形式概念计算问题,结合增量算法思想提出基于属性拓扑的可视化形式概念计算方法.本方法首先对属性拓扑进行分解,以顶层属性为中心进行子拓扑的构造.通过子拓扑的可视化表示,以顶层属性为起点,对子拓扑进行全路径搜索.该算法可以简便的得到各个属性子拓扑中的所有形式概念,并获得形式概念的直观计算过程.属性子拓扑的全路径搜索计算过程中不会造成形式概念丢失或冗余,简化了形式概念计算的过程.

2 形式背景的基本概念

形式背景是形式概念分析的数据表示方式和处理对象,以下介绍形式背景的相关定义[1].

定义 1 一个形式背景可以表示为 $K:=(G,M,I)$,是由两个集合 G 和 M 以及 G 与 M 间的关系 I 组成.其中,G 为形式背景中的对象集合,M 为形式背景中的属性集合,I 表示 G 与 M 间的关系.

定义 2 设 $K:=(G,M,I)$是一个形式背景,若 $A\subseteq G,B\subseteq M$,令

$$f(A)=\{m\in M\mid\forall\, g\in A,(g,m)\in I\}$$

及 $$g(B)=\{g_0\in G\mid\forall\, m\in B,(g_0,m)\in I\}$$

如果 A,B 满足$f(A)=B,g(B)=A$,则我们称二元组(A,B)是一个形式概念.A 是形式概念(A,B)的外延,B 是形式概念(A,B)的内涵.用 $\beta(G,M,I)$或 $\beta(K)$表示背景 $K:=(G,M,I)$上的所有形式概念集合.

定义 3 形式背景的全局概念是指以该形式背景的所有属性为内涵,对应的对象集为外延构成的形式概念(称为全属性全局概念)或者以所有的对象为外延,对应的属性集为内涵构成的形式概念(称为全对象全局概念).形式背景 $K:=(G,M,I)$下的全局概念只有两个,即$(G,f(G))$和$(g(M),M)$.

3 属性拓扑的构造与性质

3.1 形式背景的预处理

属性拓扑主要分析形式背景中属性之间的关系,其计算复杂度将随形式背景中的属性和对象数目增长而增长.因此,在不损失信息的情况下对形式背景进行预处理,其主要目的是对形式背景中的属性和对象数目进行净化表示,以降低后期计算复杂度.

定义 4 全局属性是指形式背景中全部的对象均包含的属性.在形式背景 $K:=(G,M,I)$中,若有 $m\in M$ 且 $g(m)=G$,则属性 m 称为该形式背景中的一个全局属性.与之对应,全局对象,是指具有形式背景中全部属性的对象.

全局属性定义为起点属性,全局对象定义为终点对象.

定义 5 空属性,是指形式背景中不属于任何对象的属性.在形式背景 $K:=(G,M,I)$中,若有 $m\in M$ 且 $g(m)=\varnothing$,则对象 g 称为该形式背景中的一个空对象.与之对应,空对象是指在该形式背景中不包含任何属性点对象.

空属性定义为终点属性,空对象定义为起点对象.

依据格理论可知,全局对象与全局属性不会对概念格的结构产生影响,对于概念计算是可约简的.从信息论的角度看,全局对象与全局属性的相关运算包含所有属性或对象,不具有区分特性.空对象与空属性是不与任何其他的对象属性产生关联的独立存在,对形式概念的计算过程也不产生影响.因此,运算时这四类属性对象可暂不作考虑.

定义 6 等价属性,是指在形式背景下,两个属性所属的对象集均一致.同样,等价对象是指在形式背景下,两个对象所包含的所有属性集相等.

等价属性与等价对象均符合形式背景的净化条件,在计算时可合并.

对于形式背景 $K:=(G,M,I)$,如果任意两个满足 $f(g)=f(h)$的元素 $g,h\in G$ 都有 $g=h$,且对偶的任意两个满足 $g(m)=g(n)$的元素 $m,n\in M$ 都有 $m=n$,称该形式背景是净化的.

下面结合一个典型的形式背景做分析,这个形式背景中包含了属性间的相容、包含、互斥和互补等各种关系以及全局属性、空属性和等价对象等特殊情况.其具体内容如表 1 所示.

显然,在这个形式背景中,属性 a 为全局属性(即起点属性),属性 j 为空属性,对象 8 与对象 9 出现了信息的重复(即为等价对象).因此首先需要进行形式背

景净化,去除冗余信息.净化后的形式背景如表 2.为描述简便,若不作特殊声明,下文中提到的所有形式背景均为净化背景.

表 1　典型形式背景

	a	b	c	d	e	f	g	h	i	j
1	x	x					x			
2	x	x					x	x		
3	x	x	x				x	x		
4	x		x				x	x	x	
5	x	x		x		x				
6	x	x	x	x		x				
7	x		x	x	x					
8	x		x	x		x				
9	x		x	x		x				

表 2　表 1 的净化后形式背景

	b	c	d	e	f	g	h	i
1	x					x		
2	x					x	x	
3	x	x				x	x	
4		x				x	x	x
5	x		x		x			
6	x	x	x		x			
7		x	x	x				
8		x	x		x			

3.2　属性拓扑的生成与表示

从图论角度看,属性拓扑表示是关于属性间关系的加权图表示.因此在存储上可以借鉴图的存储方式.在文献[10]中,从属性包容的角度对属性拓扑进行了邻接矩阵描述.

