基于搜索空间划分的并行概念生成算法*)

齐 红 刘大有 胡成全 卢 明 赵 亮
(吉林大学 计算机科学与技术学院 长春130012)

摘 要 概念格作为形式概念分析理论中的核心数据结构,在机器学习、数据挖掘和知识发现、信息检索等领域得到了广泛的应用。概念格的构造在其应用过程中是一个主要问题。本文提出了一种基于搜索空间划分的并行概念生成算法,它对整个闭包搜索空间进行划分,并引入一种有效的测试方法,只搜索那些能生成正规闭包的子搜索空间,从而有效提高搜索效率;同时,在计算闭包过程中保存一些必要的中间结果,用来提高闭包运算速度;由于所有子搜索空间相对独立,因此很容易得到一个并行的概念生成算法。

关键词 概念格,并行算法,搜索空间,闭包系统

A Parallel Algorithm Based on Search Space Partition for Generating Concepts

QI Hong LIU Da-You HU Cheng-Quan LU Ming ZHAO Liang
(College of Computer Science and Technology, Jilin University, Changchun 130012)

Abstract Concept Lattice, the core data structure in Formal Concept Analysis, has been used widely in machine learning, data mining and knowledge discovery, information retrieval, etc. The main difficulty with concept lattice-based system comes from the lattice construction itself. In this paper, a parallel algorithm based on the closure search space partition for computing concepts is proposed. This algorithm divides the closure search space into several subspaces in accordance with criterions prescribed ahead and introduces an efficient scheme to recognize the valid ones, in which the searching for closures is bounded. An intermediate structure is employed to judge the validity of a subspace and compute closures more efficiently. Since the searching in subspaces are independent tasks, a parallel algorithm based on search space partition can be directly reached.

Keywords Concept lattice, Parallel algorithm, Search space, Closure system

1 引言

概念格作为形式概念分析理论中的核心数据结构,被广泛应用于机器学习、数据挖掘和知识发现、信息检索等领域[2,6,11,12]。应用概念格的过程中,概念格的构造效率始终是制约概念格应用的主要问题。

随着包含多个CPU的计算机结构的发展,构造概念格的并行算法成为一个新的研究课题。Njiwoua P 和 Mephu Nguifo E[9]提出了一个基于 Bordat 算法[1]、多项式时间的并行算法。Fu H 和 Mephu Nguifo E[3]最近又提出了一个基于 Ganter 算法的并行概念格构造算法。

本文提出的并行算法是基于闭包运算的。基于闭包运算的概念格构造算法的基本思想是计算对象集(或属性集)的所有子集的闭包,并通过正规测试保证每个闭包只生成一次。由 Ganter 提出的 NextClosure 算法[4]是其中最著名的一个,它只计算对象集的某些子集的闭包,并使用一个高效的正规测试方法,不需要保存已生成的概念,所以只需要较少的存储空间。另一个基于闭包运算的算法是 CbO(Close by One)算法[7],它使用类似的正规测试和集合选择方法,并引入一个中间结构来更高效地生成概念。Norris 算法[10]基本上是 CbO 算法的增量版本。研究发现,对于大而稠密的形式背景,上述三个算法表现出良好的性能,在其他情况下,这类算法运行效果不是很好[8]。主要原因在于,首先,这些算法在每个步骤生成某个集合的闭包,如果该闭包满足正规性测试,那么可由它生成一个新概念,否则不能生成概念,这时计算该闭包花费的所有步骤都是无用的;其次,闭包运算本身要花费大量时间。

本文提出了一种基于搜索空间划分的并行概念生成算法 PCG,它对整个闭包搜索空间进行划分,并引入一种有效的测试方法,只搜索那些能生成正规闭包的子搜索空间,从而有效提高搜索效率;同时,在计算闭包过程中保存一些必要的中间结果,用来提高闭包运算速度。由于在子搜索空间中的搜索是独立的,因此很容易得到并行算法。本文第2节简要介绍有关概念格的基本概念;第3节讨论基于搜索空间划分的概念生成的基本原理;第4节给出并行概念生成算法 PCG;第5节分析算法的复杂性;最后总结全文。

