一种基于概念矩阵的概念格生成算法

陈　震[1]　张　娜[1]　王甦菁[1]

（吉林大学计算机科学与技术学院，吉林省 长春市　130012）[1]

摘 要 概念格作为形式概念分析理论中的核心数据结构，在机器学习和数据挖掘等领域有着广泛的应用。构造概念格十分重要，针对此引入了概念矩阵思想，提出了基于概念矩阵的概念格生成算法CMCG(Concept-Matrix Based Concepts Generation)。该算法从格的顶端节点开始构造，基于概念矩阵，利用属性的秩为每个节点生成它的所有子节点，完成子节点到父节点之间的链接，并生成哈斯图。给出了这种算法的理论依据。论文结尾提供了这一算法的伪码,并通过实验证明了CMCG算法时间性能优于Lattice算法。

关键词 概念格，概念矩阵，矩阵的秩，形式概念分析，哈斯图

中图分类号：T18　文献标识码：A

New Algorithm of Generating Concept Lattice Based on Concept-Matrix

CHEN Zhen[1]　ZHANG Na[1]　WANG Su-jing[1]

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

Abstract　Concept lattice, the core data structure in FCA (Formal Concept Analysis), has been widely used in machine learning and data mining. In its applications, building concept lattice is very important, for which an efficient algorithm CMCG based on concept-matrix has been put forward. The algorithm started from the top node of the lattice, generated all subnodes for each node using the rank of the concept matrix's attributes, completed the link between the subnodes and their parent, and generated the Hasse graph. The validity of the algorithm was proved in theory. In the end, the pseudo code of CMCG Algorithm was given and that performance of CMCG is superior in time to Lattice Algorithm was proved by experiments.

Key words　Concept lattice, Concept matrix, Rank of matrix, Formal concept analysis, Hasse graph

1 引言

20 世纪 80 年代初，德国的 Wille 教授提出了形式概念分析理论[1]，经过二十几年的发展，概念格作为形式概念分析的核心数据结构，已经成为一种用于数据组织和数据分析的形式化工具，在数字图书馆及文献检索、软件工程、数据挖掘等许多领域得到了广泛应用[2]。比如：在交通枢纽仿真中，可以通过概念格分析出逻辑设施（通道，入口，出口等概念）之间的相互关系，在分析仿真结果时给出合理的评价标准。概念格的构造是应用形式概念分析的前提，但是构造的效率却很难令人满意，进而人们对此进行了广泛的研究，提出了多种不同的构造算法。这些算法主要分为两大类：增量算法和批处理算法。典型的增量算法有 Godin[3]，Capineto[4]和 T. B. Ho[5]的算法。典型的批处理算法有 NextClourse 算法[6]、Titanic 算法[7]、Nourine 算法[8]、Bordat 算法[9]和 Lindig 提出的

国家高技术研究发展计划（863 计划）(the National High-Tech Research and Development Plan of China under Grant No.2007AA11Z124).国家科技支撑计划子课题(No.2006BAJ18B02-06)

作者简介: **陈震**(1949－),男,教授,主要研究领域为数据库,数据挖掘,决策支持;**张娜**(1985－),女,硕士生,主要研究领域为计算机仿真,数据挖掘,机器学习;**王甦菁**(1976－),男,博士生,主要研究领域为机器学习,数据挖掘(通讯作者:wangsj08@mails.jlu.edu.cn).

Lattice 算法[10]等，其中 Lattice 算法是一个较为高效的算法。

本文引入了概念矩阵这一思想，提出了基于概念矩阵的概念格生成算法 CMCG。基于该矩阵，利用属性的秩把一个概念的所有后继节点找出来，并生成哈斯图。并通过实验证明了 CMCG 算法时间性能要优于 Lattice 算法。

2 概念格的定义

本节简要介绍所需要的概念格的基本概念。在形式概念分析中，数据是用形式背景来表示的，下面给出它的形式化定义。文[11]中对形式概念分析中的定义有详细的介绍。

定义1.一个形式背景(Formal Context)是一个三元组：$K=(O,A,R)$，其 O 中为对象(Object)的有限集合，A 为属性(Attribute)的有限集合，R 为 O 和 A 的二元关系，即 $R\subseteq O\times A$。对于 $x\in O, y\in A, (x,y)\in R$，表示“对象 x 具有属性 y”。

在本文中，如果 $(x,y)\in R$ 记为 1，否则记为 0。这样一个形式背景就可以看成是一个由 0,1 组成的矩阵。我们称这个矩阵为形式背景矩阵。图 1 是与表 1 中形式背景对应的形式背景矩阵。

