DOI：10.7652/xjtuxb201703020

多线程并行构建三支概念

祁建军，汪文威
（西安电子科技大学计算机学院，710071，西安）

摘要：针对三支概念分析理论中三支概念数量庞大、构建耗时的问题，提出了一种三支概念的并行构建算法PCbO3C。PCbO3C以提高三支概念的构建效率为目标，在三支概念串行构建算法CbO3C的基础上进行并行化改进，利用多线程技术并行计算给定形式背景的所有核心三支概念。并行化处理借鉴了算法PCbO的思想，通过串行算法CbO3C计算出第L层的所有三支概念，并存放到P个队列中，第L层当前生成的三支概念循环依次放入P个队列中，以使算法达到较高的负载均衡；创建P个线程，利用CbO3C并行处理P个队列中的三支概念，使得CPU资源得到充分利用。由于多线程间没有同步操作，使得PCbO3C算法的整体效率得到了进一步提高。为了验证算法PCbO3C的效率，在8核CPU环境下对多组UCI和随机数据进行实验，实验结果表明：PCbO3C速度上明显优于CbO3C，当线程数不超过8时，线程数每增加1倍，并行算法的速度可以提高约67%。

关键词：形式概念分析；三支概念分析；形式背景；多线程
中图分类号：TP18 **文献标志码**：A **文章编号**：0253-987X(2017)03-0116-06

A Multithreaded Parallel Algorithm for Constructing Three-Way Concepts

QI Jianjun, WANG Wenwei
(School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

Abstract: Aiming at the problems that the number of three-way concepts is large and the time constructing three-way concepts is lengthy, a parallel algorithm named PCbO3C was proposed to construct three-way concepts in this paper. In order to enhance the efficiency of constructing three-way concepts, PCbO3C is aimed at improving the sequential construction algorithm CbO3C by parallelization of using multithreading technology to compute all the core three-way concepts of a given formal context. This parallelization idea is similar to the algorithm PCbO. Firstly, PCbO3C computes all the three-way concepts of the Lth layer by CbO3C, and puts these three-way concepts into P queues in turn to get better load balance. Secondly, PCbO3C creates P threads and parallelly processes the three-way concepts in these P queues by CbO3C, i.e., one thread deals with one queue. This can take full advantage of CPU resources. Since there is no synchronous operation, the running speed of the algorithm is improved efficiently. To verify the efficiency of PCbO3C, experiments on some UCI databases and random datasets were conducted in the situation of 8-core CPU. The results show that the speed of PCbO3C can be increased by approximately 67% with the number of threads doubled if the number of threads is no more than 8.

Keywords: formal concept analysis; three-way concept analysis; formal contexts; multithreading

收稿日期：2016-08-23。 作者简介：祁建军(1970—)，男，副教授。 基金项目：国家自然科学基金资助项目(11371014，11071281)；陕西省自然科学基础研究计划资助项目(2014JM8306)。
网络出版时间：2016-12-27 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20161227.1022.004.html

三支概念分析(3WCA)是 Qi 等结合三支决策和形式概念分析提出的，其中三支概念及三支概念格是 3WCA 的关键内容[1-5]。

3WCA 包含两种新的概念构建形式，即由属性导出的三支概念以及由对象导出的三支概念。三支概念与形式概念形式类似，具有内涵和外延两个部分，不同的是：形式概念是一种二支决策的观点，仅能表示“共同具有”；三支概念是基于三支决策的，其内涵或外延本身也是一个二元组，可同时表示“共同具有”和“共同不具有”，涵盖更多的信息量。

目前，3WCA 已得到了有关学者的关注[6-9]。在三支概念的构造算法方面，已经提出了三支概念的串行构建算法 CbO3C[10]。CbO3C 通过深度优先和广度优先搜索所有概念空间，以字典序遍历所有属性子集；加入部分闭包检测和失效正则检测，对三支概念进行预判剪枝，提高算法效率；引入约简条件以过滤非核心三支概念。