形式背景 $K:=(G,M,I)$中,定义 $T=(V,\boldsymbol{E})$为属性拓扑的表示.其中 $V=M$ 为拓扑的顶点集合,$\boldsymbol{E}$ 为邻接矩阵,表示拓扑中边的权值集合.

$$\boldsymbol{E}(v_i,v_j)=\begin{cases} m'_i, & \text{当 } i=j \text{ 时} \\ \varnothing, & \text{当 } m'_i\cap m'_j=\varnothing\ (i\neq j) \\ \varnothing, & \text{当 } m'_i\cap m'_j=m'_i\ (i\neq j) \\ m'_i, & \text{当 } m'_i\cap m'_j=m'_j\ (i\neq j) \\ m'_i\cap m'_j, & \text{其他} \end{cases} \tag{1}$$

式(1)中,当 $i=j$ 时,$\boldsymbol{E}(v_i,v_j)$为属性 m_i 所属对象集;当 $m'_i\cap m'_j=\varnothing\ (i\neq j)$时,$m_i$ 与 m_j 互斥,此时 $\boldsymbol{E}(v_i,v_j)=\varnothing$;当 $m'_i\cap m'_j=m_i\ (i\neq j)$时,$m'_i$ 包含于 m'_j,拓扑中表现为由顶点 m_i 不能到达顶点 m_j(式中表示为 $\boldsymbol{E}(v_i,v_j)=\varnothing$),而由顶点 m_j 却能到达顶点 m_i;当 m_i 和 m_j 为相容关系时,$\boldsymbol{E}(v_i,v_j)=m'_i\cap m'_j$.

在本文中,为便于利用属性拓扑进行形式概念计算,对文献[10]的邻接矩阵方法进行精简并提出关联矩阵表示.精简后属性拓扑的邻接矩阵表达式如式(2)所示:

$$\boldsymbol{E}(v_i,v_j)=\begin{cases} \varnothing, & g(m_i)-g(m_j)=\varnothing \\ & \text{或 } g(m_i)-g(m_j)=g(m_i) \\ g(m_i)\cap g(m_j), & \text{其他} \end{cases} \tag{2}$$

定义 $T=(V,\boldsymbol{E}')$为属性拓扑的表示,其中 $V=M$ 为拓扑中的顶点集合,$\boldsymbol{E}'$ 为关联矩阵,表示拓扑中边的走向,如式(3)所示:

$$\boldsymbol{E}'(v_i,v_j)=\begin{cases} -1, & g(m_i)-g(m_j)=\varnothing \\ 0, & g(m_i)-g(m_j)=g(m_i) \\ 1, & \text{其他} \end{cases} \tag{3}$$

在式(2)与式(3)中,当 $g(m_i)-g(m_j)=g(m_i)$时,包含两种情况:

$(i)\ g(m_i)\cap g(m_j)=\varnothing$,即 m_i 与 m_j 二者不相关,图中表示为二者间没有直接相连的边.

$(ii)\ g(m_j)=\varnothing$,显然,符合背景净化条件,在预处理中被净化掉,该情况不可能出现.

当 $g(m_i)-g(m_j)=\varnothing$时,包含三种情况:

(i)$g(m_i)=\varnothing$,符合形式背景的净化条件,这种情况不可能出现.

(ii)$g(m_i)=g(m_j)$,符合被净化的条件,不可能出现.

(iii)$g(m_i)\subset g(m_j)$,m_i 为 m_j 的伴生属性.

其他,即$\{g(m_i)-g(m_j)\}\subset(\varnothing,g(m_i))$,此时 $g(m_i)$与 $g(m_j)$间的交集既不为空集又不为 $g(m_i)$,包含两种情况:

(i)$g(m_i)\supset g(m_j)$,m_j 为 m_i 的伴生属性.

(ii)属性 m_i 与 m_j 为互不包含属性,即为相容关系.

综上,$g(m_i)-g(m_j)=g(m_i)$即表示属性 m_i 与 m_j 间为互斥关系,$g(m_i)-g(m_j)=\varnothing$ 表示属性 m_i 与 m_j 间为包含关系,其他则表示在图中与属性 m_i 直接相连的边均为单向指出边或双向边.