2 基本概念

本节简要介绍概念格的基本概念,关于概念格的比较详尽的形式化描述可参考文[5]。

在形式概念分析中,形式背景通常定义为一个三元组 $K=(G, M, I)$,其中,G 是对象集合,M 是属性集合,$I\subseteq G\times M$ 是 G 与 M 之间的一个二元关系。若 $(g, m)\in I$,读作"对象 g 具有属性 m"。

对 $A\subseteq G$,定义 $A'=\{m\in M \mid \forall\ g\in A, (g, m)\in I\}$;对 $B\subseteq M$,定义 $B'=\{g\in G\mid \forall\ m\in B, (g, m)\in I\}$。

背景 K 上的一个形式概念定义为一个二元组 (A, B),满足 $A\subseteq G$, $B\subseteq M$, $A'=B$, $B'=A$。其中 A 称为概念 (A, B) 的外延,B 称为概念 (A, B) 的内涵。形式背景 K 上的任何两个概念 (A, B) 和 (C, D),如果 $B\subseteq D$,则称 (A, B) 是 (C, D) 的超概念,(C, D) 称为 (A, B) 的子概念,记为 $(C, D)\leqslant(A,$

*)本课题得到国家自然科学基金(项目编号60173006)、国家高技术研究发展计划(项目编号2003AA118020)、吉林省科技发展计划重大项目(吉科合字20020303)、吉林大学符号计算与知识工程教育部重点实验室资助。齐 红 博士研究生,讲师,目前研究兴趣为 KDD、Rough Sets 和 Concept Lattice。刘大有 教授,博士生导师,从事人工智能、数据挖掘和计算机应用等方面的研究。

B)。通过这种序关系，得到一个有序集$\underline{B(K)}=(B(K), \leqslant)$，其中$B(K)$为形式背景$K$上的所有形式概念的集合，这是一个完备格，称为形式背景K的概念格。

表1是一个例子形式背景k，其对应的概念格在图1中给出。

表1　一个例子形式背景k

	M								
I	a	b	c	d	e	f	g	h	i
1	1	0	1	0	0	1	0	1	0
2	1	0	1	0	0	0	1	0	1
G 3	1	0	0	1	0	0	1	0	1
4	0	1	1	0	0	1	0	1	0
5	0	1	0	0	1	0	1	0	0

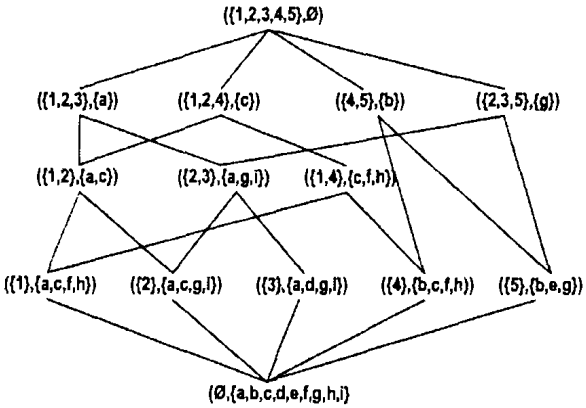

图1　形式背景k对应的概念格

3　基于搜索空间划分的概念生成

形式背景$K=(G, M, I)$的所有概念的外延的集合是G上的一个闭包系统，同样它的所有概念的内涵是M上的一个闭包系统[4]。我们的目标是在M的幂集$P(M)$中找出所有闭集。本文提出的算法将$P(M)$作为闭包的搜索空间，并将它划分为一些子搜索空间。在搜索过程中保存一些必要的中间结果，用来识别那些能生成正规闭包的子搜索空间，并提高闭包运算速度。下面给出一些基本定义。

对于属性集M，可以定义其上的任意一个线性序，使得

$M=\{m_1<m_2<\cdots<m_{|M|}\}$。

对$A\subseteq M$，称$m\in A$是A的最大元素，表示为$Max(A)=m$，如果$\forall\ m_i\in A$且$m_i\neq m$有$m_i<m$。特别地，如果$A=\{m\}$，则$Max(A)=m$。