关于三支概念的并行构建算法，目前尚没有相关的研究报道，而对于经典形式概念的并行构建算法，已有不少学者进行了研究。Njiwoua 等于 1997 年提出了 ParGaL 算法[11]。ParGaL 算法以 Bordat 串行算法为基础，需要和处理器之间进行很多交互，通过 PVM 并行虚拟机架构实现并行化。Fu 等于 2004 年提出了 ParallelNextClosure 算法[12]，该算法是在串行算法 NextClosure 的基础上进行并行化改进的，相比 ParGaL 算法减少了和处理器之间的交互。ParallelNextClosure 可以更自由地分解搜索空间，每个处理器可单独计算形式概念。Krajca 等于 2008 年和 2010 年分别提出了 PCbO 算法和 PFCbO 算法[13-14]，二者均利用多核多线程的并行化思想实现并行化。其中 PCbO 在串行算法 CbO 的基础上进行改进，PFCbO 在串行算法 FCbO 的基础上进行改进，均利用字典序遍历所有属性子集，关键操作采用位操作提高了算法效率，同时不需要同步操作，计算速度明显提升。

考虑到多核计算机的普遍性，为充分利用计算机的 CPU 核心资源，本文借鉴 PCbO 的思想将三支概念的串行构建算法 CbO3C 进行并行化改进，设计出三支概念的并行构建算法 PCbO3C。

1　3WCA 简介

定义 1[4]　形式背景(U,V,R)包括两个集合 U、V 和二者之间的关系 R。U 中的每一个元素称为对象，V 中的每一个元素称为属性。对象 x 与属性 a 有关系 R，记为 xRa，读作对象 x 具有属性 a。

对于形式背景(U,V,R)上的对象子集 $X\subseteq U$ 和属性子集 $A\subseteq V$，定义一对算子，$^*:\mathcal{P}(U)\to\mathcal{P}(V)$ 和 $^*:\mathcal{P}(V)\to\mathcal{P}(U)$，具体为 $X^*=\{v\in V\mid X\subseteq Rv\}$ 和 $A^*=\{u\in U\mid A\subseteq uR\}$，其中 $\mathcal{P}(\cdot)$ 表示幂集。

当 $X^*=A$ 且 $X=A^*$ 时，称(X,A)是一个形式概念，简称概念。其中 X 称为概念的外延，A 称为概念的内涵。

Qi 等将定义 1 给出的形式概念分析中传统的算子称为正算子[1]，并相应地定义了一对负算子 $^{\overline{*}}:\mathcal{P}(U)\to\mathcal{P}(V)$ 和 $^{\overline{*}}:\mathcal{P}(V)\to\mathcal{P}(U)$，具体为 $X^{\overline{*}}=\{v\in V\mid X\subseteq R^c v\}$ 和 $A^{\overline{*}}=\{u\in U\mid A\subseteq uR^c\}$。

定义 2[1]　设(U,V,R)是一个形式背景。对任意对象子集 $X,Y\subseteq U$ 和属性子集 $A,B\subseteq V$，定义一对 AE 算子 $^{\lhd}:\mathcal{P}(V)\to\mathcal{DP}(U)$ 和 $^{\rhd}:\mathcal{DP}(U)\to\mathcal{P}(V)$，具体为 $A^{\lhd}=(A^*,A^{\overline{*}})$ 和 $(X,Y)^{\rhd}=X^*\cap Y^{\overline{*}}$；定义一对 OE 算子，$^{\lhd}:\mathcal{P}(U)\to\mathcal{DP}(V)$ 和 $^{\rhd}:\mathcal{DP}(V)\to\mathcal{P}(U)$，具体为 $X^{\lhd}=(X^*,X^{\overline{*}})$ 和 $(A,B)^{\rhd}=A^*\cap B^{\overline{*}}$。其中，$\mathcal{DP}(\cdot)=\mathcal{P}(\cdot)\times\mathcal{P}(\cdot)$。

当 $(X,Y)^{\rhd}=A$ 与 $A^{\lhd}=(X,Y)$ 同时成立，称$((X,Y),A)$为属性导出的三支概念，即 AE 概念[2]。(X,Y)称为三支概念$((X,Y),A)$的外延，A 称为内涵。满足 $X\neq\varnothing$ 且 $Y\neq\varnothing$ 的 AE 概念$((X,Y),A)$称为核心 AE 概念，所有核心 AE 概念的集合称为 AE 核，记作 AECore[10]。

