软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(12):3129−3145 [doi: 10.13328/j.cnki.jos.005238]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

属性拓扑的并行概念计算算法*

张 涛, 白冬辉, 李 慧

(燕山大学 信息科学与工程学院,河北 秦皇岛 066004)

通讯作者: 张涛, E-mail: zhtao@ysu.edu.cn

摘 要: 随着并行计算时代的到来,形式概念的并行计算成为形式概念分析领域的研究热点之一.以属性拓扑为基本表示形式,通过属性拓扑的图特性进行并行概念计算算法设计.首先,根据属性拓扑中属性的伴生关系对属性拓扑进行自下而上的分解,将一个整体拓扑分解为若干个子拓扑;其次,根据属性间的相关关系去除各子拓扑间的概念耦合,保证不同子拓扑在概念计算层面的各自独立性,以避免后期合并运算的大规模时间消耗;最后,在各子拓扑上进行概念计算,并将各子拓扑概念直接累加可得原始背景的全部概念集合.实验结果表明:所提方法不但可以无重复地计算全部概念,而且可以根据硬件平台情况提高计算效率,减少概念计算所需时间.

关键词: 属性拓扑;概念计算;形式概念分析;并行计算;自下而上分解

中图法分类号: TP181

中文引用格式: 张涛,白冬辉,李慧.属性拓扑的并行概念计算算法.软件学报,2017,28(12):3129−3145. http://www.jos.org.cn/1000-9825/5238.htm

英文引用格式: Zhang T, Bai DH, Li H. Parallel concept computing based on bottom-up decomposition of attribute topology. Ruan Jian Xue Bao/Journal of Software, 2017,28(12):3129−3145 (in Chinese). http://www.jos.org.cn/1000-9825/5238.htm

Parallel Concept Computing Based on Bottom-Up Decomposition of Attribute Topology

ZHANG Tao, BAI Dong-Hui, LI Hui

(School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China)

Abstract: With the arrival of parallel computing era, parallel computing of formal concepts has become a hot issue in the field of formal concept analysis. This paper proposes a parallel concept computing algorithm by means of the graph characteristics of an attribute topology used in representing formal context. First, according to the parent relations, the bottom-up decomposition of attribute topology is conducted to generate sub-topologies. Then, concept-couplings among sub-topologies are removed based on the correlations in attribute-pairs in order to ensure the independence of the sub-topologies when carrying out concept computing and then to avoid large time consumption of the merging operation in later stage. Finally, all the concepts without repetition can be calculated by accumulating directly all the concept-sets computed in different sub-topologies. The experiment shows that the approach proposed in this paper can not only obtain all the concepts without repetition, but also improve the computational efficiency in accordance with the hardware platform and reduce the time required for the concept calculation.

Key words: attribute topology; concept computing; formal concept analysis; parallel computing; bottom-up decomposition

形式概念分析(formal concept analysis)[1]是应用数学的一个分支,其中,形式背景为研究对象和数据来源,形

* 基金项目: 国家自然科学基金(61201111); 河北省自然科学基金(F2015203013); 河北省社会科学基金(HB14YY005); 燕山大学信息科学与工程学院学术骨干培养计划(XSGG2015003)

Foundation item: National Natural Science Foundation of China (61201111); Hebei Province Natural Science Foundation of China (F2015203013); Hebei Province Social Science Fund (HB14YY005); Academic Backbone Training Program of School of Information Science and Engineering in Yanshan University (XSGG2015003)

收稿时间: 2016-04-19; 修改时间: 2016-11-14; 采用时间: 2016-12-19; jos 在线出版时间: 2017-03-24

CNKI 网络优先出版: 2017-03-24 15:31:37, http://kns.cnki.net/kcms/detail/11.2560.TP.20170324.1531.002.html

式概念描述了内涵和外沿的统一,概念格描述了形式概念之间的泛化和例化关系.由于形式概念和概念格可以体现数据集中很重要的信息[2],分别与认知的基本单元和认知的过程相对应,使得形式概念分析在软件工程[3]、知识发现[4,5]、数据挖掘[6,7]、认知计算[8]等很多领域得到了广泛的应用.概念格的构建是两个步骤的组合:一是计算形式概念,二是求取形式概念间的偏序关系.因此,形式概念的计算作为形式概念分析中的基础且核心问题,受到了广泛的关注和研究.直观上看,概念格的生成和概念的计算需要枚举全体对象和属性的所有子集,其复杂度和形式背景的规模呈指数增长的关系.因此,如何降低应用的复杂性,减少时间的消耗,同时产生精确的、适合阅读和分析的概念格,是当前面临的一个重要挑战.

近年来,并行和分布式产生了飞速的发展,并行算法成为处理大规模数据的基本方法之一,它允许将工作负荷分摊到一组计算单元中,并通过多种协作模式解决大量数据的处理问题.在形式概念分析的很多应用中,大规模的二元关系形式背景越来越常见,并行的形式概念计算方法也成为降低应用复杂性的研究热点之一.在形式概念的计算中,也出现了很多种并行算法.如,Kengue 提出了 DAC-ParaLaX 算法[9],该算法将形式背景分解为 D&C 树,树中每个叶子节点是只有一个属性的形式背景,中间节点则是其孩子节点的并置;叶子节点对应的概念格可以直接得出,第 i–1 层节点的概念格则由第 i 层的概念格获得,通过将每层中概念格的处理分配到多个线程中实现并行计算;当处理到根节点时,即得到了完整的概念格.Krajca 提出的 PCbO 算法[10]从概念格的最顶端概念($f(\phi)$,$g(f(\phi))$)开始,使用深度优先搜索方法,从根节点开始递归生成包含 L 个属性的形式概念(A,B),并将这些形式概念存入待处理的队列;然后,将每一个待处理的形式概念对应地放入一个线程,求取内涵扩大的新的概念.其中利用了概念计算的封闭性,即:可以将一个已知概念的内涵扩大或外延缩小来获得新的概念,表示为 *GenerateFrom*((A,B),y),并讨论了如何实现概念的去重,以完成全体概念的计算.其后提出一种加入了 Map-Reduce 框架的算法[11],将 *GenerateFrom* 拆分为 *MapConcepts* 和 *ReduceConcepts*,分别完成概念的生成和概念的去重,并用广度优先搜索替换深度优先搜索算法,这样的好处是将所有的概念按层输出.Bhatnagar 提出的并行算法[9]中,在 Map-Reduce 框架中求取一个足够充足的概念集合,然后利用这个概念集合单线程串行地枚举其他的形式概念.但 Krajca 与 Bhatnagar 所提出的并行概念计算方法都是基于 Map-Reduce 框架,因此不可避免地会产生线程之间的同步锁和线程等待问题.文献[12,13]分别提出了两种概念格并行构造算法,这两种算法分别基于闭包系统的划分和分解的思想.马冯提出了 VCMDCL 算法[14],先将形式背景纵向拆分,然后将分布在各站点上的、具有相同属性集的形式背景构建出的子概念格进行有效合并.文献[15]通过对 $f(A)$和 $g(B)$进行修正,以适应研究客体的异构信息,讨论了异构数据集上的偏序形成,并介绍了一种广义概念格的并行生成算法.在模糊概念格方面,张卓通过简化搜索空间表示、划分和遍历过程,划分子搜索空间提出了 ParaFuNeC 算法[16]和基于负载均衡的模糊概念并行构造算法[17].

属性拓扑(attribute topology)[18]是一种新型的形式背景的表示方法,其思想来源于计算机的网络拓扑结构.在属性拓扑中,将每一个属性看做是一个信息终端,则整个的形式背景表现为终端间的信息沟通拓扑结构[19].该结构可以通过加权图的形式进行直观展现,使其具有良好的可视化特性.由于人脑对视觉信息的处理要优于对文本和数字的处理,这使得属性拓扑在信息理解方面具有一种天然的优势.目前,属性拓扑在挖掘关联规则、认知计算等领域都已有所发展[18−20],也提出了多种形式概念的计算方法[21−25],但目前尚缺乏并行概念计算算法的研究.以图为基本思想的形式背景表示方法还有直观图[26]、属性偏序图[27]等方法,也同样未见关于并行计算的研究报道.基于此,本文在属性拓扑的理论基础上提出了一种基于属性拓扑的并行概念计算方法,并通过数学证明和实验验证了其可行性.

本文在属性拓扑的理论基础上,寻求一种高效的并行形式概念计算方法.首先将属性拓扑进行分解为子属性拓扑,为并行计算提供数据来源;其次提出了子拓扑的约简方法,为并行计算去除概念耦合;最后给出了一种基于属性拓扑的并行概念计算方法,该方法以分解和约简为基础,可以避免 Map-Reduce 并串交叉实现全并行的形式概念计算,并通过数学证明和实验验证了其可行性.

本文第 1 节简要介绍形式概念分析和属性拓扑的基本概念.第 2 节给出一种基于伴生关系的自下而上的属性拓扑分解方法.第 3 节通过讨论属性合并和概念生成的关系完成分解后子属性拓扑的约简,以降低子属性拓

扑的规模以及子属性拓扑之间的耦合度.第 4 节给出一种几乎全并行概念计算的算法和流程.第 5 节通过实验验证该算法的可行性,并分析该算法的性能.第 6 节是结束语.