由式(2)与式(3)可得,表 2 对应的邻接矩阵和关联矩阵分别为 $\boldsymbol{E}(v_i,v_j)$和 $\boldsymbol{E}'(v_i,v_j)$:

$$\boldsymbol{E}(v_i,v_j)=\begin{bmatrix}
\{1,2,3,5,6\} & \{3,6\} & \{5,6\} & \varnothing & \{5,6\} & \{1,2,3\} & \{2,3\} & \varnothing\\
\{3,6\} & \{3,4,6,7,8\} & \{6,7,8\} & \{7\} & \{6,8\} & \{3,4\} & \{3,4\} & \{4\}\\
\{5,6\} & \{6,7,8\} & \{5,6,7,8\} & \{7\} & \{5,6,8\} & \varnothing & \varnothing & \varnothing\\
\varnothing & \varnothing & \varnothing & \{7\} & \varnothing & \varnothing & \varnothing & \varnothing\\
\{5,6\} & \{6,8\} & \varnothing & \varnothing & \{5,6,8\} & \varnothing & \varnothing & \varnothing\\
\{1,2,3\} & \{3,4\} & \varnothing & \varnothing & \varnothing & \{1,2,3,4\} & \{2,3,4\} & \{4\}\\
\{2,3\} & \{3,4\} & \varnothing & \varnothing & \varnothing & \varnothing & \{2,3,4\} & \{4\}\\
\varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{4\}
\end{bmatrix}$$

$$\boldsymbol{E}'(v_i,v_j)=\begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 1 & 1 & 0\\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0\\
0 & -1 & -1 & 1 & 0 & 0 & 0 & 0\\
1 & 1 & -1 & 0 & 1 & 0 & 0 & 0\\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1\\
1 & 1 & 0 & 0 & 0 & -1 & 1 & 1\\
0 & -1 & 0 & 0 & 0 & -1 & -1 & 1
\end{bmatrix}$$

已知形式背景,由式(2)式(3)分别得到其邻接矩阵和关联矩阵. T 的属性集合作为拓扑的顶点集合,按照关联矩阵画各属性间的边及其走向,按照邻接矩阵的值标注属性间的耦合关系. 对于与伴生属性相连的双向边,图中用虚线表示,以表明二者不能直接关联. 据此,可得到属性拓扑的图表示. 表 2 的属性拓扑如图 1 所示.

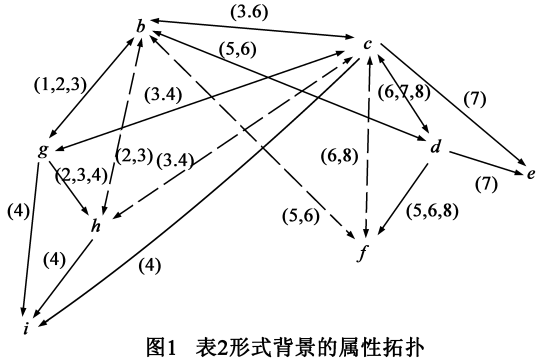


图1 表2形式背景的属性拓扑

定义 7 属性拓扑中的完全多边形是指多边形的每对顶点间都有一条边直接关联,关联形式可以为单向边或双向边.

定理 1 形式背景 $K:=(G,M,I)$中,若满足$g(m_i)\cap g(m_j)\neq\varnothing$,$(i\in T,j\in T)$,则在属性拓扑中 $M_T=\{m_i\mid m_i\in M\}$必定为完全多边形.

证明:设 $i\in T,j\in T$,若 $g(m_i)\cap g(m_j)\neq\varnothing$且 $M_T=\{m_i\mid m_i\in M\}$为非完全多边形. 则在 M_T 所属的拓扑图中,一定存在两个属性 m_a 与 m_b 无直接关联,即属性 m_a 与 m_b 为互斥关系,即 $g(m_a)\cap g(m_b)=\varnothing$. 此时与题设相矛盾. 因此,命题得证. 证毕

3.3 基于拓扑的属性分类及其性质

文献[10]中定义了顶层属性,过渡属性和底层属性. 为了形式概念计算的需要,本文引入伴生属性定义,并分析其性质.

定义 8 在 $K:=(G,M,I)$中,$m_i\in M$. 若存在集合 N,对于 $\forall m_j\in N$, $N\subseteq M-m_i$,且都满足 $g(m_i)\subset g(m_j)$,则属性 m_i 为属性 m_j 的伴生属性.

性质 1 在形式背景中,可能存在 $A\subset M$,使得任意 $a_i\in A$ 均满足 $g(m)\subset g(a_i)$. 即属性 m 可以为多个属性的伴生.

性质 2 伴生属性中,$\boldsymbol{E}(v_i,v_j)=\varnothing$ 且 $\boldsymbol{E}'(v_i,v_j)=-1$. 即在属性拓扑中,与伴生属性相连的边中至少存在一条单向指入边.

证明:由伴生属性的定义可知,若属性 m_i 为属性 m_j 的伴生属性,则 $g(m_i)\subset g(m_j)$,即 $\boldsymbol{E}(v_i,v_j)=\varnothing$,且 $\boldsymbol{E}(v_j,v_i)\neq\varnothing$,也就是 $\boldsymbol{E}(v_j,v_i)$是 m_i 的一条单向指入边. 证毕

性质 3 属性 m_i 可以是属性 m_j 的伴生,即属性 m_i 与属性 m_j 之间为包含关系,同时也可以是属性 m_k 的互不包含属性,即属性 m_i 与属性 m_k 之间为相容关系.