所有 AE 概念形成一个完备格，形式背景(U,V,R)的 AE 概念具有以下偏序关系：$((X,Y),A)\leqslant((W,Z),B)\Leftrightarrow(X,Y)\subseteq(W,Z)\Leftrightarrow B\subseteq A$，如果$((X,Y),A)\leqslant((W,Z),B)$，则$((X,Y),A)$称为$((W,Z),B)$的亚概念，而$((W,Z),B)$称为$((X,Y),A)$的超概念。$((X,Y),A)<((W,Z),B)$表示$((X,Y),A)\leqslant((W,Z),B)$且$((X,Y),A)\neq((W,Z),B)$，如果$((X,Y),A)<((W,Z),B)$且不存在$((X,Y),A)<((S,T),C)<((W,Z),B)$，则$((X,Y),A)$称为$((W,Z),B)$的子概念，而$((W,Z),B)$称为$((X,Y),A)$的父概念。

当 $X^{\lhd}=(A,B)$ 与 $(A,B)^{\rhd}=X$ 同时成立，称$(X,(A,B))$为对象导出的三支概念即 OE 概念[2]。X 称为三支概念$(X,(A,B))$的外延，(A,B)称为内涵。满足 $A\neq\varnothing$ 且 $B\neq\varnothing$ 的 OE 概念$(X,(A,B))$称为核心 OE 概念，所有核心 OE 概念的集合称为 OE 核，记作 OECore[10]。所有 OE 概念也形成一个完备格，OE 概念具有和 AE 概念类似的偏序关系。

2 并行构建算法 PCbO3C

借鉴形式概念并行构建算法 PCbO 的思想，在三支概念的串行算法 CbO3C 的基础上加入并行化，提出三支概念的并行构建算法 PCbO3C，并以构建核心 AE 概念为例给出具体算法。

2.1 PCbO3C 算法思想

算法 PCbO3C 在串行构建算法 CbO3C 的基础上加入并行化，在函数递归中加入了参数 l 代表当前递归的层次，通过多个线程并行计算核心 AE 概念。并行化处理借鉴了算法 PCbO 的思想，假定程序创建 P 个线程，L 代表递归调用的概念层次，$0<L<n$，在递归调用到 L 层之前处理方式和 CbO3C 基本一样，仅有两个区别：第一个区别是，在属性遍历到 $n-1$ 或者父概念的内涵为 V 时，程序并未直接退出；第二个区别是，在递归调用时增加对递归层次 l 更新操作。

算法的递归过程在到达 L 层后停止，位于 L 层的三支概念被存在队列中而非被处理。算法中设计 P 个队列，如何选择队列对 L 层概念进行存储以达到较高的负载均衡是算法的关键。由于在计算出所有三支概念之前，无法得到三支概念在搜索空间中的分布，所以很难达到完全负载均衡。PCbO3C 的策略是利用当前 P 个队列中所有三支概念数 C 对线程数 P 取模，即 $r=C\%P$，得到要选择的队列 r，将当前三支概念放入其中。P 个队列被不断填充，当递归层次 l 返回到 0 时，算法通过 P 个线程从 P 个队列取出三支概念，并行计算对应的所有核心 AE 子概念。

为并行计算出所有核心 AE 概念，PCbO3C 算法的初始对设为 $((X,Y),A)=((U,U),\varnothing)$，初始属性 $v=0$，初始失效集合 $\{N^v=\varnothing \mid v\in V\}$，初始递归层次 $l=0$。经过有限次计算，算法可以产生所有核心 AE 概念，而且每个核心 AE 概念仅计算一次。三支概念的并行构建算法 PCbO3C 具体如下。