1 基本概念

为了便于理解本文提出的算法,在本节简要地回顾了形式概念分析和属性拓扑相关的基本概念理论.

定义 1[1]. 形式背景可以用三元组$\mathbb{K}:=(G,M,I)$表示,其中,G 表示所有对象的集合;M 表示所有属性的集合;$I\subseteq G\times M$ 表示对象与属性之间的关系,$G\times M$ 表示的是集合 G 与集合 M 的笛卡尔积.

定义 2[1]. 设$\mathbb{K}=(G,M,I)$是一个形式背景,若 $A\subseteq G,B\subseteq M$,令:

$$f(A):=\{m\in M|\forall g\in A,(g,m)\in I\} \tag{1}$$

及

$$g(B):=\{g\in G|\forall m\in B,(g,m)\in I\} \tag{2}$$

如果 A,B 满足 $f(A)=B,g(B)=A$,则称二元组(A,B)是形式背景$\mathbb{K}$中的一个概念.并将 A 称为概念(A,B)的外延,B 称为概念(A,B)的内涵.由此可见,概念是外延和内涵的统一.通常用 $\mathfrak{B}(G,M,I)$或 $\mathfrak{B}(\mathbb{K})$表示形式背景$\mathbb{K}=(G,M,I)$上的所有概念构成的集合.

由于原始形式背景中可能会存在冗余信息,Wille 给出了净化的形式背景的概念[1].属性拓扑是来源于网络拓扑图的思想,以属性为节点、以属性间的耦合关系为边的图.为了简洁对形式背景进行表示,在给出属性拓扑的定义之前,首先给出一种形式背景的预处理方法(见表 1).

Table 1 The pretreatment in a formal context

表 1 形式背景的预处理

名称	条件	处理方式
空对象	$f(g)=\varnothing,\forall g\in G$	忽略对象 g
全局对象	$f(g)=M,\forall g\in G$	忽略对象 g
对等对象	$f(g)=f(h),\forall g,h\in G$	忽略对象 g 或 h
空属性	$g(m)=\varnothing,\forall m\in M$	忽略属性 m
全局属性	$g(m)=G,\forall m\in M$	忽略属性 m
对等属性	$g(m)=g(n),\forall m,n\in M$	忽略属性 m 或 n

生物和水是形式概念分析领域中最为常见的形式背景之一(列于表 2(a)),背景中列举了 8 种生物(对象)所具有的特征(属性),其中,对象 1~对象 8 分别表示蚂蝗、娃娃鱼、青蛙、狗、水草、芦苇、豆、玉米,属性 a~属性 i 分别表示需要水、在水中生活、在陆地生活、有叶绿素、双子叶植物、单子叶植物、能移动、有四肢、哺乳动物.在该背景中,所有的生物都需要水来生存,即属性 a 是一个全局属性,满足$f(a)=M$,因此根据表 1 中的预处理条件及方式,可以得到预处理后的生物和水形式背景(列于表 2(b)).

Table 2 The formal context of Living Beings and Water

表 2 生物和水的形式背景

(a) 原始的形式背景

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

(b) 预处理后的形式背景

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

定义 3[23]. 当给定一个形式背景$\mathbb{K}=(G,M,I)$,定义 $AT:=(V,Edge)$为属性拓扑的邻接矩阵表示.其中,V 为属性

拓扑的顶点集合,这里考虑形式背景中所有的属性,取 $V=M.Edge$ 称为该属性拓扑的邻接矩阵.

$$Edge(m_i,m_j)=\begin{cases}\varnothing, & g(m_i)-g(m_j)=\varnothing\\ \varnothing, & g(m_i)-g(m_j)=g(m_i)\\ g(m_i)\cap g(m_j), & \text{其他}\end{cases} \tag{3}$$

其中,$m_i,m_j\in M$.

对于表 2(b)列出的形式背景对应的属性拓扑为 $AT=(V,Edge)$,其中,顶点集合 $V=M=\{b,c,d,e,f,g,h,i\}$,由公式(3)可以求得属性拓扑的邻接矩阵,如 $Edge(b,c)=Edge(c,b)=\{3,6\}$.属性拓扑可以与形式背景形成一一对应的关系,其中,以属性为顶点,邻接矩阵中的元素直观地体现了该属性对之间的耦合关系和耦合强度;同时,直观地描述了属性对间的包含、相容和互斥.将属性对间的 3 种关系分别用单向箭头的连线、双向箭头的连线和无连线表示,将邻接矩阵中的对应元素作为连线上的权值,所得属性拓扑如图 1 所示.属性拓扑作为一种形式背景的表示方法,其邻接矩阵的存储方式在空间开销上高于形式背景的传统表示法.为了权衡计算效率和存储开销,可以使用十字链表等数据结构进行存储.为了便于理解,本文中采用邻接矩阵的方式定义描述属性拓扑.

定义 4[23]. 根据属性间的关联性不同,可以将所有属性划分为两个属性集合.

$$\begin{cases}SupAttr=\{v_i\mid\forall Edge(v_i,v_j)\neq\varnothing,\forall Edge(v_j,v_i)\neq\varnothing,j\neq i\}\\ SubAttr=V-SupAttr\end{cases} \tag{4}$$

若属性 $v\in SupAttr$,则称属性 v 为顶层属性,顶层属性一定可以独立的构成一个概念的内涵;若属性 $v\in SubAttr$,则称属性 v 为伴生属性,伴生属性一定不可以独立地构成一个概念的内涵,即:若一个概念的内涵中存在伴生属性,那么在这个概念的内涵中必然至少存在一个顶层属性.

如图 1(a)所示的属性拓扑中,根据定义 4 可以求得 $SupAttr=\{b,c,d,g\}$,$SubAttr=\{e,f,h,i\}$.在属性拓扑中,顶层属性和伴生属性的分界线如图 1(b)所示.

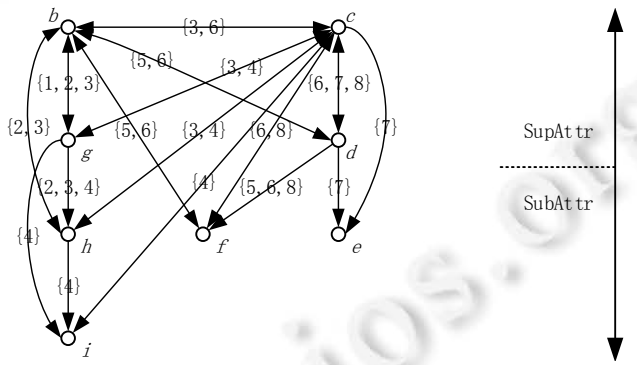


Fig.1 Attribute topology and attribute classification of Table 2(b)

图 1 表 2(b)对应的属性拓扑以及顶层伴生属性分界线

从图 1 中可以明显地看出,邻接矩阵这种存储方式是存在很大空间冗余度的.对于一般的属性拓扑,以十字链表的方式存储将极大地减小空间占用,同时保持较为良好的数据吞吐效率.但由于邻接矩阵在理解和存取上都较为容易,因此本文仍然选用这种表示方式,以便更好地描述本文算法.

定义 5. 设 $m_i\in M$,定义映射:

$$m_i\rightarrow Parent(m_i)=\{m\mid g(m_i)\subset g(m),\forall m\in M\} \tag{5}$$

若存在 $m_0,m_1\in M$,且 $m_0\in Parent(m_1)$,即 $g(m_1)\subset g(m_0)$,则称属性 m_0 和属性 m_1 构成一组父子属性对,称属性 m_0 为属性 m_1 的父属性,同时称属性 m_1 为属性 m_0 的子属性.

定理 1. 若 $(A,B)\in\mathfrak{B}(\mathbb{K}),m_j\in B$,则 $\forall m_i\in Parent(m_j),m_i\in B$.

证明:使用反证法.假设 $m_i\notin B$,

由于 $m_j\in B,B\subseteq M$

因此,$(A,B)\in\mathfrak{B}(\mathbb{K})$等价于$(A,B\cup m_j)\in\mathfrak{B}(\mathbb{K})$.

又 $m_i\in Parent(m_j)$,即 $g(m_j)\subset g(m_i)$,

则 $A=g(B\cup m_j)=g(B)\cap g(m_j)=g(B)\cap g(m_j)\cap g(m_i)=g(B\cup m_j\cup m_i)$

即$(A,B\cup m_j)=(A,B)\notin\mathfrak{B}(\mathbb{K})$

与题设不符,因此假设 $m_i\notin B$ 不成立,即 $m_i\in B$. □

定理 2. 若 $Edge(m_i,m_j)=Edge(m_j,m_i)=\varnothing,(A,B)\in\mathfrak{B}(\mathbb{K})-\{(\varnothing,M)\}-\{(G,\varnothing)\}$,则$\forall m_i\in B,\forall m_j\in M,m_j\notin B$.