证明:在 $K:=(G,M,I)$中,$m_i\in M$. 显然可以存在 $m_j\in M$, $m_k\in M$,满足 $g(m_i)\subset g(m_j)$,同时满足 $g(m_i)\cap g(m_k)\neq\varnothing$, $g(m_i)\not\subset g(m_k)$, $g(m_k)\not\subset g(m_i)$. 即伴生属性同时也可以与其它属性互不包含. 证毕

性质 4 属性 m_i 可以为多重伴生属性.

定义 9 形式背景 $K:=(G,M,I)$中若存在 $g(m_i)\subset g(m_j)\subset g(m_k)$,则属性 m_i 称为属性 m_k 的多重伴生属性,记为 $m_i=B(m_k)$.

性质 5 顶层属性必不是伴生属性.

证明:对于伴生属性 $m_i\in M$, $\forall n\subseteq M-m_i$,都满足 $g(m_i)\subset g(n)$. 显然与顶层属性的定义相矛盾. 因此顶层属性一定不是伴生属性. 证毕

性质 6 顶层属性在拓扑图中表现为与之相连的边只有单向指出边和双向边,不存在单向指入边.

证明:由定义可知,顶层属性 m_j 对任意的 $\boldsymbol{E}(m_i,m_j)\neq\varnothing$,都有 $\boldsymbol{E}(m_j,m_i)\neq\varnothing$,即不存在满足 $\boldsymbol{E}(m_i,m_j)\neq\varnothing$,同时 $\boldsymbol{E}(m_j,m_i)=\varnothing$,条件的属性 m_i. 因此不存在单向指入边. 证毕

定义 10 若底层属性同时也是多重伴生属性,则

称 m_i 为最小伴生属性.

性质 7　过渡属性可能为伴生属性,但不能是最小伴生属性.

证明:由过渡属性 m_i 的定义,与之相邻的边中,可能存在 $\boldsymbol{E}(m_k, m_i) \neq \varnothing$, $i \neq k$,使 $\boldsymbol{E}(m_i, m_k) = \varnothing$,即 m_i 存在单向指入边,因此过渡属性可能为伴生属性.同时,与过渡属性相邻边中一定至少存在一个 $\boldsymbol{E}(m_i, m_j) \neq \varnothing$, $i \neq j$,即 m_i 存在指出边,因此过渡属性不可能为最小伴生属性.证毕

显然,图 1 中属性 b, c, d, g 为顶层属性;属性 f, h 为过渡属性;属性 e, i 为底层属性,同时也是最小伴生属性;属性 e, f, h, i 为伴生属性;其中属性 i 为多重伴生.

4　子拓扑的可视化形式概念计算

4.1　拓扑的分解与子拓扑的构造

属性拓扑的分解以顶层属性为中心,其目的是为了减少不必要步骤,简化形式概念的计算.

根据属性拓扑中各属性类别及其之间的关系,可判断是否可分解为子拓扑.下面是三种不可分解的情形:

(1)若属性拓扑中只包含一个顶层属性,则该拓扑不可分解.

(2)若属性拓扑为完全多边形,则该拓扑不可分解.

证明:设形式背景的属性拓扑中,有对于 $\forall\, i \neq j$,有 $\boldsymbol{E}(m_i, m_j) \cup \boldsymbol{E}(m_j, m_i) \neq \varnothing$,则以 $T = (V, \boldsymbol{E})$ 中的一个顶层属性 m_0 为中心的子拓扑 $subT_0 = (subV_0, sub\boldsymbol{E}_0)$ 中,顶点属性集为 $subV_0 = V = M$,与之对应的 $G_0 = G$,子拓扑中各边的对应权值为 $sub\boldsymbol{E}_0(m_i, m_j) = \boldsymbol{E}(m_i, m_j)$, $sub\boldsymbol{E}_0(m_j, m_i) = \boldsymbol{E}(m_j, m_i)$,其中 $\forall\, m_i, m_j \in M$,因此,可以得到 $subT_0 = T$,即,原属性拓扑不可分解.证毕

(3)若属性拓扑非完全多边形,但顶层属性间两两相关,则该拓扑不可分解.

证明:设形式背景 $K := (G, M, I)$ 的属性拓扑为 $T = (V, \boldsymbol{E})$,假设任意一个顶层属性 m_i 相连的边都满足 $\boldsymbol{E}(m_i, m_j) \neq \varnothing$, $\forall\, j \neq i$,则以 m_i 为中心的子拓扑 $subT_i = (subV_i, sub\boldsymbol{E}_i)$ 中,顶点属性集为 $subV_i = V = M$,与之对应的 $G_i = G$,子拓扑中各边的对应权值为 $sub\boldsymbol{E}_i(m_k, m_j) = \boldsymbol{E}(m_k, m_j)$, $sub\boldsymbol{E}_i(m_j, m_k) = \boldsymbol{E}(m_j, m_k)$,其中 $\forall\, m_k, m_j \in M$,因此,可以得到 $subT_i = T$,即,原属性拓扑不可分解.证毕

设形式背景 $K := (G, M, I)$ 可分解,其属性拓扑中的顶层属性的个数为 n,则子拓扑数最多也为 n.拓扑的分解与子拓扑的构造步骤如下:

Step 1　选点.自顶点属性集中选取出度最小的顶层属性 m 为子图的中心属性.