对$A\subseteq M$，本文将$(A')'$简化表示为A''，它是属性集合上的闭包算子。

对$A\subseteq M$，称A''是正规的，如果$\forall\ m_i\in A''\backslash A$，有$Max(A)<m_i$。

3.1　搜索空间描述及子搜索空间划分

首先，我们给出搜索空间的定义及子搜索空间的划分方法。

定义1　称$SS(CORE, CR)=\{ss|ss=CORE\cup cr,\ cr\in P(CR)\}$为一个搜索空间，其中，$CORE\subseteq M$，$CR\subseteq M$，且$\forall\ m_i\in CR$，$Max(CORE)<m_i$，$P(CR)$表示$CR$的幂集。称$CORE$为$SS$的核集，$CR$为$SS$的候选集。

由定义1知，一个搜索空间是由若干个属性集合构成的集合，$CORE$是搜索空间中每个属性集都包含的属性的集合，CR是在搜索空间的每个属性集中可能出现的属性的集合。显然，在搜索空间的每个属性集中一定不会出现的属性的集合为$M\backslash(CORE\cup CR)$，用$\overline{CR}$来表示，即$\overline{CR}=M\backslash(CORE\cup CR)$。

根据定义1，整个属性闭包的搜索空间$P(M)$可表示为搜索空间$SS(\varnothing, M)$。表2以特征向量形式给出形式背景k对应的整个搜索空间。

表2　形式背景k对应的整个搜索空间

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 0
… …
1 0 1 0 0 0 0 0 1
… …
1 1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1 1

定义2　给定搜索空间$SS(CORE, CR)$，定义$SS_i(CORE_i, CR_i)=\{ss|ss=CORE_i\cup cr,\ cr\in P(CR_i)\}$为由属性$m_i(m_i\in CR)$确定的子搜索空间，其中$SS_i$的核集是$CORE_i=CORE\cup\{m_i\}$，候选集是$CR_i=\{m|m_i<m,\ m\in CR\}$。特别地，定义$SS_0(CORE_0, CR_0)=\{CORE\cup\varnothing\}$。

根据定义2，可以将给定搜索空间$SS(CORE, CR)$划分为$|CR|+1$个子搜索空间。对表2给出的搜索空间，表3给出由每个属性确定的子搜索空间。

表3　形式背景k的子搜索空间

SS_0	0 0 0 0 0 0 0 0 0 0
SS_9(由i确定的)	0 0 0 0 0 0 0 0 0 [1]
SS_8(由h确定的)	0 0 0 0 0 0 0 [1] 0
	0 0 0 0 0 0 0 [1] 1
SS_7(由g确定的)	0 0 0 0 0 0 [1] 0 0
	0 0 0 0 0 0 [1] 0 1
	0 0 0 0 0 0 [1] 1 0
	0 0 0 0 0 0 [1] 1 1
… …	… …
SS_1(由a确定的)	[1] 0 0 0 0 0 0 0 0
	[1] 0 0 0 0 0 0 0 1
	… …
	[1] 1 1 1 1 1 1 1 0
	[1] 1 1 1 1 1 1 1 1

定理1　给定搜索空间$SS(CORE, CR)$，对$\forall\ i\neq j$，如果$SS_i(CORE_i, CR_i)$和$SS_j(CORE_j, CR_j)$为SS的子搜索空间，则有$SS_i\cap SS_j=\varnothing$，且$\bigcup_0^{|CR|}SS_i=SS$。

定理2　给定搜索空间$SS(CORE, CR)$，$SS_i(CORE_i, CR_i)$为属性$m_i(m_i\in CR)$确定的子搜索空间，对$\forall\ ss\in SS_i$，如果ss''是正规的，则$ss''\in SS_i$。

由定义1和定义2很容易证明定理1和定理2成立。由定理1和定理2可以得到如下结论，可以将一个搜索空间划分为若干个子搜索空间，在每个子空间搜索正规闭集，并且能够保证在所有子搜索空间找到的正规闭集的并等于在原搜索空间直接搜索得到的闭集的集合。