证明:使用反证法.假设 $m_j\in B$,

由于 $m_i\in B,m_j\in B,B\subseteq M$

因此,$(A,B\cup m_i\cup m_j)\in\mathfrak{B}(\mathbb{K})-\{(\varnothing,M)\}-\{(G,\varnothing)\}$

因为 $A=g(B\cup m_i\cup m_j)=g(B)\cap g(m_i)\cap g(m_j)$

又 $Edge(m_i,m_j)=Edge(m_j,m_i)=\varnothing$

则 $g(m_i)\cap g(m_j)=\varnothing,A=\varnothing$

因此,$(A,B\cup m_i\cup m_j)=(A,B)\notin\mathfrak{B}(\mathbb{K})-\{(\varnothing,M)\}-\{(G,\varnothing)\}$

与题设不符,因此假设 $m_j\in B$ 不成立,即 $m_j\notin B$. □

2 属性拓扑的自下而上分解

当形式背景规模较大时,求取概念的方法之一是:先将形式背景拆分分别进行计算,再进行概念的并置或叠置.属性拓扑可以唯一地表示一个形式背景,形式背景的拆分可以对应于属性拓扑的分解.本节首先给出一种适合概念计算的属性排序算法,然后按照这种属性排列顺序,将属性拓扑自下而上的分解为多个子属性拓扑,完成对形式背景的划分.

2.1 基于Upper-set和Level的属性排序

对于同一个形式背景,其生成的概念格是唯一的,即,生成概念格的结果不受对象和属性排列次序的影响.在概念的计算过程中,判别伪概念和重复概念的过程是非常耗时且繁琐的,属性的排序可以有效地控制概念的生成顺序,从而降低检验伪概念和重复概念过程中的计算复杂度.为了实现属性的排序,定义 Upper-set 和 Level 两种映射.

定义 6(upper-set 映射). 设$\mathbb{K}=(G,M,I)$为形式背景,$\forall m_i\in N\subseteq M$,

$$m_i\to Up(N,m_i)=\{m_i\}\cup\{m|g(m_i)\subset g(m),\forall m\in N\} \tag{6}$$

在形式背景的某非空属性集合 N 下,可以通过公式(6)将该集合 N 中的任意属性 m_i 映射为属性集合,且具有如下性质.

性质 1.

(1) $\forall m_i\in SupAttr,Up(N,m_i)=\{m_i\}$;

(2) $\forall m_i\in M,Parent(m_i)\cap N\subset Up(N,m_i)$;

(3) $\forall m_i\in M,\bigcup_i Up(N,m_i)=N$;

(4) $\forall m_i\in M,g(Up(N,m_i))=g(m_i)$.

证明:

(1) 根据顶层属性的定义:

$$\forall Edge(m_j,m_i)\neq\varnothing,\forall Edge(m_i,m_j)\neq\varnothing,$$

又由 $Edge$ 的定义,必不存在属性 m_j 满足 $g(m_i)\subset g(m_j)$.

因此,$Up(N,m_i)=\{m_i\}\cup\{m|g(m_i)\subset g(m),\forall m\in N\}=\{m_i\}\cup\varnothing=\{m_i\}$ □

(2) 如果 m_i 是顶层属性,$Parent(m_i)\cap N=\varnothing$.根据性质(1):

$$Parent(m_i)\cap N=\varnothing\subset\{m_i\}=Up(N,m_i).$$

如果 m_i 是伴生属性,$\forall m\in Parent(m_i)$均满足 $g(m_i)\subset g(m)$.又由公式(6)得证.

又,属性 m_i 不是顶层属性则必为伴生属性,因此$\forall m_i\in M,Parent(m_i)\cap N=\varnothing\subset Up(N,m_i)$. □

(3) 由公式(6)可知:

$$\{m_i\}\subseteq Up(N,m_i)\subseteq N.$$

两边取并集:

$$\bigcup_i m_i\subseteq\bigcup_i Up(N,m_i)\subseteq N.$$

即, $N\subseteq\bigcup_i Up(N,m_i)\subseteq N$.

因此,$\bigcup_i Up(N,m_i)=N$. □

(4) 设 $m_{jk}\in N$ 是 m_i 的父属性,即 $g(m_i)\subset g(m_{jk})$,显然 $g(m_i)\cap g(m_{jk})=g(m_i)$,因此,

$$\begin{aligned}g(Up(N,m_i))&=g(\{m_i\}\cup(Parent(m_i)\cap N))\\&=g(m_i)\cap g(m_{j1})\cap g(m_{j2})\cap\ldots\cap g(m_{jr})\\&=g(m_i)\cap g(m_{j2})\cap\ldots\cap g(m_{jr})\\&=\cdots\\&=g(m_i)\cap g(m_{jr})=g(m_i).\end{aligned}$$

根据性质 1 可以得出:定义 6 中,经过 Upper-set 映射所得的属性集合是该属性的在定义域 N 下的所有父属性.在属性拓扑中,由于父属性位于子属性之上,表现为一种偏序的覆盖思想,因此,将该映射命名为 Upper-set,并作为自下而上排序的一个基础特征.

定义 7(Level 映射). 设$\mathbb{K}=(G,M,I)$为形式背景,$\forall m\in M$,可以定义从属性 m 到非负整数的映射$\mathcal{L}:m\to N^0$.

$$\mathcal{L}(m)=\begin{cases}0, & Up(M,m)-\{m\}=\varnothing\\ \mathcal{L}_{\max}(Up(M,m)-\{m\})+1, & Up(M,m)-\{m\}\neq\varnothing\end{cases}\tag{7}$$

$\mathcal{L}_{\max}(\cdot)$表示集合中所有元素 Level 的最大值.

根据属性 Upper-set 和 Level 的映射,给出一种适合与形式概念计算的属性排序算法.

算法 1. 属性的自下而上排序算法.

输入:属性拓扑 $AT=(V,Edge)$;

输出:对属性集合 $M=V$ 的排序后的有序属性集合 P_m.

Step 1. $\forall m\in M$,计算 $Up(M,m)$和$\mathcal{L}(m)$.

Step 2. 令 $P_m=\varnothing$.

Step 3. 将属性 m 加入到有序属性集合 P_m 的最后:

$$\{m|\mathcal{L}(m)\geqslant\mathcal{L}(m'),\#Up(M,m)\leqslant\#Up(M,m'),\forall m'\in M-P_M\}\tag{8}$$

Step 4. 若 $M-P_M=\varnothing$,则输出有序属性集合 $P_M=\{m_1,m_2,\ldots,m_i,\ldots,m_n\}$;否则,跳转到 Step 3.

由定义 6、定义 7 和性质 1 可知:集合 $Up(M,m)$包含了属性 m 的所有的父属性,描述了属性间的伴生关系;Level 则描述了属性在属性拓扑中所处的层次.对比文献[28],$Up(M,m)$是属性动度的延伸,当取 $V_G=M-P_M$ 时,$\#Up(M,m)=D_{dy}(m)+1$.算法 1 输出的有序属性集合中,若父属性出现,则它的所有子属性必然位于父属性之前,保留了文献[28]所述排序算法中便于父属性查找的优点.同时,由于加入了选取属性的层次限制,使得在 V_G 发生变化时,动度值并不发生变化,从而使复杂度得到了降低.

以生物和水为例,其属性拓扑如图 1 所示.首先计算每个属性的 Upper-set 和 Level,列于表 3 中.

Table 3 Level and upper-set properties of each attribute

表 3 每个属性的 Upper-set 和 Level 值

m	b	c	d	e	f	g	h	i
$\mathcal{L}(m)$	0	0	0	1	1	0	1	2
$\#Up(M,m)$	1	1	1	3	2	1	2	4

算法最初令有序属性集合 $P_M=\varnothing$.备选属性集合为 $M-P_M=M$,而在备选属性集合中,属性 i 满足公式(8),则将属性 i 加入到有序属性集合中.此时,有序属性集合 $P_M=\{i\}$.重复 Step 3 和 Step 4,最后可以得到所有的有序属性集合 P_M,所得有序属性集合 P_M 中的属性与 M 中的属性相同,只是排列顺序不同.

$$P_M=\{i,e,f,h,d,g,b,c\} \tag{9}$$

对于一个形式背景或属性拓扑,算法 1 的输出结果不是唯一的,例如,有序属性集合$\{i,e,h,f,d,g,b,c\}$是另一种排序结果.这是由于满足限制条件(8)的元素个数可能不唯一.当存在多个属性满足选取条件时,本文采用属性的自然顺序(形式背景中属性所对应的列序号递增的顺序)作为限制条件.不同的排序结果会对计算性能造成一定的影响,但这些不同的排列顺序不会影响到后续算法的讨论.

2.2 属性拓扑的自下而上分解

本节给出的属性拓扑的分解方法可以作为普适的属性拓扑分解方法.由于此处作为输入的有序属性集合中,各元素在属性拓扑中的位置是自下而上的,则称此分解算法为属性拓扑的自下而上分解方法(bottom-up decomposition of attribute topology,简称 BDAT).通过属性拓扑的分解算法,可以将具有 n 个属性的属性拓扑分解成 n 个子属性拓扑,简称为子拓扑.每个子拓扑都是原始拓扑的一部分.在子拓扑中,任意属性 m 都与一个固定的属性 m_0 有关,这个固定的属性 m_0 被称为中心属性,对应的子拓扑称为以 m_0 为中心的子属性拓扑.