表 1　一个简单的形式背景

Table 1 A Simple Formal Context

	a	b	c	d	e	f	g	h
1	1	1	1	0	0	0	0	0
2	1	1	0	1	0	0	1	0
3	0	1	1	0	1	0	0	0
4	1	0	0	1	1	0	1	0
5	0	0	1	0	0	1	0	0
6	0	0	0	1	0	0	0	0
7	0	0	0	0	0	0	0	1

定义2.设 $K=(O,A,R)$ 为一形式背景。对于集合 $X\subseteq O$，记：$f(X)=\{y\in A|\forall x\in X,(x,y)\in R\}$，表示对象 X 的共同属性集合，相应地，对于集合 $Y\subseteq A$，记：$g(Y)=\{x\in O\mid\forall y\in Y,(x,y)\in R\}$，表示具有共同属性 Y 的对象集合；

	a	b	c	d	e	f	g	h
1	1	1	1	0	0	0	0	0
2	1	1	0	1	0	0	1	0
3	0	1	1	0	1	0	0	0
4	1	0	0	1	1	0	1	0
5	0	0	1	0	0	1	0	0
6	0	0	0	1	0	0	0	0
7	0	0	0	0	0	0	0	1

图 1. 表 1 中形式背景对应的形式背景矩阵

Figure 1 Matrix Corresponding to Formal Context in Table 1

定义3.设 $K=(O,A,R)$ 为一形式背景，$X\subseteq O$，$Y\subseteq A$，如果 $f(X)=Y$ 且 $g(Y)=X$，则称 $C=(X,Y)$ 为 K 的一个概念。此时称 X 为 C 的外延，本文用 $Extent(C)$ 表示 C 的外延；称 Y 为 C 的内涵，本文用 $Intent(C)$ 表示 C 的内涵。我们用 $B(K)$ 记 K 的所有概念组成的集合。

定义4.设 $K=(O,A,R)$ 为一形式背景，$C_1=(X_1,Y_1)$，$C_2=(X_2,Y_2)$ 是 K 的两个概念，规定：

$$C_1\le C_2\Leftrightarrow X_1\subseteq X_2$$

此时，C_2 称为 C_1 的超概念(superconcept)，称 C_1 为 C_2 的子概念(subconcept)。

定义5.称偏序集 $CS(K)=(B(K),\le)$ 称为概念格。

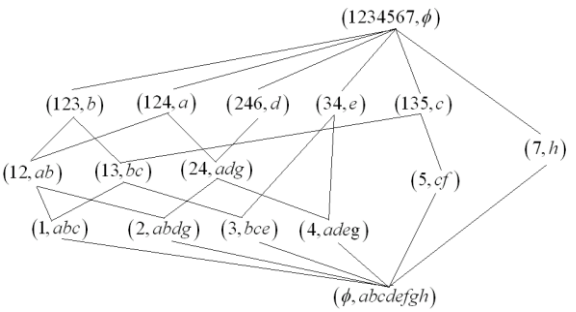

图 2.对应表 1 形式背景的概念格

Figure 2 Concept Lattice Corresponding to Formal Context in Table 1

定义6.设 $K=(O,A,R)$ 为一形式背景，$CS(K)$ 中两个不同的结点 $C_1=(X_1,Y_1)$ 和 $C_2=(X_2,Y_2)$，如果满足 $C_1\le C_2$，则记为 $C_1<C_2$。

定义7. 设 $K=(O,A,R)$ 为一形式背景，$CS(K)$ 中两个不同的结点 $C_1=(X_1,Y_1)$ 和 $C_2=(X_2,Y_2)$，如果 C_1 是 C_2 的子概念且不存在其它的结点 C_3 满足 $C_1<C_3<C_2$，则称 C_1 是 C_2 的子结点(直接后继)，而称 C_2 是 C_1 的父结点(直接前驱)。

由定义可知对于 C_2 的任何一个子结点 C_1，都有 $X_1 \subset X_2$，因为如果 $X_1 = X_2$ 时，C_1 和 C_2 是同一个结点。

3 概念矩阵的定义和原理

定义8[12].设 $K=(O,A,R)$ 为一形式背景，对于某个属性 $y \in A$，在 K 的矩阵中，如果 y 所对应的列有 t 个1时我们称该属性的秩为 t，记为 $r(y)=t$。记 $m=\max\{r(y) \mid y \in A\}$，称 m 为形式背景的秩。

定义9.设 $K=(O,A,R)$ 为一形式背景，$C=(X,Y)$ 是 K 中的一个概念，对于 X 中的每一个元素 x，从形式背景矩阵中取出 x 对应的行，组成新的矩阵 M，则称 M 为概念 C 的概念矩阵。例如概念 $(124,a)$ 的矩阵如图3所示。

	a	b	c	d	e	f	g	h
1	1	1	1	0	0	0	0	0
2	1	1	0	1	0	0	1	0
4	1	0	0	1	1	0	1	0

图 3 表 1 中概念 $(124,a)$ 的矩阵

Figure 3 Matrix of Concept $(124,a)$ in Table 1

定义10. 设 $K=(O,A,R)$ 为一形式背景，$C=(X,Y)$ 是 K 中的一个概念，在 C 的概念矩阵中，如果 y 所对应的列有 t 个1时我们称在概念 C 中该属性的秩为 t，记为 $R_C(y)=t$，记 $m=\max\{R_C(y) \mid y \in A, y \notin Y\}$，称 m 为概念 C 的秩。

性质1. 概念 C 的所有不同于 C 的子概念的外延个数最大的是 m.