Step 2　构图.在属性拓扑图的剩余部分中,将与属性 m 之间关联的属性作为属性 m 为中心子图的顶点集,记为 $subV$.

Step 3　清理.对于多重伴生属性 $m_0 = B(a_i)$,若 $\forall\, a_i \notin subV$,则删除顶点 m_0.

Step 4　画边.对于 *step* 3 整理后子图顶点集 $subV$,取原拓扑中这些顶点间的边集作为子图边集.即 $sub\boldsymbol{E} = \{(v_i, v_j) \mid v_i \in subV, v_j \in subV\}$.对于与伴生属性相连的双向边,用虚线表示.

Step 5　整理.在原拓扑中,删除顶点 m 及其相关边,并将 m 从顶点属性集移除,若此时顶点属性集为空,则停止运算,否则进行 *step* 1.

该过程的构造流程图如图 2 所示.

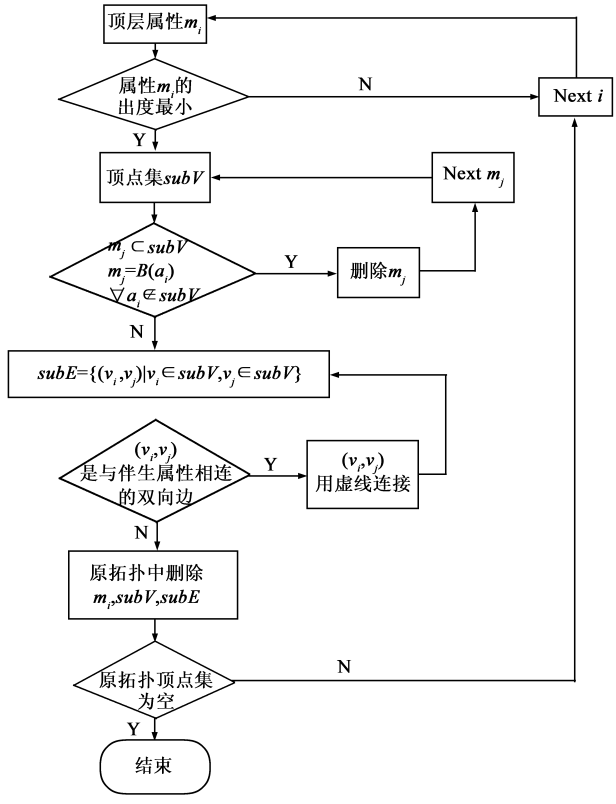
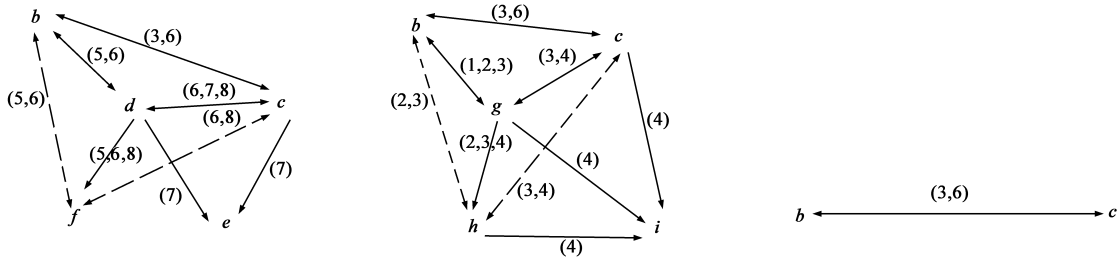


图2　子拓扑分解流程图

由上述属性拓扑的分解和子拓扑的构造步骤,可得图 1 的属性子拓扑如图 3 所示.

4.2　基于属性拓扑的形式概念计算

由属性拓扑的构造可知,其包含了背景中所有属性对间的关联及其关联强度,而子拓扑的分解过程并没有破坏形式概念计算所需关联.因此属性拓扑的形式概念计算定理在属性子拓扑中同样成立.



(*a*) *d*为中心子拓扑 (*b*) *g*为中心子拓扑 (*c*) *b*为中心子拓扑

图3 图1对应子拓扑

引理 1 属性拓扑的全路径搜索可计算全模式集合.