算法 2. 属性拓扑的自下而上分解算法(自下而上的子属性拓扑生成算法).

输入:

(1) 属性拓扑 $AT=(V,Edge)$;

(2) 由算法 1 获得的确定的有序属性集合;

(3) 中心属性 m_i;

输出:以属性 m_i 为中心的子属性拓扑 $AT_i=(V_i,E_i)$.

Step 1. 构造两个集合 M_{i0} 和 M_{i1}:

$$M_{i0}=\bigcup_{q=1}^{i-1}m_q \tag{10}$$

$$M_{i1}=M-M_{i0}=\bigcup_{q=i}^{n}m_q \tag{11}$$

Step 2. 选取顶点集合:

$$V_i=\{m\in M_{i1}|Edge(m,m_i)\neq\varnothing\text{或 }Edge(m_i,m)\neq\varnothing\} \tag{12}$$

Step 3. 复制顶点之间的边:

$$E_i(\cdot,\cdot)=Edge(\cdot,\cdot) \tag{13}$$

Step 4. 输出以属性 m_i 为中心的子属性拓扑 $AT_i=(V_i,E_i)$

公式(12)说明:在构造以有序属性集合中第 i 个属性 m_i 为中心的子属性拓扑时,已不再考虑排在第 i 位之前的属性.公式(13)说明:选定顶点之间的关联没有发生任何变化,在子拓扑中和原始拓扑是一样的,这确保了概念的不丢失性.

虽然在实现属性拓扑的自下而上分解算法时需要首先完成对属性的排序,但从算法 2 中可以发现:一旦属性的优先级顺序确定后,每个子属性拓扑可以同时地独立地被生成,因此,算法 2 属性拓扑的自下而上分解方法也被称为自下而上的子属性拓扑生成算法.在每个子属性拓扑中,所有的属性都与中心属性相关,并且所有的边不需要重新计算,因此,算法的计算复杂度较低.

对形式背景为$\mathbb{K}$=(G,M,I)的属性拓扑 AT=(V,$Edge$),其中,V=M={m_1,m_2,…,m_n}.则在每个子属性拓扑中的顶点数不超过#$SupAttr$+1 个.对比文献[23],使用 BDAT 算法分解得到的子拓扑更加细小,更具有可控性.

已知生物和水的属性拓扑如图 1 所示,有序属性集合 P_M={i,e,f,h,d,g,b,c}.下面以属性 d 为例,构造以属性 d 为中心的子属性拓扑.易知 P(d)=5,根据公式(10)和公式(11)可得:

$$M_{i0|i=5}=\bigcup_{q=1}^{4}m_q=\{i,e,f,h\},$$

$$M_{i1|i=5}=\bigcup_{q=5}^{8}m_q=\{d,g,b,c\}.$$

由于 $Edge$(g,d)=∅且 $Edge$(d,g)=∅,根据公式(12)可得:$V_{i|i=5}$={b,c,d}.画出所有的边:$E_{i|i=5}$(·,·)=$Edge$(·,·).最后得到子属性拓扑为 AT_5=(V_5,E_5),如图 2(e)所示.同理可以得到以每个属性为中心的子属性拓扑.此处不再复述生成过程,仅将所有子属性拓扑绘于图 2 中.

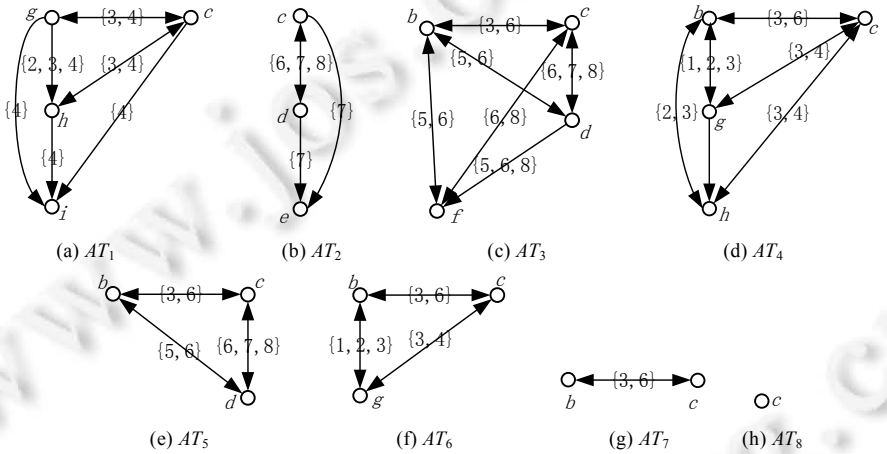


Fig.2 Sub-Attribute-Topology centered by i,e,f,h,d,g,b,c respectively

图 2 分别以属性 i,e,f,h,d,g,b,c 为中心的子属性拓扑

3 BDAT 的子属性拓扑约简

属性拓扑是以属性为顶点的有向图,直观地表示了属性对之间的关联关系和关联强度.结合图理论,在属性拓扑中,概念的生成对应经过内涵中所有对象的路径,而外延则是这些边上权值的交集.第 2 节中,已经将原始形式背景的划分为多个子拓扑,本节将通过讨论属性组合与概念之间的关系,并在不丢失概念的前提下给出一种子拓扑约简算法,使各个子拓扑在概念层面具有各自独立性.

3.1 BDAT与概念之间的关联

设 AT=(V,$Edge$)为属性拓扑,其对应的形式背景为$\mathbb{K}$=(G,M,I).自下而上排序的有序属性集合为 P_M={m_1,m_2,…,m_n}(见算法 1).为便于描述,将所有概念中去除外延为空集和内涵为空集的两个概念所得的集合记$\mathcal{B}$($\mathbb{K}$).由于属性拓扑和形式背景是对应的,因此,$\mathcal{B}$($\mathbb{K}$)也可记为$\mathcal{B}$(AT).在生成每个子属性拓扑时,得到的属性集合和子属性拓扑分别为 M_{i0},M_{i1} 和 AT_i(见算法 2).则$\mathcal{B}$($\mathbb{K}$)可以表示为

$$\mathcal{B}(\mathbb{K})=\mathcal{B}_1(\mathbb{K})+\mathcal{B}_2(\mathbb{K})+\ldots+\mathcal{B}_n(\mathbb{K}) \tag{14}$$

其中,$\mathcal{B}_i(\mathbb{K})$={($A$,$B$)∈$\mathcal{B}$($\mathbb{K}$)|$m_i$∈$B$,$M_{i0}$∩$B$=∅,$A$⊆$G$,$B$⊆$M$}.

定理 3. Up(M_{i1},m_i)=Up(M,m_i)一定是一个概念的外延.

证明:

(1) $Up(M_{i1},m_i)=Up(M,m_i)$是显而易见的;

(2) 若 m_i 为顶层属性,$Up(M_{i1},m_i)=\{m_i\}$,又由顶层属性一定是某个概念的内涵,得证;

(3) 若 m_i 为伴生属性:

$$\begin{aligned} f(g(Up(M_{i1},m_i))) &= f(g(m_i)) \\ &= \{m\in M \mid \forall u\in g(m_i),(u,m)\in I\} \\ &= \{m\in Up(M_{i1},m_i) \mid \forall u\in g(m_i),(u,m)\in I\}\cup\{m\in M-Up(M_{i1},m_i) \mid \forall u\in g(m_i),(u,m)\in I\} \end{aligned}.$$

对于$\forall m\in Up(M_{i1},m_i),g(m_i)\subseteq g(m)$,即$(u,m)\in I,\forall u\in g(m_i),\forall m\in Up(M_{i1},m_i)$

对于$\forall m\in M-Up(M_{i1},m_i),g(m_i)\subseteq g(m)$不成立,即$(u,m)\notin I,\forall m\in M-Up(M_{i1},m_i),\forall u\in g(m_i)$

因此,$f(g(Up(M_{i1},m_i)))=Up(M_{i1},m_i)\cup\varnothing=Up(M_{i1},m_i)$

结合公式(2)、公式(3),证毕. □

命题 1. 若$(A,B)\in\mathcal{B}_i(\mathbb{K})$,则$(A,B)\in\mathcal{B}(AT_i)$.

证明:由于 AT_i 包含了所有与 m_i 相关的属性,并保留了所有属性之间的关联和关联强度,根据定理 2,此命题得证. □

命题 2. 若$(A,B)\in\mathcal{B}(AT_i)$, $\{X \mid g(X)\cap g(m_i)\in 2^{g(m_j)},\forall X\in V_i,\forall m_j\in M_{i0}\}\in B$,则有$(A,B)\notin\mathcal{B}_i(\mathbb{K})$.其中,$2^s$ 表示 S 集合的幂集.