PCbO3C($((X,Y),A),v,\{N^v \mid v\in V\},l$)

```
1  if l=L then
2     select r from 0 to P-1
3     Store((X,Y),A),v,queue[r])
4     return
5  if A=V or v>n-1 then
6     goto line 27
7  for j←v upto n-1 do
8     M^j←N^j
9     if j∉A and N^j⊆A∩V_j then
10       W←X∩{j}^*
11       if W=∅ then
12          continue
13       Z←Y∩{j}^(-*)
14       if Z=∅ then
15          continue
16       if X=W and Y=Z then
17          A← A∪ {j}
18       else
19          if A∩V_j=W^(*j)∩ Z^(-*j) then
20             PutInQueue(W,Z,j)
21          else
22             M^j←W^(*j)∩Z^(-*j)
23 ProcessConcept(((X,Y),A))
24 while GetFromQueue(W,Z,j)) do
25   B←A∪{j}
26   PCbO3C (((W,Z),B),j+1,{M^v|v∈V},
   l+1)
27 if l=0 then
28   for r←1 upto P-1 do
29     new process
30       while Get((W,Z),A),v,queue[r])
31          CbO3C(((W,Z),B),j+1,
   {M^v|v∈V})
32   while Get((W,Z),A),v,queue[0])
33     CbO3C(((W,Z),B),j+1,{M^v|v∈V})
```

算法步骤 1～4：当递归层次 l 等于预设递归层次 L 时，将 L 层的三支概念保存到队列 r 中而非进行计算处理。

算法步骤 7～26：算法在到达 L 层之前执行，执行过程和串行构建算法 CbO3C 基本一样，只是递归搜索过程中增加对概念层次 l 加 1 操作。

步骤 27～33：当 $l\neq 0$ 时，算法退出当前递归分支；当 $l=0$ 时，此时递归堆栈中没有递归分支，即到达顶部递归层。

在顶部递归层中，算法创建 $P-1$ 个线程分别调用串行算法 CbO3C 并行计算队列 1 到 $P-1$ 中三支概念对应的核心 AE 子概念。然后主程序通过算法 CbO3C 计算队列中三支概念对应的核心 AE 子概念。这样所有由 L 个或更多属性产生的三支概念将都通过多个线程并行处理，可以有效节约算法的运行时间。其中，三支概念的串行构建算法 CbO3C 具体如下[10]。

```
CbO3C(((X,Y),A),v,{N^v|v∈V})
1  for j←v upto n−1 do
2    M^j←N^j
3    if j∉A and N^j⊆A∩V_j then
4      W←X∩{j}^*
5      if W=∅ then
6        continue
7      Z←Y∩ {j}^(*j)
8      if Z=∅ then
9        continue
10     if X=W and Y=Z then
11       A← A∪ {j}
12     else
13       if A∩ V_j=W^(*j)∩ Z^(*j) then
14         PutInQueue(W,Z,j)
15       else
16         M^j←W^(*j)∩Z^(*j)
17 ProcessConcept(((X,Y),A))
18 while GetFromQueue(W,Z,j)) do
19   B←A∪{j}
20   CbO3C (((W,Z),B),j+1,{M^v|v∈V})
```

算法 PCbO3C 中多个线程之间不需要进行交互操作，提高了算法的性能。PCbO3C 中参数 L 的大小决定各线程并行处理的三支概念的数量。如果 $L=1$ 则大部分核心 AE 概念被一个或两个线程处理。随着 L 的增加，所有核心 AE 概念的计算可以更负载均衡的分配到各 CPU 核心上。然而，随着 L 的增加，会有更多的三支概念被串行处理，CPU 资源利用不充分。经过实验分析，L 取 2 或 3 时，算法的总体性能比较好。

2.2　PCbO3C 算法分析

在算法具体实现中，采用三支概念串行构建算法 CbO3C 中的位运算技术，所有的集合运算均利用位操作实现，节约运行时间，提高算法效率。算法 CbO3C 的时间复杂度为 $O(N|U||V|^2|\mathrm{AECore}|)$，其中 $|\mathrm{AECore}|$ 表示给定形式背景所有核心 AE 概念的数量，N 表示并行算法 PCbO3C 相对于串行算法 CbO3C 速度提升的倍数，N 最大接近 CPU 的核心数。

3　实验及分析

实验在 Intel Xeon E5640 双 CPU、8 核心、主频 2.66 GHz、内存 4 GiB 的台式机上进行，算法中的递归调用层 L 取 2。通过对 UCI 数据和随机数据进行实验，分析不同线程下算法 PCbO3C 的运行时间，验证在多 CPU 核心下，随着算法线程个数的增加，算法 PCbO3C 是否可以更加高效地计算出给定形式背景的所有核心三支概念。

