基于搜索空间划分的概念生成算法*

齐 红[1+], 刘大有[1,2], 胡成全[1], 卢 明[1], 赵 亮[1]

[1](吉林大学 计算机科学与技术学院,吉林 长春 130012)

[2](符号计算与知识工程教育部重点实验室(吉林大学),吉林 长春 130012)

An Algorithm for Generating Concepts Based on Search Space Partition

QI Hong[1+], LIU Da-You[1,2], HU Cheng-Quan[1], LU Ming[1], ZHAO Liang[1]

[1](College of Computer Science and Technology, Jilin University, Changchun 130012, China)

[2](Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education (Jilin University), Changchun 130012, China)

+ Corresponding author: Phn: +86-431-5166493, Fax: +86-431-5168752, E-mail: qihong@jlu.edu.cn, http://www.jlu.edu.cn

Received 2004-04-26; Accepted 2005-01-07

Qi H, Liu DY, Hu CQ, Lu M, Zhao L. An algorithm for generating concepts based on search space partition. *Journal of Software*, 2005,16(12):2029−2035. DOI: 10.1360/jos162029

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. This paper proposes a new algorithm called SSPCG (search space partition based concepts generation) based on the closures search space partition. The algorithm divides the closures search space into several subspaces in accordance with the criterions prescribed ahead, and introduces an efficient scheme to recognize the valid ones, which bounds searching just in these valid subspaces. An intermediate structure is employed to judge the validity of a subspace and compute closures more efficiently. Since the partition of the search space is recursive and the searching in subspaces is independent, a parallel version can be directly reached. The algorithm is experimental evaluated and compared with the famous NextClosure algorithm proposed by Ganter for random generated data, as well as for real application data. The results show that the algorithm performs much better than the later.

Key words: formal concept analysis; concept lattice; search space; closure system; closed set

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

* Supported by the National Natural Science Foundation of China under Grant No.60173006 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2003AA118020 (国家高技术研究发展计划(863)); the Major Program of Science and Technology Development Plan of Jilin Province of China under Grant No.20020303 (吉林省科技发展计划重大项目)

作者简介: 齐红(1970－),女,吉林省吉林市人,副教授,主要研究领域为数据挖掘,概念格;刘大有(1942－),男,教授,博士生导师,CCF 高级会员,主要研究领域为知识工程与专家系统,分布式人工智能,多 Agent 系统,不确定性推理,空间推理,地理信息系统;胡成全(1957－),男,教授,主要研究领域为软件逆向工程,网络安全;卢明(1979－),男,硕士生,主要研究领域为数据挖掘,概念格;赵亮(1979－),女,硕士生,主要研究领域为数据挖掘,概念格.

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.提出了一种基于搜索空间划分的概念生成算法SSPCG(search space partition based concepts generation),它将属性集合的幂集看作初始闭包搜索空间,迭代地将每个搜索空间划分为一些子搜索空间,并引入了子搜索空间的有效性判断,只搜索那些能生成正规闭包的子搜索空间,有效地提高了搜索效率;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是独立的,所以该算法可以很容易地扩展为并行算法.在随机生成的数据集和真实数据集上进行的实验测试表明,本算法的时间性能要优于 Ganter 提出的 NextClosure 算法.

关键词: 形式概念分析;概念格;搜索空间;闭包系统;闭集
中图法分类号: TP18 文献标识码: A

概念格作为形式概念分析[1]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、软件工程等领域[2−5].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.许多学者对此进行了广泛研究,提出了一些概念格构造算法,如:Bordat 算法、NextClosure 算法、CbO(close by one)算法、Norris 算法、Chein 算法、Godin 算法等[6].实验比较表明,对于大而稠密的数据集,基于闭包运算的 NextClosure 算法[7]性能最好[6].但是,对于规模较大的数据集,使用该算法生成概念集仍然要耗费大量的时间.

本文提出了一种改进的基于闭包运算的概念生成算法 SSPCG(search space partition based concepts generation),它将属性集合的幂集(或对象集合的幂集)视为闭包的搜索空间,并对整个搜索空间进行划分,基于文中提出的测试方法识别出能生成正规闭包的子搜索空间,进而生成相应的最大概念,通过子搜索空间的进一步划分,重复这一过程直到子搜索空间不能再行划分为止.由于所有子搜索空间是独立的,所以该算法可以很容易地扩展为并行算法.

1 基本概念

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

形式背景 K 上的一个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 $(C,D)\leq(A,B)$.通过这种序关系,得到一个有序集 $\underline{B(K)}=(B(K),\leq)$,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[8].