证明:假设 $X\in V_i,m_j\in M_{i0}$ 满足 $g(X)\cap g(m_i)\in 2^{g(m_j)}$, 即 $g(X)\cap g(m_i)\subseteq g(m_j)$.因此,$g(X)\cap g(m_i)=g(X)\cap g(m_i)\cap g(m_j)$

由算法 2 可知,$B\subseteq M_{i1}$.因为 $m_j\in M_{i0},m_j\notin M_{i1}$,对于$\forall(A,B)\in\mathcal{B}(AT_i)$,满足:

$$\{X \mid g(X)\cap g(m_i)\in 2^{g(m_j)},\forall X\in V_i,\forall m_j\in M_{i0}\}\in B,$$

则 $B\subset B\cup m_j\subseteq f(g(B))$.即$(A,B)\notin\mathcal{B}_i(\mathbb{K})$. □

命题 3. 若$(A,B)\in\mathcal{B}(AT_i)$, $\{\bigcup X_i \mid g(\bigcup X_i)\cap g(m_i)\in 2^{g(m_j)},\forall X_i\in V_i,\forall m_j\in M_{i0}\}\in B$, 则有$(A,B)\notin\mathcal{B}_i(\mathbb{K})$.

证明方法与命题 2 类似,此处略过.

命题 4. 若$\mathcal{L}(m_j)>\mathcal{L}(m_i)+1$,则存在 $m_k\in M_{i0}$,满足:

$$\begin{cases} g(m_j)\subset g(m_k)\subset g(m_i) \\ \mathcal{L}(m_k)=\mathcal{L}(m_i)+1 \end{cases}.$$

使得 $2^{g(m_j)}\subset 2^{g(m_k)}$.

3.2 BDAT子拓扑的约简

由命题 1~命题 4 以及定理 1 易知:由 BDAT 所得的子属性拓扑可以在不丢失概念的条件下约简掉冗余信息,以进一步降低子拓扑的规模以简化计算.

算法 3. 子属性拓扑的约简.

输入:

(1) 由算法 2 所得的以属性 m_i 为中心的子属性拓扑 AT_i;

(2) 由算法 2 所得的属性集合 M_{i0} 和 M_{i1};

(3) 由算法 1 所得的 $Up(M_{i1},m_i)$;

输出:以属性 m_i 为中心的约简子属性拓扑 $\overline{AT_i}$.

Step 1. 若 $V_i-Up(M_{i1},m_i)=\varnothing$,则输出 $\overline{AT_i}=(\varnothing,\varnothing)$,算法退出;

Step 2. 约简属性拓扑中的顶点属性:

(1) 令 $S_i=\{Up(M_{i1},m_i)\cup m \mid \forall m\in V_i-Up(M_{i1},m_i)\}$;

(2) $\forall a,b\in S$ 且 $g(a)=g(b)$,则令 $S_i=S_i-\{a\}-\{b\}+\{a\cup b\}$;

(3) 令 $\overline{V_i}=S_i-\{m \mid g(m)\in 2^{g(m_k)},\forall m\in S_i,\forall m_k\in\overline{M_{i0}}\}$,其中:

$$\overline{M_{i0}}=\{m\mid \mathcal{L}(m)\leqslant \mathcal{L}(m_i)+1, E(m_i,m)\neq\varnothing, \forall m\in M_{i0}\};$$

Step 3. 约简属性拓扑中的边.

(1) 根据 Step 2 属性合并规则计算新的边上的权值 $\overline{E}_i$;

(2) $\forall m_i,m_j\cup\overline{V}_i$,若 $\overline{E_i(m_i,m_j)}\in 2^{g(m_k)}, m_k\in\overline{M_{i0}}$,则标记 $\overline{E_i(m_i,m_j)}=\varnothing$;

注:此处标记 $\overline{E_i(m_i,m_j)}=\varnothing$,意味着属性 m_i 和 m_j 不能同时出现;

Step 4. 输出以属性 m_i 为中心的约简子属性拓扑 $\overline{AT_i}$,算法结束.

命题 5. 若 $(A,B)\in\mathcal{B}(\overline{AT_i})$,则$(A,B)\in\mathcal{B}_i(\mathbb{K})-(g(m_i),Up(M_{i1},m_i))$.

证明:

(1) 对于 $\forall(A,B)\in\mathcal{B}(\overline{AT_i})$,假设 $(A,B)\in\mathcal{B}(\overline{AT_j})$,$j>i$.由于 $m_i\in B, m_i\notin B_1$,因此 $\forall(A,B)\in\mathcal{B}(\overline{AT_j})$;

(2) 对于 $\forall(A,B)\in\mathcal{B}(\overline{AT_i})$,假设 $(A,B_0)\in\mathcal{B}(\overline{AT_k})$,$k<i$.由命题 4 可知 $A\notin 2^{g(m_k)}$,因此, $\forall(A,B_2)\in\mathcal{B}(\overline{AT_k})$.

结合命题 1 至命题 4,$(A,B)\in\mathcal{B}_i(\mathbb{K})-(g(m_i),Up(M_{i1},m_i))$. □

BDAT 子拓扑约简的一个优点是:可以在不丢失概念的条件下降低子拓扑的规模,使得概念计算速度进一步提升.由于属性拓扑描述的是属性对之间的关系,相当于对形式背景的升维操作,可以更加容易地去除与求取形式概念.结合命题 1~命题 5 可知,这种约简算法的另一个优点是,各个子拓扑中求取的形式概念在原始的形式背景中仍然满足概念的条件,即,无需进行去伪操作.又由于特定的分解方法,使得子拓扑中求取的形式概念不会产生重复.

约简之后的各个子属性拓扑结构去除了概念层次的耦合,即:所得结果可以在各个并行线程中独立的完成各自的概念计算任务,而不存在线程之间的耦合,这使得 BDAT 的概念计算可以逃脱典型并行计算中反复串并交替的逆木桶效应.

接下来用以属性 h 为中心的子属性拓扑 AT_4 为例,计算约简子属性拓扑.根据上面的例子和定义可得:

$$V_4=\{b,c,g,h\},$$
$$Up(M_{41},h)=\{g,h\}.$$

由于 $V_4\neq Up(M_{41},h)$,因此,$S_h=\{gh\cup b, gh\cup c\}$,其中,

$$g(gh\cup b)=\{2,3\},$$
$$g(gh\cup c)=\{3,4\},$$

则 $\overline{M_{40}}=\{i\}$;

由于 $g(gh\cup b)\notin 2^{g(i)}, g(gh\cup c)\notin 2^{g(i)}$,则:

$$\overline{V}_4=\{bgh,cgh\},$$
$$\overline{E}_4(bgh,cgh)=\overline{E}_4(cgh,bgh)=\{3\}.$$

此时,得到以属性 h 为中心的约简子属性拓扑 $\overline{AT_4}=(\overline{V}_4,\overline{E}_4)$.

由于在子属性拓扑中,$\forall m\in Up(M_{i1},m_i)-\{m_i\}$ 都必然存在一条单向指入中心属性的边,因此算法 3 中的 Step 1 以及 Step 2.1 可以通过拓扑的可视化操作完成,如图 3 所示.

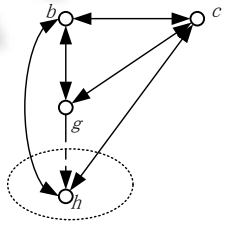


Fig.3 The visualization description for Step 1 and Step 2.1 in Algorithm 3

图 3 算法 3 Step 1 和 Step 2.1 中,公式与拓扑可视化描述

由于子属性拓扑的约简过程是类似的,此处不再复述.

在生物和水的 8 个约简子属性拓扑中,$\overline{AT_1}=\overline{AT_2}=\overline{AT_8}=(\varnothing,\varnothing)$,其他 5 个约简子拓扑如图 4 所示.

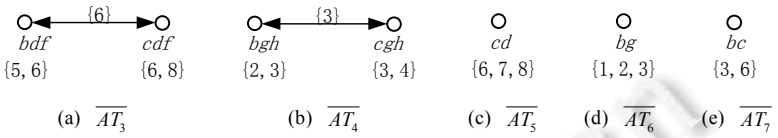


Fig.4 Optimized Sub-Attribute-Topologies in Table 2(b)

图 4 表 2(b)中所有的非空约简子属性拓扑

从生物和水的示例中可知:原始的形式背景(见表 2)规模为 64=8*8,经过 BDAT 分解和约减后的形式背景总规模为 15=2×3+2×3+1×1+1×1+1×1.

4 基于 BDAT 的并行概念计算算法

由 BDAT 算法以及对 BDAT 所得的子拓扑进行约简的算法可知:这些算法在考虑以某个属性为中心的条件下,各个子属性拓扑的生成和约简可以独立无交互地并行完成.同时,讨论了 BDAT 与概念之间的关系.本节中,将依托于上文提出的算法,给出一种基于 BDAT 的并行概念计算算法.

算法 4. 基于 BDAT 的并行概念计算算法.

输入:属性拓扑 $AT=(V,Edge)$;

输出:所有的形式概念.