3.1　UCI 数据实验分析

对 UCI Machine Learning Repository 网站中不同领域的数据 Nursery、Lung-Cancer 和 Mushroom 进行实验，实验原始数据通过软件 FcaBedrock 转化为算法使用格式，实验结果见表 1，其中 U 表示对象集，V 表示属性集，R 表示对象和属性的关系集，$d=\dfrac{|R|}{|U|\times|V|}\times 100$ 表示形式背景的密度[15]，t 表示串行算法 CbO3C 计算所有核心 AE 概念的时间，t_i 表示程序有 i 个线程时算法 PCbO3C 计算所有核心 AE 概念的时间。

由表 1 可知，在程序单线程时，三支概念的并行构建算法 PCbO3C 运行时间和其串行构建算法 CbO3C 相当。在程序多线程时，并行算法 PCbO3C 运行速度明显优于串行算法 CbO3C。具体表现为：在线程数达到 CPU 核心数 8 之前，针对形式背景 Mushroom 线程数每增加 1 倍提速比较明显，平均提速 75.8%；在线程数从 CPU 核心数 8 增加到核心数的 2 倍时，速度提升变缓，可提速 25.5%；在线程数超过核心数的 2 倍后，速度变化幅度较小，甚至会由于并行调度开销的增大而降低。形式背景 Nursey 和 Lung-Cancer 规模相对较小，并行的额外开销权重较大，速度提升低于 Mushroom。

3.2　随机数据实验分析

为进一步验证并行算法的正确性、高效性，对随机数据进行实验，固定形式背景的其他变量，分别考虑属性数量、对象数量和密度大小对并行算法 PCbO3C 性能的影响。

表 1　3 个 UCI 数据集实验结果

数据集	$\|U\|\times\|V\|$	d	$\|\mathrm{AECore}\|$	t/s	t_1/s	t_2/s	t_4/s	t_8/s	t_{16}/s	t_{32}/s	t_{64}/s
Lung-Cancer	32×162	35.19	1 120 807	4.021	4.010	2.574	2.184	1.450	0.936	0.780	0.749
Nursery	12 960×32	28.13	249 444	6.307	6.146	3.478	2.294	1.435	1.014	0.904	0.826
Mushroom	8 124×119	19.33	18 973 924	305.84	297.43	161.505	92.508	54.787	43.665	46.114	42.260

固定形式背景的属性数量为 40，密度为 30%，对象数量的变化范围为 200～2 000，实验结果如图 1 所示。由图 1 可知，保持属性数量和密度不变，算法 PCbO3C 计算三支概念的时间随着对象数量的增加而增加。另外，算法 PCbO3C 在单线程下计算速度和 CbO3C 相当；在线程数达到 CPU 核心数 8 之前，随着线程数量翻倍，平均提速为 63.4%；在线程数从 CPU 核心数增加到核心数的 2 倍时，速度提升平均为 20.8%；在线程数超过核心数的 2 倍后，速度变化幅度较小。

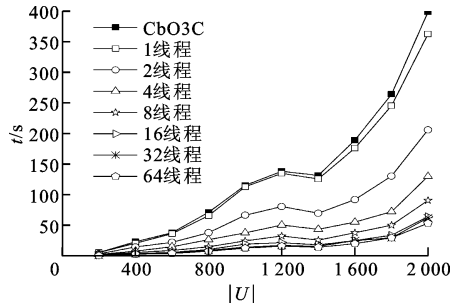

图 1 对象数对算法性能的影响($d=30\%$，$|V|=40$)

固定形式背景的对象数量为 1 000，密度为 30%，属性数量的变化范围为 30～70，实验结果如图 2 所示。由图 2 可知，在保持对象数量和密度不变的情况下，算法 PCbO3C 计算三支概念的时间随着属性数量的增加而增加。另外，PCbO3C 相比 CbO3C 速度的提升较明显，在线程数达到 CPU 核心数 8 之前，随着线程数量每增加 1 倍，平均提速为 67.3%；在线程数从 CPU 核心数增加到核心数的 2 倍时，速度提升平均为 21.2%；在线程数多于核心数的 2 倍时速度趋于稳定。