Table 1 A example: Formal context k

表 1 一个例子:形式背景 k

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
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

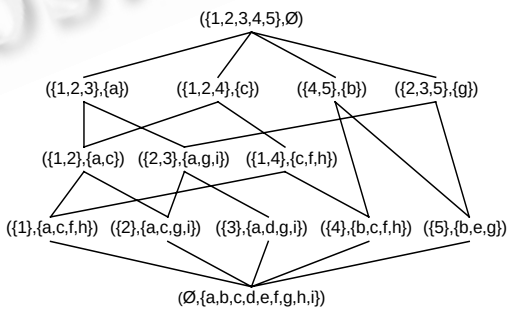

Fig.1 Concept lattice of formal context k

图 1 形式背景 k 对应的概念格

2 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[7].我们的目标是在 M 的幂集 $P(M)$中找出所有闭集.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 $M=\{m_1<m_2<\ldots<m_{|M|}\}$.

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 $\max(A)<m_i$.

2.1 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 可知,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.

Table 2 The whole search space of context k in characteristic vector

表 **2** 以特征向量形式给出的形式背景 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	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.

Table 3 Subspaces of the whole search space of context k determined by every attribute

表 **3** 由每个属性确定的形式背景 k 的子搜索空间

SS_0	0	0	0	0	0	0	0	0	0
SS_9(i-determined)	0	0	0	0	0	0	0	0	1
SS_8(h-determined)	0	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
SS_7(g-determined)	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(a-determined)	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 我们可以得到如下结论: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.

2.2 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 $m_k\notin CORE_i$.对$\forall ss\in SS_i$,有 $CORE_i\subseteq ss$ 故 $CORE_i''\subseteq ss''$,进而 $m_k\in ss''$.又因为 $m_k\notin ss$,所以 $m_k\in(ss''\backslash ss)$且 $m_k<\max(ss)$,即 ss''不是正规的. □

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:($\Rightarrow$)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,即 SS_i是无效的,与已知 SS_i是有效的矛盾.($\Leftarrow$)若 $CORE_i''$是正规的,则 SS_i是有效的. □

由定理 3 可知,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 m_i 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$m_k\in(CORE\cup\{m_i\})''$,即 $m_k\in CORE_i''$.又因为 $m_k<m_i$且 $m_k\notin CORE$,有 $m_k\notin CORE_i$,所以 $CORE_i''$不是正规的,从而由属性 m_i 确定的子搜索空间 SS_i 是无效的. □

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 识别那些不能确定的有效子搜索空间的属性.例如表 3 中,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.

2.3 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,并作进一步划分.

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 $Int(B)=CORE_i''=CORE_i\cup\{m_j|m_j\in CR,m_i<m_j,G_{ss}(m_i)\subseteq G_{ss}(m_j)\}$,$Ext(B)=G_{ss}(m_i)$.

证明:对$\forall ss\in SS_i$,有 $CORE_i\subseteq ss$,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 $G_{ss}(m_i)\subseteq G_{ss}(m_j)$,则 $m_j\in CORE_i''$,所以 $Int(B)=CORE_i''=CORE_i\cup\{m_j|m_j\in CR,m_i<m_j,G_{ss}(m_i)\subseteq G_{ss}(m_j)\}$.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 $G_{ss}(m_i)$.

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 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$\forall ss\in SS_i$,若 ss'' 是正规的,则 $CORE_i''\subseteq ss''$.

证明:对$\forall ss\in SS_i$,有 $CORE_i\subseteq ss$,所以 $CORE_i''\subseteq ss''$. □

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 $CORE_i''\subseteq ss_1''$,不妨设 $ss_1''=CORE_i''\cup cr_1$,其中 $cr_1\in P\ (CR_i\backslash\{m_j \mid m_j\in CR, m_i<m_j, G_{ss}(m_i)\subseteq G_{ss}(m_j)\})$,所以 $ss_1''\in SS_i^{\sim}$.设 $ss_2=ss_1''$,则 $ss_2\in SS_i^{\sim}$,且 $ss_1''=ss_2''$. □

由定理 6 可知,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.

3 算法 SSPCG

根据第 2 节的分析,本节给出一个基于搜索空间划分的概念生成算法 SSPCG.算法首先生成最大概念$(G,\varnothing)$,然后对初始搜索空间 $P(M)$调用过程 SpacePartition,最后输出最小概念$(\varnothing,M)$.过程 SpacePartition 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递归调用过程 SpacePartition,直到所有的子搜索空间都不能再划分为止.下面给出算法的伪码.

```
Procedure SpacePartition (CORE,CR, $\overline{CR}$ )   /*这里 CR 和 $\overline{CR}$ 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
  for ∀(m_i,G(m_i))∈CR do                            /*标记 CR 中不能确定有效子搜索空间的属性*/
    for ∀(m_k,G(m_k))∈ $\overline{CR}$ ∪CR and k<i do
      if G(m_i)⊆G(m_k) then
        标记(m_i,G(m_i))
  for ∀(m_i,G(m_i))∈CR and 未标记 do                  /*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*/
    CORE_i=CORE∪{m_i}
    for ∀(m_j,G(m_j))∈CR and i<j do
      if G(m_i)⊆G(m_j) then
        CORE_i=CORE_i∪{m_j}
      else if G(m_i)∩G(m_j)≠∅ then
        CR_i=CR_i∪{(m_j,G(m_i)∩G(m_j))}
    for ∀ (m_k,G(m_k))∈ $\overline{CR}$ ∪CR and k<i do
      if G(m_i)∩G(m_k)≠∅ then
        $\overline{CR_i}$ = $\overline{CR_i}$ ∪{(m_k,G(m_i)∩G(m_k))}
    输出概念(G(m_i),CORE_i)
    SpacePartition (CORE_i,CR_i, $\overline{CR_i}$ )
end
```