Step 1. 根据算法 1,对所有的属性进行排序:$P_M=\{m_1,m_2,\ldots,m_n\}$

Step 2. 对于每个属性 m_i:

(1) 根据算法 2,生成子属性拓扑 AT_i;

(2) 根据算法 3,对子属性拓扑 AT_i 进行约简,得到 $\overline{AT_i}$;

(3) 计算 $\mathcal{B}(\overline{AT_i})$;

(4) 返回概念集合 $\mathcal{B}(\overline{AT_i})\cup(g(m_i),Up(M_{i1},m_i))$;

Step 3. 对每个返回的概念集合,得到所有的概念:

$$\mathfrak{B}(\mathbb{K})=(\varnothing,M)\cup(G,\varnothing)\cup\left(\bigcup_i(\mathcal{B}(\overline{AT_i})\cup(g(m_i),Up(M_{i1},m_i)))\right).$$

由算法可知,只需将每个子拓扑中的概念集合取并即可得到所有的概念,这依赖于上文所述的分解和约简特性.正如在第 3 节分析的,实现算法中步骤 2(3)概念计算依赖于属性拓扑对形式背景中的升维操作,需要由约简子属性拓扑计算出所有的形式概念.因此,这步计算可以使用任意的基于属性拓扑的概念计算方法;也可以只使用递归反复调用 BDAT 算法,直到得到的约简子属性拓扑形成完全图,或孤立的属性顶点.同样,可以先递归调用 BDAT 算法,直到到达预先指定的最大递归调用层数 $RL_{\max}$,以获得与计算性能相匹配的独立支路数,然后对每条支路使用其他的基于属性拓扑的概念计算方法进行计算.完全递归的方法可以被视为混合算法中的最大递归调用层数 $RL_{\max}=INFINITE$.如果要使用其他形式概念方法,则需要将约简操作影响的属性对进行标记,以便做额外的处理.

算法 4 的流程图如图 5 所示.

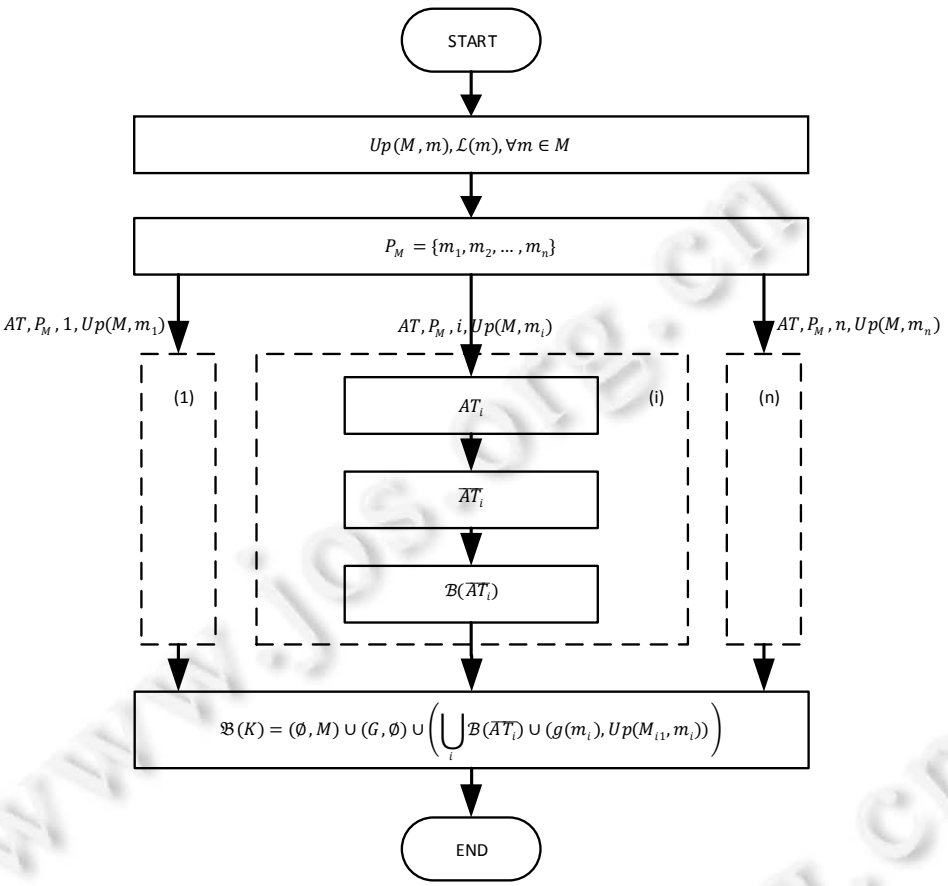


Fig.5 The flowchart of the Algorithm 4

图 5 算法 4 的流程图

对于并行的概念计算算法来说,除了考虑形式背景的拆分和概念的合并之外,使得各部分的负载得到均衡也是一个良好的并行计算算法应该具有的特性.一般来说,在将所有概念计算完成之前,形式概念的分布情况是不能预知的,因此,很难将复杂度平均的分发给各个线程[10].而在 BDAT 分解过程中,属性拓扑的分解和子属性拓扑的约简算法限制了最终拓扑的大小.对于以伴生属性为中心的约简子属性拓扑中,其中的属性顶点数不会超过#*SupAttr*;对于以顶层属性为中心的约简子属性拓扑中,其中的属性顶点数必小于#*SupAttr*,并且以有序属性集合的顺序,其约简子属性拓扑的大小是越来越小的.考虑到形式概念数大致与形式背景的大小成指数型关系增长[29],因此,我们可以大致地认为,约简子属性拓扑的概念计算复杂度随着顶层属性的排序逐渐递减.则在线程有限的情况下,按照有序属性集合中的顺序,首先计算以伴生属性为中心的子属性拓扑,然后通过适当的分配顶层属性为中心的子属性拓扑,可以使不同线程的负载尽可能均衡,这使得 BDAT 算法更适合并行计算.

本节中,使用生物和水的属性拓扑作为数据源.在第 3.2 节中,已经求解出以每个属性为中心的各个约简子属性拓扑(如图 4 所示),只需计算出每个约简子属性拓扑的所有概念,再根据定理 3,由 8 个中心属性构成的 8 个概念,通过简单的并集运算,即可得到没有重复概念和伪概念的全体形式概念集合(见表 4).表 4 中列出了生物和水的形式背景下使用 BDAT 得到的全体概念,所得概念是正确的且没有遗漏和丢失,其中,第 12 个概念的外延为{6,7,8}={芦苇,豆,玉米},内涵为{*c*,*d*}={在陆地上生活,有叶绿素},在当前的形式背景中,则可以将该概念理解为陆生植物.同样的,在该背景下的两栖生物对应于第 16 个概念,其外延为{3,6}={青蛙,芦苇},内涵为{*b*,*c*}={在水中生活,在陆地上生活}.

Table 4 All the concepts of Living Beings and Water computed by BDAT algorithm

表 4 使用 BDAT 算法计算出的生物和水的形式概念

编号	外延	内涵	来源	编号	外延	内涵	来源
1	4	*c*,*g*,*h*,*i*	中心属性 *i*	11	5,6,7,8	*d*	中心属性 *d*
2	7	*c*,*d*,*e*	中心属性 *e*	12	6,7,8	*c*,*d*	顶层属性 $\overline{AT_5}$
3	5,6,8	*d*,*f*	中心属性 *f*	13	1,2,3,4	*g*	中心属性 *g*
4	6,8	*c*,*d*,*f*	顶层属性 $\overline{AT_3}$	14	1,2,3	*b*,*g*	顶层属性 $\overline{AT_6}$
5	5,6	*b*,*d*,*f*	顶层属性 $\overline{AT_3}$	15	1,2,3,5,6	*b*	中心属性 *b*
6	6	*b*,*c*,*d*,*f*	生成概念 $\overline{AT_3}$	16	3,6	*b*,*c*	顶层属性 $\overline{AT_7}$
7	2,3,4	*g*,*h*	中心属性 *h*	17	3,4,6,7,8	*c*	中心属性 *c*
8	3,4	*c*,*g*,*h*	顶层属性 $\overline{AT_4}$	18	∅	*b*,*c*,*d*,*e*,*f*,*g*,*h*,*i*	全局概念
9	2,3	*b*,*g*,*h*	顶层属性 $\overline{AT_4}$	19	1,2,3,4,5,6,7,8	∅	全局概念
10	3	*b*,*c*,*g*,*h*	生成概念 $\overline{AT_4}$				

5 实验结果与分析

为了验证本文提出的并行形式概念计算算法的正确性,并评估并行概念计算算法的效率,实验中选取了 5 个数据集,除了典型的形式背景生物和水(living beings and water)之外,还选取了 4 组来自 UCI 的数据集: Balance scale[30],Tic tac toc[31],Mushroom[32]和 Nursery[33].这些数据集大多是多值的,因此实验前需要首先将它们转化为二值背景[1],然后经过预处理去除冗余的对象和属性,得到净化后的二值形式背景,实验中使用的各个二值形式背景的基本信息列于表 5 中.

Table 5 Information of formal contexts in discussion

表 5 实验中使用的二值形式背景的基本信息

编号	名称	对象数	属性数	复杂度	概念数
1	Living beings and water	8	8	0.41	19
2	Balance scale	625	23	0.22	2 106
3	Tic tac toc	958	28	0.34	52 717
4	Mushroom	2 744	79	0.20	47 458
5	Nursery	12 960	27	0.30	115 201

本节实验在相同的硬件和软件环境下(见表 6),测试 3 种不同的形式概念计算算法,这 3 种不同的算法是:

(1) Krajca Petr 提出的并行递归算法,PcbO;

(2) 本文提出的基于 BDAT 的并行计算算法,工作在单线程递归模式,本实验中简写为 BDAT/s;

(3) 本文提出的基于 BDAT 的并行计算算法,工作在多线程递归模式,本实验中简写为 BDAT/p.