3.2　子搜索空间的有效性判断

将一个搜索空间划分为一些子搜索空间后，并不是在所有的子搜索空间都能找到正规的闭集。对于能够生成正规闭集的子搜索空间可以进一步划分，而不能生成正规闭集的子搜索空间可以不必考虑，这样可以有效提高算法的效率。

定义3 称由属性 m_i 确定的子搜索空间 $SS_i(CORE_i, CR_i)$ 是有效的，如果$\exists\ ss\in SS_i, ss"$是正规的；否则，称 SS_i 是无效的。

引理1 对属性 m_i 确定的子搜索空间 $SS_i(CORE_i, CR_i)$，若 $CORE"_i$ 不是正规的，则$\forall\ ss\in SS_i, ss"$一定不是正规的。

证明：因为 $CORE"_i$ 不是正规的，则$\exists\ m_k<m_i, m_k\in CORE"_i$ 且 $m_k\notin CORE_i$（由正规性定义）。对$\forall\ ss\in SS_i$，有 $CORE_i\subseteq ss$（由定义2），故 $CORE"_i\subseteq ss"$（由闭包操作的性质），进而 $m_k\in ss"$。又因为 $m_k\notin ss$（由定义2），所以 $m_k\in(ss"\backslash ss)$ 且 $m_k<Max(ss)$，即 $ss"$ 不是正规的（由正规性定义）。

定理3 由属性 m_i 确定的子搜索空间 $SS_i(CORE_i, CR_i)$ 是有效的，当且仅当 $CORE"_i$ 是正规的。

证明：（⇒）用反证法，假设 $CORE"_i$ 不是正规的，则$\forall\ ss\in SS_i, ss"$一定不是正规的（由引理1），即 SS_i 是无效的（由定义3），与已知 SS_i 是有效的矛盾。

（⇐）若 $CORE"_i$ 是正规的，则 SS_i 是有效的（由定义3）。

由定理3知，考察一个子搜索空间是否有效，只需考察该子搜索空间的核集是否能生成正规的闭集。例如表3中，由 a、b、c 和 g 分别确定的子搜索空间 SS_1、SS_2、SS_3 和 SS_7 是有效的，因为 $CORE"_1$、$CORE"_2$、$CORE"_3$ 和 $CORE"_7$ 是正规的。

定义4 给定搜索空间 $SS(CORE, CR)$，对$\forall\ m_i\in M$，定义 $G_{ss}(m_i)=(CORE\cup\{m_i\})'$ 为在搜索空间 SS 中与属性 mi 对应的对象集。

定理4 给定搜索空间 $SS(CORE, CR)$，由属性 m_i 确定的子搜索空间 $SS_i(CORE_i, CR_i)$ 是无效的，如果$\exists\ m_k$，$m_k<m_i$，且 $m_k\notin CORE$，有 $G_{ss}(m_i)\subseteq G_{ss}(m_k)$。

证明：由已知 $G_{ss}(m_i)\subseteq G_{ss}(m_k)$，即 $(CORE\cup\{m_i\})'\subseteq((CORE\cup\{m_k\})'$，有 $(CORE\cup\{m_i\})"\supseteq(CORE\cup\{m_k\})"$，所以 $m_k((CORE\cup\{m_i\})"$，即 $m_k\in CORE"_i$。又因为 $m_k<m_i$ 且 $m_k\notin CORE$（由已知），有 $m_k\notin CORE_i$，所以 $CORE"_i$ 不是正规的（由正规性定义），从而由属性 m_i 确定的子搜索空间 SS_i 是无效的（由定理3）。