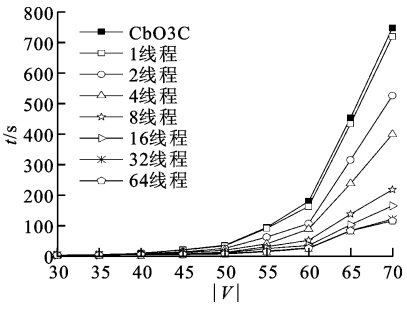

图 2 属性数对算法性能的影响($d=30\%$，$|U|=1\ 000$)

固定形式背景的对象数量为 1 000，属性数量为 40，密度的变化范围为 5%～95%，实验结果如图 3 所示。由图 3 可知，保持对象数量和属性数量不变，在形式背景密度处于 50%左右时，计算时间较长，在密度趋于 0%和 100%时，计算时间较短。另外，PCbO3C 在速度上优于 CbO3C，在线程数达到 CPU 核心数 8 之前，随着线程数量每增加 1 倍，平均提速为 66.2%；在线程数从 CPU 核心数增加到核心数的 2 倍时，速度提升平均为 20.2%；在线程数多于核心数的 2 倍时速度变化较小。

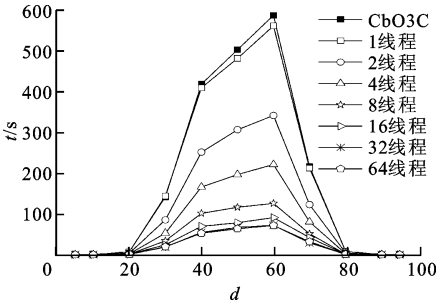

图 3 密度对算法性能的影响($|U|=1\ 000$，$|V|=40$)

通过对 UCI 数据和随机数据的实验分析可知，PCbO3C 在单线程时的效率和串行算法 CbO3C 相当；在线程数达到 CPU 核心数之前，随着线程数增加算法提速较大；当线程数增加到核心数的 2 倍时，速度仍有明显提升，但升幅变小；在线程数超过核心数的 2 倍后，速度变化幅度趋于稳定，几乎没有提升。

4 结 论

为节约三支概念的计算时间，本文对三支概念串行构建算法 CbO3C 进行了并行化改进，提出了三支概念的并行构建算法 PCbO3C。PCbO3C 通过多个线程并行处理核心三支概念，可以更加充分地利用 CPU 资源，提高算法的运行效率。对多组 UCI 数据和随机数据进行了实验，结果表明：在 8 核 CPU 环境下，并行构建算法 PCbO3C 性能明显优于串行构建算法 CbO3C；当线程数不超过 8 时，线程数每增加 1 倍 PCbO3C 的速度可以提升约 67%。目前，三支概念的并行构建算法有待进一步改进。

参考文献：

[1] QI J, WEI L, YAO Y. Three-way formal concept analysis [C]// Proceedings of 2014 International Conference on Rough Sets and Knowledge Technology. Berlin, Germany: Springer, 2014: 732-741.

[2] QI J, QIAN T, WEI L. The connections between three-way and classical concept lattices [J]. Knowledge-Based Systems, 2016, 91: 143-151.

[3] YAO Y. An outline of a theory of three-way decisions [C]// Proceedings of 2012 International Conference on Rough Sets and Current Trends in Computing. Berlin,

Germany: Springer, 2012: 1-17.

[4] GANTER B, WILLE R. Formal concept analysis, mathematical foundations [M]. Berlin, Germany: Springer, 1999.

[5] 王德兴，胡学钢，刘晓平. 一种新颖的基于量化概念格的属性归纳算法 [J]. 西安交通大学学报，2007，41(2)：176-179.
WANG Dexing, HU Xuegang, LIU Xiaoping. Novel attribute induction algorithm based on quantized concept lattice [J]. Journal of Xi'an Jiaotong University, 2007, 41(2): 176-179.

[6] LI J, REN Y, MEI C, et al. A comparative study of multigranulation rough sets and concept lattices via rule acquisition [J]. Knowledge-Based Systems, 2016, 91: 152-164.