证明:属性拓扑 $T=(V,\boldsymbol{E})$中,$V=M$ 为拓扑的顶点集合,$\boldsymbol{E}$ 为拓扑中边的集合,其中边的权值可按式(1)得到,即属性拓扑中包含了形式背景中所有的属性及各属性间关于对象的关联强度.而搜索过程中若下一个到达的属性为 m_i,路径上已经经过的属性集为 M_i,将计算 $g(m_i)\cap g(M_i)$.若 $g(m_i)\cap g(M_i)\neq\varnothing$,则存储$((g(m_i)\cap g(M_i)),(m_i\cup M_i))$,并进行下一步.

如此搜索完整体属性拓扑,得到的便是图中所有属性间的所有关联,其对应的对象集就是其所有的关联强度.于是二者构成的二元组,即为属性拓扑的全模式集合.证毕

定理 2 利用属性拓扑的全路径搜索算法可以得到形式背景的所有形式概念,并且不会产生伪概念.

证明:由引理 4.1 可知,属性拓扑的全路径搜索可得到形式背景的全模式集合,包括了所有的形式概念.在搜索过程中若存在集合 $K\subset\{M-\{m_i,m_t\}\}$,其中 m_i 为顶层属性,m_t 为与 m_i 直接关联的其中一个属性,可使得 $(g(m_im_t),m_im_tK)$,则将上一步所存储的 $(g(m_im_t),m_im_t)$更新为$(g(m_im_t),m_im_tK)$.更新过程消除了伪概念产生的条件.故属性拓扑的全路径算法可得到形式背景的所有概念,且不会产生伪概念.证毕

属性拓扑的分解过程中,在选点和构图步骤里都体现了全拓扑与子拓扑的一致性,不会造成与形式概念相关的属性对象关联丢失.因此,对子拓扑全局搜索同样可以得到子拓扑中的所有形式概念,而不会产生伪概念.

4.3 算法描述

利用上述得到的子拓扑进行形式概念计算.设子拓扑分别为 $subT(V_1,\boldsymbol{E}_1),subT(V_2,\boldsymbol{E}_2),\cdots,subT(V_n,\boldsymbol{E}_n)$.在得到子拓扑中,设 M 为子拓扑的全部属性集.其顶层属性集为$\{A\mid m_i\in A\}$,最小被包含属性集为$\{B\mid m_j\in B\}$.利用图的全路径搜索原理,分别以集合 A 中的各 m_i 为起点,以集合 B 中的各 m_j 为终点,作全路径搜索.若图中 $B=\varnothing$,则直接对集合 M 中的各属性进行全路径搜索.

Step 1 取与顶层属性 m_i 直接关联的 m_t,存储$(g(m_im_t),m_im_t)$.

Step 2 若$\exists K\subset\{M-\{m_i,m_t\}\}$,使得 $g(m_im_t)=g(m_im_tK)$,则将上一步所存储的$(g(m_im_t),m_im_t)$将更新为$(g(m_im_t),m_im_tK)$.

Step 3 若$\exists n\in\{M-\{m_i,m_t\}\}$,都有 $g(m_im_t)\neq g(m_im_tn)\neq\varnothing$,则上一步存储不改变的情况下,再存储$(g(m_im_tn),m_im_tn)$.

Step 4 判断与 m_i 直接关联的属性是否全部搜索过.若是,则结束;否则返回到 step1,进行下一个属性 m_r 的遍历.

对子拓扑 $subT(V_1,\boldsymbol{E}_1),subT(V_2,\boldsymbol{E}_2),\cdots,subT(V_n,\boldsymbol{E}_n)$进行全路径搜索,便可以得到各子拓扑中对应所有的形式概念集 $\beta(subT(V_1,\boldsymbol{E}_1)),\beta(subT(V_2,\boldsymbol{E}_2)),\cdots,\beta(subT(V_n,\boldsymbol{E}_n))$.

该算法流程图如图 4 所示.

依据算法理论可知,采用属性子拓扑方法进行可视化形式概念计算,计算复杂度为 $O(2^n)$,与目前主流方法复杂度相同.依以上算法进行子图的遍历,图 5 为遍历的过程示意图.

图 5 中,每个节点代表遍历路径中所路过的属性集,边为其路径,权值为右侧节点属性集所属对象集,其中实线边上的权值及其右侧属性集组成的二元组即为子拓扑中的形式概念,虚线边连接的右侧属性集为伪内涵,其在搜索过程中被内涵更新替代.

由图 5 的示意图可知:

d 为中心属性的子拓扑中:(56, bdf),(6, $bdcf$),(678, cd),(7, cde),(68, cdf),(568, df).

g 为中心属性的子拓扑中:(123, bg),(3, $bcgh$),(23, bgh),(34, cgh),(4, $cghi$),(234, gh).

b 和 c 为中心属性的子拓扑中:(36, bc).

又由中心属性必为形式概念内涵[10],得到:(5678, d),(1234, g),(12356, b),(34678, c).进而直接得到了各子拓扑中的所有形式概念.

加上全局概念,即(12345678, $\varnothing$)和($\varnothing$, $bcde$-

fghi),得净化形式背景下的所有形式概念.