Table 6 The hardware environment and software environment

表 6 实验的硬件和软件环境

名称	型号	核心参数
CPU/L2	Intel Core i3-3220	3.30GHz/512K
内存	Kingston	4GB/1600MHz
硬盘	Seagate	500G/7200RPM
操作系统	Windows 10	x64
编译平台	Visual Studio	2013

PCbO 是著名的形式概念计算算法之一,也被认为是最快速的算法之一[34],被广泛应用在很多领域[35,36].由于在直观图、属性偏序图等图方法中未见并行计算的相关论文,本文与第 1 种并行算法 PCbo 进行对比.PCbO 的源码来自 sourceforge 中提供的符合 GPL V2 标准的 C 语言代码,实验中将其作为本文算法的对比算法,在开源协议条款的允许下,在代码的入口和出口添加了必要的时间监测,以获取该算法的运行时间.算法执行中设置的参数均设置为作者建议的参数.

(1) 线程数设置为 4,因为代码文档中指出:通常情况下,线程数设置为 CPU 内核的 2 倍~3 倍;

(2) 其他的参数均保留其默认值.

为了更好地研究本文提出的算法,使用 C 语言编写设计了一个测试性的基于 BDAT 概念计算程序.对于属性拓扑中的完全连接图而言,任意属性的组合都将构成一个形式概念的内涵,无需进行递归调用.受限于 C 矩阵运算的复杂性,用于测试的 BDAT 算法程序在递归分解的过程中没有充分利用完全连接图的优势特征,而是将每一个拓扑分解成完全隔离的节点.因此,测试数据显示的是 BDAT 算法中最坏的情况.这个程序不但能帮我们验证本文提出的形式概念并行计算结果的正确性,同时也将通过代码监视程序运行的时间,即计算所有概念所消耗的时间,以评估本文提出算法的计算性能.

在进行实验时,考虑到算法的运行时间将会受到系统中其他进程的影响,为了减小实验的偶然误差,提高计算时间记录的准确性,每种算法分别对每个数据集进行了 10 次测试,最后对这 10 次记录的时间监视数据求取平均值,作为最终的算法运行时间(见表 7).通过实验结果比对,这 3 种算法输出的概念集合是一致的,验证了本文提出的基于 BDAT 的形式概念并行计算算法的正确性.

Table 7 The time costs in computing different formal context by dissimilar algorithms

表 7 输入不同形式背景时不同算法计算所有形式概念所消耗的时间

编号	形式背景	PCbO 并行耗时(ms)	BDAT 串行耗时(ms)	BDAT 并行耗时(ms)	BDAT 并串比(%)	BDAT 递归次数	速度提升
1	Living beings and water	0.0	0.1	0.5	500	3	–
2	Balance scale	6.4	5.3	4.1	77.36	4	35.94%
3	Tic tac toc	163.4	193.5	113.4	58.60	9	30.60%
4	Mushroom	229.0	260.3	143.8	55.24	10	37.21%
5	Nursery	360.3	1 275.9	599.1	46.96	8	–

在表 7 中,第 3 列~第 5 列分别记录了在 3 种不同算法下,计算形式概念消耗的时间,作为算法性能对比的基础数据;第 6 列的并串比描述了 BDAT 并行算法与串行算法时间的比值,可以清晰看出并行算法提升的速度;第 8 列(最后一列)清晰地描述了 BDAT 并行算法较 PCbO 并行算法提升的效果;第 7 列记录了使用 BDAT 算法完成概念计算过程中,递归调用的最大次数.正如前文所述,算法中没有利用完全图特性,因此本文算法理论上会小于等于实验记录中的递归次数,即,表 7 中记录的 BDAT 实验数据是算法的最坏情形.

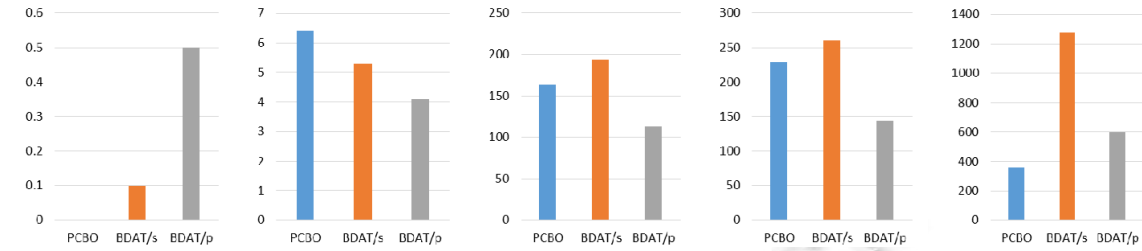
结合实验数据进行分析,可以得出以下结论.

(1) 对于较小的形式背景,如生物和水的背景,BDAT 算法慢于 PCBO,且表现出串行优于并行的现象(如图 6(a)所示).出现这种现象的根本原因是:虽然经过 BDAT 的分解和约简使子拓扑的规模得到降低(见第 3 节),但以此减少的概念计算时间并未抵消属性拓扑的构建、分解、约简和线程操作所占用的时间;

(2) 当形式背景规模较大且对象数较少时,如第 2~4 个形式背景,由于属性拓扑分解和约简算法的相互独立性,加之属性拓扑对形式背景的升维操作,本文提出的 BDAT 并行算法比 PCbO 并行算法速度提升 30%左右(如图 6(b)~图 6(d)所示).类似的,我们可以使用形式背景的转置,使本文的并行概念计算算法适应规模较大且属性数较小的形式背景;

(3) 对于 Nursery 背景,其规模较大对象数较多但只有 27 个属性,每个子拓扑的分解程度不够细小,每个线程需要处理大量的数据,因此,BDAT 算法并没有表现出优势(如图 6(e)所示).若将形式背景的转置后计算形式概念,则需要大量的线程,每个线程计算量较为细小,受到实验环境并行执行线程数的限制,没有给出并行线程数足够情形下的实验结果,因此,本文算法对此类背景的适应性有待进一步研究;

(4) 随着形式背景规模的增大,BDAT 并串比如图 7 所示,其最后将近似趋于一个常数,这个常数与 CPU 的核数有关.



(a) Living beings and water (b) Blance Scale (c) Tic tac toc (d) Mushroom (e) Nursery

Fig.6 The average time costs (ms) by different algorithms in different contexts

图 6 在不同形式背景下不同算法的计算平均时间(ms)

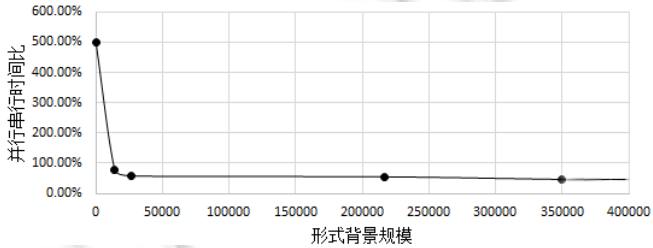


Fig.7 The chart showing the ratio of parallel time complexity and sequential with context size

图 7 并行串行时间复杂度比值随形式背景规模的曲线示意图

假定形式背景的规模相同,属性拓扑的结构将是影响计算性能的关键因素,现在已知的是顶层属性的个数将限制子属性拓扑的规模(见第 4 节).在形式背景规模达到或超过系统底层操作的影响可忽略的规模时,随着顶层属性所占比例的递增,BDAT 计算性能的曲线将呈现先上升后下降的趋势.为了讨论顶层对性能曲线的影响,设伴生属性的各个 Level 呈现随机分布,且当样本数足够多时,各 Level 层的伴生属性数据呈现均匀分布,此时,若顶层属性所占比例极小,则由于该算法是自下而上进行分解约简,虽然所得各个子拓扑规模较小,由于顶点的重复出现的概率很大,此时计算效率较低.随着顶层属性数所占比例的增大,计算效率将逐步加快.若顶层所占比率继续增大,子拓扑的平均规模也将由于顶层属性的增多而增大,此时,各个线程计算的耗时也将增大.

6 结束语

本文利用属性拓扑中对于属性间耦合表示的特性,提出了自下而上的拓扑分解方法,配合子拓扑的约简实现了各子拓扑间的完全并行运算.该方法解决了原有并行概念计算中并行算法串并交替问题,并从理论和实验中验证了本文方法的正确性,为快速概念分析提供了基础性工具.通过实验分析可知:与经典并行概念分析算法相比,本文方法具有快速、准确的特点.本文算法不但丰富了属性拓扑理论,为属性拓扑更广泛的研究提供了新的思路,使得大规模形式背景的概念分析成为了可能,同时也可应用于认知计算等相关的领域,作为基础算法为高层算法提供数据计算服务.

由于属性拓扑可以理解为广义的有向加权图,在下一步的研究中将利用属性拓扑作为媒介,将分布式图计算与形式概念计算连接起来,应用于机器学习等领域.

References:

[1] Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations. Springer Science & Business Media, 2012.