对给定的搜索空间，若已知所有属性在当前搜索空间中所对应的对象集，可以根据定理4识别那些不能确定有效子搜索空间的属性。

例如形式背景 k，在整个搜索空间 $SS(\varnothing, M)$ 中每个属性对应的对象集是 $G_{ss}(a)=\{1, 2, 3\}$、$G_{ss}(b)=\{4, 5\}$、$G_{ss}(c)=\{1, 2, 4\}$、$G_{ss}(d)=\{3\}$、$G_{ss}(e)=\{5\}$、$G_{ss}(f)=\{1, 4\}$、$G_{ss}(g)=\{2, 3, 5\}$、$G_{ss}(h)=\{1, 4\}$ 和 $G_{ss}(i)=\{2, 3\}$。因为 $G_{ss}(d)\subseteq G_{ss}(a)$、$G_{ss}(e)\subseteq G_{ss}(b)$、$G_{ss}(f)\subseteq G_{ss}(c)$、$G_{ss}(h)\subseteq g(c)$ 和 $G_{ss}(i)\subseteq Gss(g)$，所以由 d、e、f、h 和 i 分别确定的子搜索空间 SS_4、SS_5、SS_6、SS_8 和 SS_9 是无效的。

3.3 概念生成及子搜索空间缩减

将一个搜索空间划分为若干个子搜索空间后，在每个有效的子搜索空间中生成该子搜索空间中的最大概念，然后对子搜索空间进行缩减，并进行进一步划分。

引理2 给定搜索空间 $SS(CORE, CR)$，$SS_i(CORE_i, CR_i)$ 是由属性 m_i 确定的有效子搜索空间，对$\forall\ m_j\in CR$，如果 $m_i<m_j$ 且 $G_{ss}(m_i)\subseteq G_{ss}(m_j)$，则 $m_j\in CORE"_i$。

证明：由已知 $G_{ss}(m_i)\subseteq G_{ss}(m_j)$，即 $(CORE\cup\{m_i\})'\subseteq(CORE\cup\{m_j\})'$，有 $(CORE\cup\{m_i\})"\supseteq(CORE\cup\{m_j\})"$，所以 $m_j\in(CORE\cup\{m_i\})"=CORE"_i$。

定理5 给定搜索空间 $SS(CORE, CR)$，$SS_i(CORE_i, CR_i)$ 是由属性 m_i 确定的有效子搜索空间。子搜索空间 SS_i 中的最大概念为 B，则 $Int(B)=CORE"_i=CORE_i\cup\{m_j\mid m_j\in CR, m_i<m_j, G_{ss}(m_i)\subseteq G_{ss}(m_j)\}$，$Ext(B)=G_{ss}(m_i)$。

证明：对$\forall\ ss\in SS_i$，有 $CORE_i\subseteq ss$（由定义2），所以 $CORE"_i$ 是 SS_i 中的最大概念 B 的内涵，即 $Int(B)=CORE"_i$。对$\forall\ m_j\in CR$，如果 $m_i<m_j$ 且 $G_{ss}(m_i)\subseteq Gss(m_j)$，则 $m_j\in CORE"_i$（由引理2），所以 $Int(B)=CORE"_i=CORE_i\cup\{m_j\mid m_j\in CR, m_i<m_j, G_{ss}(m_i)\subseteq G_{ss}(m_j)\}$。而概念 B 外延 $Ext(B)=CORE'_i=(CORE\cup\{m_i\})'=G_{ss}(m_i)$。

根据定理5，可以找到子搜索空间 $SS_i(CORE_i, CR_i)$ 中的最大概念，概念的内涵包含该子搜索空间的核集 $CORE_i$ 和所有满足 $G_{ss}(m_i)\subseteq G_{ss}(m_j)$ 的属性 m_j；概念的外延为具有这些属性的对象集合，即 $G_{ss}(m_i)$。例如，子搜索空间 $SS_3(\{c\}, \{d, e, f, g, h, i\})$ 可以用属性 f 进一步划分为有效的子搜索空间 $SS_{36}(\{c, f\}, \{g, h, i\})$。如果 SS_{36} 中最大的概念是 B，那么 $Int(B)=\{c, f, h\}$、$Ext(B)=\{1, 4\}$，因为在 SS_3 中，$G_{SS_3}(f)=(\{c\}\cup\{f\})'=(\{c, f\})'=\{1, 4\}$、$G_{SS_3}(h)=(\{c\}\cup\{h\})'=(\{c, h\})'=\{1, 4\}$ 且 $G_{SS_3}(f)\subseteq G_{SS_3}(h)$。