算法 **SSPCG**.

输入:形式背景 $K=(G,M,I)$.

输出:$K=(G,M,I)$上的所有概念.

begin

输出最大概念$(G,\varnothing)$;

SpacePartition $(\varnothing, \bigcup_{m\in M}(m,m'),\varnothing)$;

输出最小概念$(\varnothing,M)$;

end

4 实验结果

为了进行实验评价,我们使用 C++语言实现了 SSPCG 算法和 NextClosure 算法.在算法的实现过程中,使用了一些特殊的数据结构对两种算法进行优化,以获得最好的时间性能:(1) 通过自定义的位集合类(BitSet)及其操作来提高集合之间运算的速度.位集合类是将集合看成是内存中某个连续的空间,集合中的每个元素对应于这段内存空间中的某些位.这样,集合之间的操作就可以用内存中位运算来实现,从而提高集合运算的速度;(2) 为了方便对对象和属性进行随机访问,在定义的形式背景类中同时包含了两个 Hash 表.其中,一个以对象作为索引关键字,以该对象具有的所有属性的位集合作为该表项的内容;另一个则以属性作为索引关键字,以具有该属性的所有对象的位集合作为该表项的内容;(3) 对 NextClosure 算法,采用了一个辅助的栈结构[6],使得求一个集合的闭包时只需要考虑栈中的元素,而不需要遍历整个形式背景.

对比实验在 CPU 主频 PIV 2.6G,内存 512MB,操作系统为 Windows 2000 的 PC 机上进行.使用两类测试数据:随机生成的数据集和真实数据集.随机数据集是使用统一分布模拟一个二元关系生成的.随机生成的数据集虽然不是真实数据,但使用它可以方便地测试算法在不同参数下的性能,例如对象数$|G|$、属性数$|M|$、每个对象所具有的平均属性数$|g'|$等.真实数据集使用 UCI 机器学习数据库[9]中的 MUSHROOM 数据集(8 124 个对象,127 个属性).图 2~图 4 给出不同参数的随机数据集的测试结果,图 5 为 MUSHROOM 数据集的测试结果.实验结果表明,SSPCG 算法的时间性能要优于 NextClosure 算法.

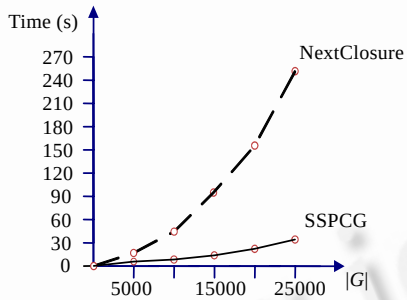

Fig.2 Comparison of SSPCG and NextClosure on the simulation database ($|M|$=100,$|g'|$=4)

图 2 算法 SSPCG 和 NextClosure 在模拟数据集($|M|$=100,$|g'|$=4)上的测试结果

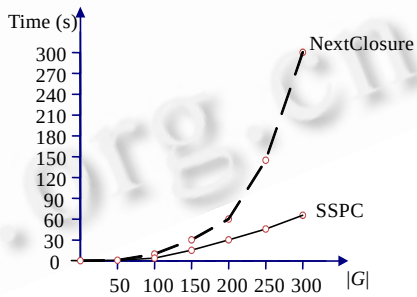

Fig.3 Comparison of SSPCG and NextClosure on the simulation database ($|M|$=100,$|g'|$=25)

图 3 算法 SSPCG 和 NextClosure 在模拟数据集($|M|$=100,$|g'|$=25)上的测试结果

本文提出的 SSPCG 算法本质上是一种基于闭包运算的概念生成算法,但算法的基本思想不同于传统的基于闭包运算的概念生成算法,如著名的 NextClosure 算法.NextClosure 算法以某种顺序生成所有的闭包,从而避免了概念的重复生成,但当所生成的闭包不满足规定的顺序时,该闭包被丢弃,相应的计算步骤也就浪费了.本文提出的算法使用一种全新的方法,将闭包空间视为搜索空间,并进行划分.对于子搜索空间,引入了子搜索空间的有效性判断,只搜索那些能生成正规闭包的子搜索空间,这样就在很大程度上避免了计算步骤的浪费,提高了搜索效率.但由于 SSPCG 算法对搜索空间进行划分,并保存一些中间结果,所以它的空间复杂性高于 NextClosure 算法.