[2] Bhatnagar R, Kumar L. An efficient map-reduce algorithm for computing formal concepts from binary data. In: Proc. of the 2015 IEEE Int'l Conf. on Big Data. IEEE, 2015. 1519–1528.

[3] Sun XB, Li Y, Li BX, Wen WZ. A survey of using formal concept analysis for software maintenance. Acta Electronica Sinica, 2015,43(7): 1399−1406 (in Chinese with English abstract).

[4] Shao MW, Yang HZ, Wu WZ. Knowledge reduction in formal fuzzy contexts. Knowledge-Based Systems, 2014,73:265−275.

[5] Li J, Mei C, Wang J, Zhang X. Rule-Preserved object compression in formal decision contexts using concept lattices. Knowledge-Based Systems, 2014,71:435−445.

[6] Li X, Luo J, Shi A. An improved data mining algorithm based on concept lat-tice. In: Proc. of the 2nd Int'l Conf. on Computer Science and Electronics Engineering. Atlantis Press, 2013.

[7] Kaytoue M, Codocedo V, Buzmakov A, Baixeries J, Kuznetsov SO, Napoli A. Pattern structures and concept lattices for data mining and knowledge processing. In: Proc. of the Machine Learning and Knowledge Discovery in Databases. Springer Int'l Publishing, 2015. 227−231.

[8] Singh PK, Kumar CA, Li J. Knowledge representation using interval-valued fuzzy formal concept lattice. Soft Computing, 2015, 19(1):1−18.

[9] Kengue JFD, Valtchev P, Djamegni CT. A parallel algorithm for lattice construction. In: Proc. of the Formal Concept Analysis. Berlin, Heidelberg: Springer-Verlag, 2005. 249−264.

[10] Krajca P, Outrata J, Vychodil V. Parallel recursive algorithm for FCA. In: Proc. of the CLA 2008. 2008. 71−82.

[11] Krajca P, Vychodil V. Distributed algorithm for computing formal concepts using map-reduce framework. In: Proc. of the Advances in Intelligent Data Analysis VIII. Berlin, Heidelberg: Springer-Verlag, 2009. 333−344.

[12] Dong H, Ma Y, Gong X. A new parallel algorithm for construction of concept lattice. Journal of Frontiers of Computer Science and Technology, 2008,2(6):651−657 (in Chinese with English abstract).

[13] Ma C. A parallel constructing algorithm based on dividing of closure system for concept lattice. China Management Informationization, 2009,12(21):20−24 (in Chinese with English abstract).

[14] Ma F, Zeng ZY, Yu JK. Research on vertically combine method of distributed concept lattices. Computer Engineering and Applications, 2011,47(34):68−63 (in Chinese with English abstract).

[15] Zhi HL. Extended model of formal concept analysis oriented for heterogeneous data analysis. Acta Electronica Sinica, 2013,41(12):2451−2455 (in Chinese with English abstract).

[16] Zhang Z, Chai YM, Wang LM, Fan M. A parallel algorithm generating fuzzy formal concepts. Pattern Recognition & Artificial Intelligence, 2013,26(3):260−269 (in Chinese with English abstract).

[17] Zhang Z, Du J, Wang LM. Load balance-based algorithm for parallelly generating fuzzy formal concepts. Control and Decision, 2014,29(11):1935−1942 (in Chinese with English abstract).

[18] Zhang T, Wei X, Hong W, Luan J. Attribute characteristics analysis and compare between AT and APOSD. In: Proc. of the 2015 5th Int'l Conf. on Instrumentation and Measurement, Computer, Communication and Control (IMCCC). IEEE, 2015. 947−951.

[19] Xu WH, Li JH, Wei L, Zhang T. Formal Concept Analysis: Theory and Application. Beijing: Science Press, 2016 (in Chinese).

[20] Zhang T, Wei X, Hong W, Li S. Transformation properties in attribute topology and attribute partial ordered structure diagram. In: Proc. of the 5th Int'l Conf. on Instrumentation and Measurement, Computer, Communication and Control (IMCCC). IEEE, 2015. 1085−1089.

[21] Zhang T, Li H, Wei X, Li L. Attribute topology and concept lattice bridged by concept tree. In: Proc. of the 5th Int'l Conf. on Instrumentation and Measurement, Computer, Communication and Control (IMCCC). IEEE, 2015. 1037−1041.

[22] Zhang T, Li H, Hong W, Yuan X, Wei X. Deep first formal concept search. Scientific World Journal, 2014,2014:275679−275679.

[23] Zhang T, Ren HL, Hong WX, Li H. The visualizing calculation of formal concept that based on the attribute topologies. Acta Electronica Sinica, 2014,42(5):925−932 (in Chinese with English abstract).

[24] Zhang T, Ren H, Wang X. A calculation of formal concept by attribute topology. ICIC Express Letters, Part B: Applications, An Int'l Journal of Research & Surveys, 2013,4:793−800. http://ci.nii.ac.jp/naid/40019602751

[25] Li G, Ma YC, Zhang T, Hong WX. Formal concept construction algorithm based on attribute topology. System Engineering—Theory & Practice, 2015,35(1):254−259 (in Chinese with English abstract).

[26] Wei L, Wan Q. Granular transformation and irreducible elements judgment based on pictorial diagrams. IEEE Trans. on Cybernetics, 2016,46(2):380−387.

[27] Hong WX, Li SX, Yu JP. A new approach of generation of structured partial ordered attribute diagram based on covering. ICIC Express Letters, Part B: Applications, 2015,6(4):1049−1054.

[28] Bai DH, Zhang T, Wei XY. Attributes-sorting algorithm based on attribute degree. Computer Engineering and Applications, 2015 (in Chinese with English abstract). http://www.cnki.net/kcms/detail/11.2127.TP.20150929.1045.018.html

[29] Carpineto C, Romano G. Concept Data Analysis: Theory and Applications. John Wiley & Sons, 2004.

[30] Gharehchopogh FS, Khaze SR. Data mining application for cyber space users tendency in blog writing: A case study. Int'l Journal of Computer Applications, 2013,47(18):40−46.

[31] Klahr D, Siegler RS. The representation of children's knowledge. Advances in Child Development and Behavior, 1978,12:62−116.

[32] Lincoff GH. The Audubon Society Field Guide to North American Mushrooms. Knopf: Distributed by Random House, 1981.

[33] Zupan B, Bohanec M, Bratko I, Demsar J. Machine learning by function decomposition. In: Proc. of the ICML. 1997. 421−429.

[34] Strok F, Neznanov A. Comparing and analyzing the computational complexity of FCA algorithms. In: Proc. of the 2010 Annual Research Conf. of the South African Institute of Computer Scientists and Information Technologists. ACM Press, 2010. 417−420.

[35] Kirchberg M, Leonardi E, Tan YS, Link S, Ko RKL, Lee BS. Formal concept discovery in semantic Web data. In: Proc. of the Formal Concept Analysis. Berlin, Heidelberg: Springer-Verlag, 2012. 164−179.

[36] Wray T, Eklund P. Using Formal Concept Analysis to Create Pathways through Museum Collections. Faculty of Engineering & Information Sciences Papers, 2014.

附中文参考文献:

[3] 孙小兵,李云,李必信,文万志.形式概念分析在软件维护中的应用综述.电子学报,2015,43(7):1399−1406.

[12] 董辉,马垣,宫玺.一种新的概念格并行构造算法.计算机科学与探索,2008,2(6):651−657.

[13] 马驰.基于闭包系统划分的概念格并行构造算法.中国管理信息化,2009,12(21):20−24.

[14] 马冯,曾志勇,余建坤.分布式概念格的纵向合并方法研究.计算机工程与应用,2011,47(34):68−71.

[15] 智慧来.面向异构数据分析的形式概念分析扩展模型.电子学报,2013,41(12):2451−2455.

[16] 张卓,柴玉梅,王黎明,范明.模糊形式概念并行构造算法.模式识别与人工智能,2013,26(3):260−269.

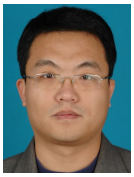
[17] 张卓,杜鹃,王黎明.基于负载均衡的模糊概念并行构造算法.控制与决策,2014,29(11):1935−1942.

[19] 徐伟华,李金海,魏玲,张涛.形式概念分析理论与应用.北京:科学出版社,2016.

[23] 张涛,任宏雷,洪文学,李慧.基于属性拓扑的可视化形式概念计算.电子学报,2014,42(5):925−932.

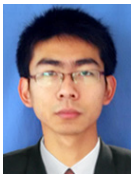
[25] 李刚,马彦超,张涛,洪文学.基于属性拓扑图的形式概念构造算法.系统工程理论与实践,2015,35(1):254−259.

[28] 白冬辉,张涛,魏昕宇.基于属性度的属性排序算法.计算机工程与应用,2015. http://www.cnki.net/kcms/detail/11.2127.TP.20150929.1045.018.html



张涛(1979－),男,河北唐山人,博士,副教授,CCF 专业会员,主要研究领域为形式概念分析,认知计算,医学信号处理.

李慧(1989－),女,硕士,主要研究领域为形式概念分析,属性拓扑.



白冬辉(1990－),男,硕士,主要研究领域为形式概念分析,认知计算.