[7] REN R, WEI L. The attribute reductions of three-way concept lattices [J]. Knowledge-Based Systems, 2016, 99: 92-102.

[8] 刘琳，钱婷，魏玲. 基于属性导出三支概念格的决策背景规则提取 [J]. 西北大学学报(自然科学版)，2016，46(4)：481-487.
LIU Lin, QIAN Ting, WEI Ling, et al. Rules extraction in formal decision contexts based on attribute-Induced three-way concept lattices [J]. Journal of Northwest University, Natural Science Edition, 2016, 46(4): 481-487.

[9] YAO Y. Interval sets and three-way concept analysis in incomplete contexts [J/OL]. International Journal of Machine Learning & Cybernetics, 2016: 1-18 [2016-06-26]. http://link.springer.com/article/10.1007/s13042-016-0568-1.

[10] 汪文威，祁建军. 三支概念的构建算法 [J]. 西安电子科技大学学报，2017，44(1)：71-76.
WANG Wenwei, QI Jianjun. An algorithm for constructing three-way concepts [J]. Journal of Xidian University, 2017, 44(1): 71-76.

[11] NJIWOUA P, NGUIFO E M. A parallel algorithm to build concept lattice [C/OL]. Proceedings of the 4th Groningen International Information Technology Conference, 1997 [2016-06-18]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.926.

[12] FU H, NGUIFO E M. A parallel algorithm to generate formal concepts for large data [C]// International Conference on Formal Concept Analysis. Berlin, Germany: Springer, 2004: 394-401.

[13] KRAJCA P, OUTRATA J, VYCHODIL V. Parallel recursive algorithm for FCA [C/OL]. Proceedings of the 6th International Conference on Concept Lattices and Their Applications, 2008 [2016-06-20]. //http://ceur-ws.org/Vol-433/paper6.pdf.

[14] KRAJCA P, OUTRATA J, VYCHODIL V, et al. Advances in algorithms based on CbO [C/OL]. Proceedings of the 7th International Conference on Concept Lattices and Their Applications, 2010 [2016-06-18]. http://ceur-ws.org/Vol-672/paper29.pdf.

[15] GAJDOŠ P, SNÁŠEL V. A new FCA algorithm enabling analyzing of complex and dynamic data sets [J]. Soft Computing, 2014, 18(4): 683-694.

（编辑　赵炜）

（上接第 115 页）

ZHENG Zedong, WANG Kui, LI Yongdong, et al. Current controller for AC motors using model predictive control [J]. Transactions of China Electrotechnical Society, 2013, 28(11): 118-123.

[3] GDAIM S, MTIBAA A, MOHAMED F M. Design and experimental implementation of DTC of an induction machine based on fuzzy logic control on FPGA [J]. IEEE Transactions on Fuzzy Systems, 2015, 23(3): 644-655.

[4] 尹忠刚，牛剑博，钟彦儒. 采用免疫算法的感应电机内模控制策略 [J]. 中国电机工程学报，2012，33(24)：97-105.
YIN Zhonggang, NIU Jianbo, ZHONG Yanru. Research on an internal model control strategy for induction motors using the immune algorithm [J]. Proceedings of the CSEE, 2012, 33(24): 97-105.

[5] PODLUBNY L. Fractional-order systems and $PI^{\lambda}D^{\mu}$ controllers [J]. IEEE Transactions on Automatic Control, 1999, 44(1): 208-214.

[6] 王瑞萍，皮佑国. 基于分数阶 PI 速度控制器的永磁同步电动机控制[J]. 电工技术学报，2012，27(11)：69-75.
WANG Ruiping, PI Youguo. Fractional-order PI speed controller for permanent magnet synchronous motor [J]. Transactions of China Electrotechnical Society, 2012, 27(11): 69-75.

[7] BENDJEDIA M, TEHRANI K A, AZZOUZ Y. Design of RST and fractional order PID controllers for an induction motor drive for electric vehicle application [C]// The 7th IET International Conference on Power Electronics, Machines and Drives. Piscataway, NJ, USA: IEEE, 2014: 8-10.

（编辑　赵炜）