证明:假设存在 $C_1=(X_1,Y_1)$，满足 $C_1<C$ 且 $m<|X_1|$。那么至少存在一个属性 y，在 C 的概念矩阵中，y 所对应的列中是1的位置对应的行的集合就是 X_1，所以 y 所对应的列有 $|X_1|$ 个1，这与 C 的秩是 m 矛盾。所以概念 C 的所有不同于 C 的子概念的外延个数最大的是 m。 □

定义11. 设 $K=(O,A,R)$ 为一形式背景，$C=(X,Y)$ 是 K 中的一个概念，对于集合 $Y_1 \subseteq A$，记：$g_C(Y_1)=\{x \in X \mid \forall y \in Y_1,(x,y) \in R\}$，表示 X 中具有共同属性 Y_1 的对象集合。

定理1. 设 $K=(O,A,R)$ 为一形式背景，$C=(X,Y)$ 是 K 中的一个概念，m 为概念 C 的秩，$\forall y \in \{y \mid R_C(y)=m, y \in A\}$，记 $C_1=(g_C(y), f(g_C(y)))$，则 C_1 是 C 的子结点。

证明:由定义10和定义11可知 $|g_C(y)|=m$，如果存在一个不同于 C 和 C_1 的结点 $C_2=(X_2,Y_2)$，使得 $C_1<C_2$，则有 $m=|g_C(y)|<|X_2|<|X|$，又有性质1可知概念 C 的所有不同于 C 的子概念的外延个数最大的是 m，所以 $|X_2| \le m$，这与 $m<|X_2|$ 矛盾。所以 C_1 是 C 的子结点。 □

定理2. 设 $K=(O,A,R)$ 为一形式背景，$C=(X,Y)$ 是 K 中的一个概念，m 为概念 C 的秩，$C_1=(g_C(y_1), f(g_C(y_1)))$ 是 C 的子概念，其中 $y_1 \in A$，且 $R_C(y_1)=m_1>0$，如果 $\forall C_2 \in \{C_2=(X_2,Y_2) \mid C_2<C, m_1<|X_2|\}$，都有 $g_C(y_1) \not\subset X_2$，则 C_1 是 C 的子结点。

证明:假设存在 $C_2'=(X_2',Y_2')$，使得 $C_1<C_2'<C$，则 $m_1=|g(y_1) \cap X|<|X_2'|<|X|$，所以 $C_2' \in \{C_2=(X_2,Y_2) \mid C_2<C, m_1<|X_2|\}$，于是 $g_C(y_1) \not\subset X_2'$。又因为 $C_1<C_2'<C$，所以 $g_C(y_1) \subset X_2'$，这与 $g_C(y_1) \not\subset X_2'$ 矛盾。所以 C_1 是 C 的子结点。 □

4 CMCG 算法

4.1 计算子结点的算法

本算法的思想是通过计算概念 C 的矩阵中每个属性的秩，从而得到 C 的所有子结点集合。

例如对于表1的形式背景，来求概念 $C=(124,a)$ 的所有子结点集合。C 的秩等于2。从图3看出 $C=(124,a)$ 的矩阵中，$R_C(b)=R_C(d)=R_C(g)=2=C$ 的秩，其中属性 d 对应的列和属性 g 对应的列相同。由定理1知 $(g_C(b), f(g_C(b)))=(12,ab)$ 是 $(124,a)$ 的一个子结点。对于属性 d 和属性 g 来说，因为它们在概念 $(124,a)$ 的矩阵中对应的列相同，所以应用定理2会产生两个相同的 $(124,a)$ 的子结点。为了避免重复计算，可以把定理1中的属性 y 看作为一个在概念矩阵中具有相同列向量的属性集合，这样就得出

$\left(g_C(dg), f\left(g_C(dg)\right)\right)=(24,adg)$。

$f\left(g_C(dg)\right)=\{adg\}=\{a\}\cup\{dg\}$，其中属性 d 和属性 g 是用矩阵秩的方法找到的，而 $\{a\}$ 是概念 C 的内涵。于是 $f\left(g_{(X,Y)}(Y_1)\right)=Y\cup Y_1$。我们在下面将介绍的算法中就是这样的计算 $f\left(g_{(X,Y)}\left(Y_1\right)\right)$ 的。