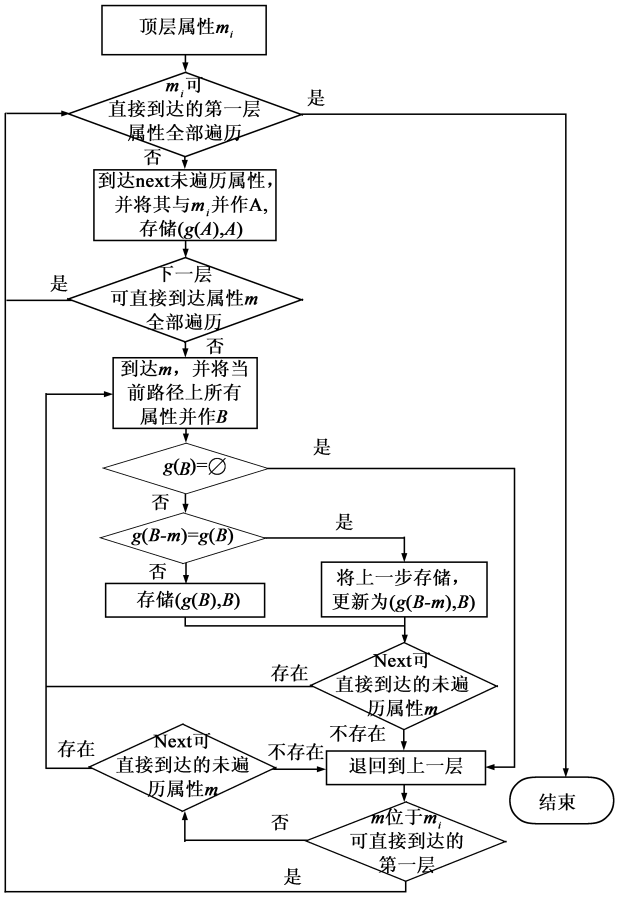


图4　子拓扑计算形式概念算法流程图

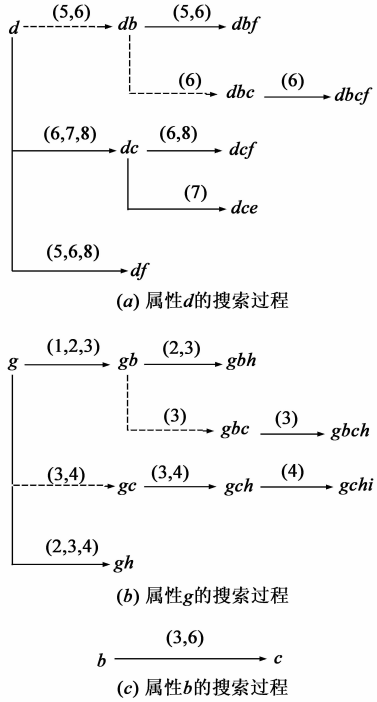


图5　子拓扑求形式概念的遍历过程示意图

4.4　形式概念的原始形式背景恢复

4.3 节计算中得到净化形式背景下的所有形式概念,而实际应用时需要的是净化前的原始形式背景下的形式概念.于是,对于形式概念的原始形式背景恢复也是必要步骤.

对于起点属性,只需在每个形式概念的内涵直接做添加;对于终点对象,同样要在每个形式概念的外延直接添加这个对象.而终点属性和起点对象,由于与其他任何属性对象不产生关联,则不需要在形式概念上做出表示.对于等价属性与等价对象,则需要直接将合并后的集合拆开,分别作为内涵或外延中的元素.

如表 1 中的属性 a 为起点属性,则在最后得到的所有形式概念的内涵集中,均需要添加属性 a 为其元素;属性 j 为终点属性,只需在全属性全局概念的内涵中加上属性 j 作为全部内涵;对象 8 与对象 9 为等价对象,前文中用对象 8 代替了{8,9},所以,恢复时需要用对象{8,9}替换回对象 8.

于是得到全部形式概念的原始形式背景复原如下:$\beta(K)=\{(123,abg),(3,abcgh),(23,abgh),(34,acgh),(4,acghi),(234,agh),(36,abc),(56,abdf),(6,abdcf),(6789,acd),(7,acde),(689,acdf),(5689,adf),(56789,ad),(1234,ag),(12356,ab),(346789,ac),(123456789,a),(\varnothing,abcdefghij)\}$.

5　结论

本文考虑形式背景拆分的思想,以顶层属性为中心,按照一定的规则纳入其他与顶层属性相关的顶点属性集,将属性拓扑分解若干个子拓扑,并结合增量算法思想,以子拓扑中顶层属性为起点,其他顶点属性为终点进行全路径搜索,进而获得形式背景的所有形式概念.该方法不仅使整个过程更具逻辑性和可操作性,并且易于实现,适用于大规模数据集.相对于整体属性拓扑,对分解后的子拓扑进行操作缩小了算法的搜索空间,提高了运算效率,降低了计算复杂度.属性拓扑为形式背景提供了全新的表示方法,下一步的工作将对本方法进行完善和优化,并将之实现于应用领域.