定义5 给定搜索空间 $SS(CORE, CR)$，$SS_i(CORE_i, CR_i)$ 是由属性 m_i 确定的有效子搜索空间。称 $SS_i^{\sim}(CORE_i^{\sim}, CR_i^{\sim})=\{ss\mid ss=CORE_i^{\sim}\cup cr, cr\in P(CR_i^{\sim})\}$ 为子搜索空间 SS_i 的缩减，其中，$CORE_i^{\sim}=CORE"_i=CORE_i\cup\{m_j\mid m_j\in CR, m_i<m_j, G_{ss}(m_i)\subseteq G_{ss}(m_j)\}$，$CR_i^{\sim}=CR_i\backslash\{m_j\mid m_j\in CR, m_i<m_j, G_{ss}(m_i)\subseteq G_{ss}(m_j)\}$。

引理3 给定搜索空间 $SS(CORE, CR)$，$SS_i(CORE_i, CR_i)$ 是由属性 m_i 确定的有效子搜索空间。对$\forall\ ss\in SS_i$，若 $ss"$ 是正规的，则 $CORE"_i\subseteq ss"$。

证明：对$\forall\ ss\in SS_i$，有 $CORE_i\subseteq ss$（由定义2），所以 $CORE"_i\subseteq ss"$（由闭包操作性质）。

定理6 给定搜索空间 $SS(CORE, CR)$，$SS_i(CORE_i, CR_i)$ 是由属性 m_i 确定的有效子搜索空间，$SS_i^{\sim}(CORE_i^{\sim}, CR_i^{\sim})$ 是 SS_i 的缩减。对$\forall\ ss_1\in SS_i$，若 $ss"_1$ 是正规的，则$\exists\ ss_2\in SS_i^{\sim}$，使得 $ss"_1=ss"_2$。

证明：因为 $ss"_1$ 是正规的，有 $CORE"_i\in ss"_1$（由引理2），不妨设 $ss"_1=CORE"_i\cup cr_1$，其中 $cr_1\in P(CR_i\backslash\{m_j\mid m_j(CR, m_i<m_j, G_{ss}(m_i)\subseteq G_{ss}(m_j))\}$，所以 $ss"_1\in SS_i^{\sim}$（由定义5）。设 $ss_2=ss"_1$，则 $ss_2\in SS_i^{\sim}$ 且 $ss"_1=ss"_2$（由闭包操作性质）。

由定理6知，子搜索空间中的所有正规闭集一定可以在其缩减中生成，即子搜索空间与其缩减是等价的。由于子搜索空间的缩减要小于原搜索空间，因此对缩减后的子搜索空间进行搜索可以提高算法效率。

根据上面的分析，可以知道 $SS_{36}^{\sim}(\{c, f, h\}, \{g, i\})$ 是 $SS_{36}(\{c, f\}, \{g, h, i\})$ 的缩减。尽管 $|SS_{36}^{\sim}|=4$，而 $|SS_{36}|=8$，它们是等价的。

4 算法 PCG

根据第3节的分析，本节给出一个基于搜索空间划分的并行概念生成算法 PCG。算法首先生成最大概念（G，∅），然后派生子过程 SpacePartition 来处理初始搜索空间 $P(M)$。算法的核心是子过程 SpacePartition，它将给定搜索空间划分为一些子搜索空间，使用候选集中每个属性在当前搜索空间中所对应的对象集对子搜索空间的有效性进行判断，标识能够划分有效子搜索空间的属性，进而生成有效子搜索空间中的最