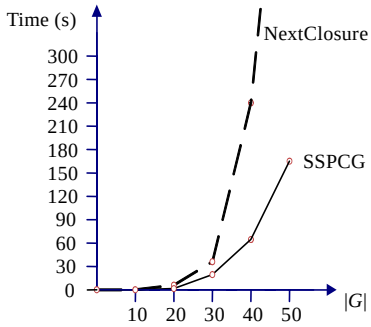

Fig.4 Comparison of SSPCG and NextClosure on the simulation database (|*M*|=100,|*g*′|=50)

图 4 算法 SSPCG 和 NextClosure 在模拟数据集(|*M*|=100,|*g*′|=50)上的测试结果

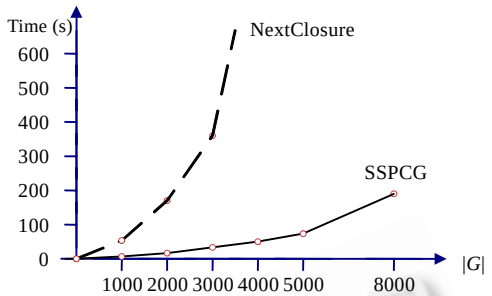

Fig.5 Comparison of SSPCG and NextClosure on the MUSHROOM database

图 5 算法 SSPCG 和 NextClosure 在 MUSHROOM 数据集上的测试结果相关工作及结论

与本文相关的工作还有 Fu 和 Nguifo[10]提出的一种基于搜索空间划分的并行概念生成算法.但该算法对搜索空间的划分与本文的划分方法有本质区别.Fu 等人的算法将搜索空间任意划分为不同子空间,然后对每个子搜索空间由一个处理器来生成概念;而算法 SSPCG 对子搜索空间的划分是动态的、迭代进行的,因此,如果将算法 SSPCG 扩展为并行算法将能够更有效地利用计算资源,达到更好的并行性.

本文提出的 SSPCG 算法属于一种批处理的构造算法,另外一类构造算法为渐近式算法,有代表性的如 Godin 算法[8].研究结果表明,Godin 算法更适合于小而稀疏的形式背景,对于大而稠密的形式背景基于闭包运算的概念生成算法的性能最好.由于真实的数据集往往是海量的,因此,本文提出的基于搜索空间划分的概念生成算法更为实用.

本文提出的算法只生成整个概念集,进一步的研究包括由概念集生成相应的格结构及基于搜索空间划分并行算法的设计和实验测试.

References:

[1] Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations. Berlin Heidelberg: Springer-Verlag, 1999.

[2] Carpineto C, Romano G. A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning, 1996, 24:95−122.

[3] Stumme G, Wille R, Wille U. Conceptual knowledge discovery in databases using formal concept analysis methods. In: Zytkow JM, Quafafou M, eds. Proc. of the 2nd European Symp. on Principles of Data Mining and Knowledge Discovery (PKDD'98). LNCS 1510, Berlin: Springer-Verlag, 1998. 450−458.

[4] Arevalo G, Mens T. Analysing object-oriented application frameworks using concept analysis. In: Bruel JM, Bellahsene Z, eds. Proc. of the Advances in Object-Oriented Information Systems (OOIS 2002) Workshops. LNCS 2426, Berlin: Springer-Verlag, 2002. 53−63.

[5] Baader F, Sertkaya B. Applying formal concept analysis to description logics. In: Eklund P, ed. Proc. of the 2nd Int'l Conf. on Formal Concept Analysis (ICFCA 2004). LNCS 2961, Berlin: Springer-Verlag, 2004. 261−286.

[6] Kuznetsov SO, Obiedkov SA. Comparing performance of algorithms for generating concept lattices. Journal of Experimental and Theoretical Artificial Intelligence, 2002,14(23):189−216.

[7] Ganter B. Formal concept analysis: Algorithmic aspects. 2002. http:www.math.tu-dresden.de/~ganter/cl02/

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

[9] Blake CL Merz CJ. UCI repository of machine learning databases. Department of Information and Computer Science, Irvine: University of California, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html

[10] Fu H, Nguifo EM. A parallel algorithm to generate formal concepts for large data. In: Eklund P, ed. Proc. of the 2nd Int'l Conf. on Formal Concept Analysis (ICFCA 2004). LNCS 2961, Berlin: Springer-Verlag, 2004. 394−401.