参考文献

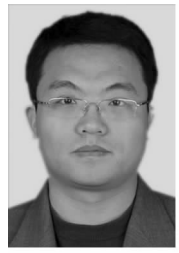
[1] Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations[M]. New York: Springer-Verlag, 1999.

[2] Radim Belohlavek, Erik Sigmund, Jir r˘i Zacpal. Evaluation of IPAQ questionnaires supported by formal concept analysis[J]. Information Sciences, 2011, 181(10): 1774 – 1786.

[3] Tam T, Nguyen, Siu Cheung Hui, Kuiyu Chang. A lattice-based approach for mathematical search using formal concept analysis [J]. Expert Systems with Applications, 2012, 39(5): 5820 –

5828.
[4] Tonella. Using a concept lattice of decomposition slices for program understanding and impact analysis[J]. IEEE Transactions on Software Engineering, 2003, 29(6): 495 – 509.
[5] Fethi Ferjani, Samir Elloumi, Ali Jaoua, et al. Formal context coverage based on isolated labels: An efficient solution for text feature extraction[J]. Information Sciences, 2012, 188(1): 198 – 214.
[6] Jin-hai Li, Chang-lin Mei, Yue-jin Lv. Knowledge reduction in real decision formal contexts[J]. Information Sciences, 2012, 189(15): 191 – 207.
[7] Mehdi Kaytoue, Sergei O Kuznetsov, Amedeo Napoli, et al. Mining gene expression data with pattern structures in formal concept analysis[J]. Information Sciences, 2011, 181(10): 1989 – 2001.
[8] Feng-jie Fan, Wen-xue Hong, Xin Li, Tao Zhang, Xu-long Liu, Wei-dong Chen, Jun Jing. Research on compatibility of prescription of TCP based on the principle of attribute partial order chart[A]. Proceedings of the First International Conference on Instrumentation, Measurement, Computer, Communication and Control(IMCCC)[C]. Beijing: IEEE, 2011. 82 – 86.
[9] 张涛,洪文学,路静. 形式背景的属性树表示[J]. 系统工程理论与实践, 2011, 31(s2): 197 – 202.
Zhang Tao, Hong Wen-xue, Lu Jing. Attribute tree representation for formal context[J]. Systems Engineering-Theory & Practice, 2011, 31(s2): 197 – 202. (in Chinese)
[10] Tao Zhang, Hong-lei Ren, Xiaomin Wang. A calculation of formal concept by attribute topology[J]. ICIC Express Letters Part B: Applications, 2013, 4(3): 793 – 800.
[11] Ganter B. Formal Concept Analysis: Algorithmic Aspects[EB/OL]. http://www.math.tu-dmsden.de/-ganter/cl02/, 2010 – 03 – 06.
[12] Huai-guo Fu. A parallel algorithm to generate formal concepts for large data[A]. Proceedings of the Second International Conference on Formal Concept Analysis(ICFCA)[C]. Berlin: Springer, 2004. 394 – 401.
[13] Godin R, Missaoui R, Alaoui H. Incremental concept formation algorithms based on Galois(concept) lattices[J]. Computational Intelligence, 1995, 11(2): 246 – 267.
[14] Merwe D vander, Obiedkov S, Kourie D. Add intent: A new incremental algorithm for constructing concept lattices[A]. Proceedings of the Second International Conference on Formal Concept Analysis(ICFCA)[C]. Berlin: Springer, 2004. 372 – 385.
[15] 智慧来,智东杰,刘宗田. 概念格合并原理与算法[J]. 电子学报, 2010, 38(2): 455 – 459.
Zhi Hui-lai, Zhi Dong-jie, Liu Zong-tian. Theory and algorithm of concept lattice union[J]. Acta Electronica Sinica, 2010, 38(2): 455 – 459. (in Chinese)
[16] 李云,刘宗田,陈▮,徐晓华,程伟. 多概念格的横向合并算法[J]. 电子学报, 2004, 32(11): 1849 – 1854.
Li Yun, Liu Zong-tian, Chen Ling, Xu Xiao-hua, Cheng Wei. Horizontal union algorithm of multiple concept lattices[J]. Acta Electronica Sinica, 2004, 32(11): 1849 – 1854. (in Chinese)
[17] 谢志鹏,刘宗田. 概念格的快速渐进式构造算法[J]. 计算机学报, 2012, 25(5): 489 – 496.
Xie Zhi-peng, Liu Zong-tian. A fast incremental algorithm for building concept lattice[J]. Chinese Journal of Computers, 2002, 25(5): 489 – 496. (in Chinese)

作者简介



张 涛 男, 1979 年 3 月生于河北省唐山市. 现为燕山大学副教授. 主要从事形式概念分析, 可视化模式识别等领域的研究.
E-mail: zhtao@ysu.edu.cn

任宏雷 女, 1989 年 4 月出生于河北省唐山市. 现为燕山大学信息科学与工程学院硕士研究生. 主要研究方向为形式概念分析, 拓扑理论.
E-mail: yuizhizao@163.com