大概念，且对有效子搜索空间进行缩减，最后对每个有效子搜索空间派生一个新的子过程 SpacePartition。下面给出算法伪码。

```
Subroutine SpacePartition (CORE, CR, CR̄)
/*这里 CR 和 CR̄ 扩展为二元组(m, G(m))，其中 m 为属性，G(m)为属性 m 在当前搜索空间中对应的对象集*/
Begin
/*第一步：标记 CR 中不能确定有效子搜索空间的属性*/
1 for ∀ (m_i, G(m_i)) ∈ CR do
  1.1 for ∀ (m_k, G(m_k)) ∈ CR̄ ∪ CR and m_k < m_i do
      1.1.1 if G(m_i) ⊆ G(m_k) then
            1.1.1.1 Mark (m_i, G(m_i))
/*第二步：对有效子搜索空间，生成其中的最大概念，计算其缩减，并进一步划分子空间*/
2 for ∀ (m_i, G(m_i)) ∈ CR and not Marked do
  2.1 CORE_i = CORE({m_i}
  2.2 for ∀ (m_j, G(m_j)) ∈ CR and m_i < m_j do
      2.2.1 if G(m_i) ⊆ G(m_j) then
            2.2.1.1 CORE_i = CORE_i ∪ {m_j}
      2.2.2 else if G(m_i) ∩ G(m_j) ≠ ∅ then
            2.2.2.1 CR_i = CR_i ∪ {(m_j, G(m_i) ∩ G(m_j))}
  2.3 for ∀ (m_k, G(m_k)) ∈ CR̄ ∪ CR and m_k < m_i do
      2.3.1 if G(m_i) ∩ G(m_k) ≠ ∅ then
            2.3.1.1 CR̄_i = CR̄_i ∪ {(m_k, G(m_i) ∩ G(m_k))}
  2.4 Output concept (G(m_i), CORE_i)
  2.5 Spawn SpacePartition (CORE_i, CR_i, CR̄_i)
End

Algorithm PCG (Parallel Concepts Generation)
Input: 形式背景 K=(G, M, I)。
Output: K=(G, M, I)上的所有概念。
Begin
    Output the largest concept (G, ∅)
    Spawn SpacePartition (∅, ∪_{m∈M} (m, m'), ∅)
End
```

图2给出算法 PCG 在表1的例子形式背景上的运算结果。从图2可以看出算法 PCG 产生一个概念树，从这个概念树可以进一步构造概念格。

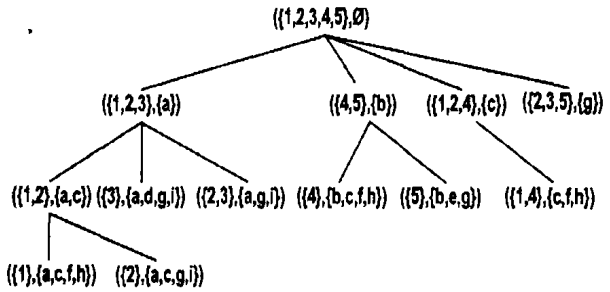

图2 算法 PCG 在形式背景 k 上的运行结果。

5 复杂性分析

本节分析 PCG 的理论复杂性。

设 Tp 是最坏情况下划分一个搜索空间并识别有效子空间所需时间，Tc 是在最坏情况下计算一个子搜索空间中最大概念及缩减所需时间，$\omega \leqslant \min(|G|, |M|)$是树的宽度(概念树中反链的最大长度)，$h \leqslant \min(|G|, |M|)$ 是树的高度(概念树中链的最大长度)，ε 为子过程 SpacePartition 的初始化时间。

定理7 算法 PCG 的时间复杂性是：

$h\cdot(Tp+Tc+\varepsilon)+(\omega-1)\cdot(Tc+\varepsilon)$

证明：概念树是包含 ω 个长度为 h 的链的有序集。这样，最大反链中的每个节点一定是一个最大链中的元素。

概念树中的每个对有一个共同的前驱节点。这样，任何两个链都有一个共同的节点。当这个节点串行计算它的子节点时，第一个子节点需要 $Tp+Tc+\varepsilon$ 的时间，每一个需要 $Tc+\varepsilon$ 的时间。这样计算第 i 个子节点的进程将在第一个节点之后的 $(i-1)\cdot(Tc+\varepsilon)$时间派生。

算法对所有最大链并行构造。构造第一个需要 $h\cdot(Tp+Tc+\varepsilon)$的时间。第二个在第一个之后的 $Tc+\varepsilon$ 时间完成……，最后一个在第一个之后的$(\omega-1)\cdot(Tc+\varepsilon)$时间完成。那么，时间复杂性是 $h\cdot(Tp+Tc+\varepsilon)+((\omega-1)\cdot(Tc+\varepsilon)$。

表4给出使用算法 PCG 时图1中所有概念的并行计算时间。并行算法计算所有的概念需$3\cdot Tp + 4\cdot(Tc+\varepsilon)$ 的时间，串行算法则需要至少$5\cdot Tp + 12\cdot Tc$ 的时间。

表4 使用算法 PCG 时图1中每个概念的并行计算时间

概念	时间（从开始算起）
({1,2,3},{a})	$Tp + Tc + \varepsilon$
({4,5}{b})	$Tp + 2Tc + 2\varepsilon$
({1,2,4},{c})	$Tp + 3Tc + 3\varepsilon$
({2,3,5},{g})	$Tp + 4Tc + 4\varepsilon$
({1,2},{a, c})	$2Tp + 2Tc + 2\varepsilon$
({4},{b, c, f, h})	$2Tp + 3Tc + 3\varepsilon$
({3},{a, d, g, i})	$2Tp + 3Tc + 3\varepsilon$
({2,3},{a, g, i})	$2Tp + 4Tc + 4\varepsilon$
({5},{b, e, g})	$2Tp + 4Tc + 4\varepsilon$
({1,4},{c, f, h})	$2Tp + 4Tc + 4\varepsilon$
({1},{a, c, f, h})	$3Tp + 3Tc + 3\varepsilon$
({2},{a, c, g, i})	$3Tp + 4Tc + 4\varepsilon$

结论 基于概念格系统的主要困难来自格构造本身。为了解决这个问题，本文提出了一个基于搜索空间划分的并行概念生成算法，并分析了它的复杂性。目前这个算法正在 MPI 下实现，并在真实的大数据集上进行实验。

参 考 文 献

1 Bordat J P. Calcul pratique du treillis de Galois d'une correspondence. Math. et Sci. Hum., 1986, 96: 31～47

2 Carpineto C, Romano G. A Lattice Conceptual Clustering System and Its Application to Browsing Retrieval. Machine Learning, 1996, 24: 95～122

3 Ganter B. Two Basic Algorithms in Concept Analysis: [Technical Report 831]. Technische Hochschule, Darmstadt, 1984

4 Fu H, Nguifo E M. A parallel algorithm to generate formal concepts for large data. In: Proc. 2nd Intl. Conf. on Formal Concept Analysis, Sydney, Feb. 2004. 394～401, B. Ganter, "Two Basic Algorithms in Concept Analysis": [Technical Report 831]. Technische Hochschule, Darmstadt, 1984

5 Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations. Springer-Verlag, 1999

6 Godin R, Mili H, Mineau G W, et al. Design of Class Hierarchies Based on Concept (Galois) Lattices. Theory and Application of Object Systems, 1998, 4 (2): 117～134

7 Kuznetsov S O. A Fast Algorithm for Computing all Intersections of Objects in a Finite Semi-lattice. Automatic Documentation and Mathematical Linguistics, 1993, 27 (5): 11～21

8 Kuznetsov S O, Obiedkov S A. Comparing performance of algorithms for generating concept lattices. Journal of Experimental and Theoretical Artificial Intelligence, 2002, 14(23)

9 Njiwoua P, Nguifo E M. A Parallel Algorithm to Build Concept Lattice. In: Proc. of 4th Groningen Intl. Information Technical Conf. for Students, 1997. 103～107

10 Norris E M. An Algorithm for Computing the Maximal Rectangles in a Binary Relation. Revue Roumaine de Mathématiques Pures et Appliquées, 1978, 23 (2): 243～250

11 Oostuizen G D. The Application of Concept Lattice to Machine Learning: [Technical Report]. University of Pretoria, South Africa, 1996

12 Stumme G, Wille R, Wille U. Conceptual Knowledge Discovery in Databases Using Formal Concept Analysis Methods. In: Proc. 2nd European Symposium on Principle of Data Mining and Knowledge Discovery (PKDD'98), Nantes, France, 1998. 450～458