这样已经把概念 $(124,a)$ 的矩阵中秩为2的属性都遍历完了。接下来遍历秩为1的属性。从图3看出 $C=(124,a)$ 的矩阵中，$R_C(c)=R_C(e)=1$。而 $g_C(c)=\{1\}\subseteq\{12\}$，由定理2知 $\left(g_C(c), f\left(g_C(c)\right)\right)$ 不是 $(124,a)$ 的子结点。同理，$g_C(e)=\{4\}\subseteq\{24\}$，由定理2知 $\left(g_C(e), f\left(g_C(e)\right)\right)$ 也不是 $(124,a)$ 的子结点。

下面给出求给定概念的所有子结点集合的算法SUBNODES。

表 2 求给定概念的所有子结点集合的算法 SUBNODES

Table 2 SUBNODES Algorithm for Seeking the Set of All Sub-nodes Set of A Given Concept

Algorithm SUBNODES (C,K)

输入：给定的概念 $C=(X,Y)$ 和形式背景 $K=(O,A,R)$

输出：概念 C 的所有子结点集合 *subnodes*

```
1   subnoes ← ∅
2   if |X|=1 then return (∅,A)
3   M ← C 对应的概念矩阵
4   计算各属性在概念矩阵中的 M 秩
5   m ← C 的秩
6   if m=0 then return (∅,A)
7   do while m>0
8      S ← 秩为 m 的属性集合
9   do while S≠∅
10  Y1 ← 从 S 中取出一个属性，并且从 S 中取出和这个属性在 M
        中具有相同列的属性组成的集合
11  S ← S−Y1
12  X1 ← gC(Y1)
1   Y1 ← Y∪Y1
2   if 对于 subnodes 中的每个概念 C2=(X2,Y2) 都有 X1⊄X∩X2
        then subnodes ← subnodes∪(X1,Y1)
3   loop
4   m ← m−1
5   loop
6   return subnodes
```

4.2 结点在有向图中的存储

我们的这个方法中，用一个有向图 G 来存储概念格。伪码如下：

表3 算法CMCG

Table 3 CMCG Algorithm

Algorithm CMCG $((O,A,R))$

输入：形式背景 (O,A,R)

输出：概念格

```
1   将顶层结点 (O,∅) 加入到图 G 中
2   do while
3   从图 G 中选择中所有出度为0的结点，放入集合 S 中去
4   if S 中只有一个元素，并且这个元素是 (∅,A) then return
5   foreach C in S
6     foreach C1 in SUBNODES(C)
7   把结点 C1 加入图 G 中
8   把边 (C,C1) 加入图 G 中
9   loop
10  loop
11  loop
```

其中第7步把结点 C_1 加入图 G 中，需要先查询结点 C_1 是否已经存在。当结点数量大的时候，这种查询的操作是相当费时的。Lattice算法中是采用搜索树来完成的。对搜索树的查询所花的时间也是随着

结点的数量增加而增加的。而且如果使用AVL树，当增加结点时，需要对AVL树进行平衡的调整，这个是需要花费一定的时间。

在CMCG算法中，我们用Trie树来用于结点的搜索。在一个概念格中，概念和概念的内涵或外延一一对应的，我们这里选择概念的内涵作为确定一个概念的关键字，因为一般情况下对于形式背景 $K=(O,A,R)$ 来说，总是 $|O|>|A|$。我们可以把内涵用一个定长的0，1组成的字符串表示，字符串长度为 $|A|$。例如概念 $(5,cf)$，其内涵 $\{cf\}$ 可以表示成00100100。然后用一个度为2，高为 $|A|$ 的Trie树来存储这个内涵。

其优点是：1.对一个结点的一次查询最坏的时间代价是一常量，这个常量和结点的数量是无关的;2.因为它是把关键字分解存储到Trie树的每个结点上的，所以它还需要的空间也比较少。

5 实验结果

为了进行实验评价,我们使用C++语言实现了CMCG算法和Lattice算法.在算法的实现过程中,使用了一些特殊的数据结构对算法进行优化,以获得最好的时间性能。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。

对比实验在CPU为迅驰1.7G,内存1GB,操作系统为Fedora Core 5的IBM ThinkPad T42笔记本上进行。使用的测试数据是通过Galicia(http://www.iro.umontreal.ca/~galicia/)随机生成两组数据。第1组是由对象数目×属性数目分别是50×20，50×25，50×30，…，50×200的37组不同的形式背景组成。形式背景填充率(|R|除以|O|×|A|)都为30%。这组数据我们让对象的数目不变，属性数目不断增加。第1组数据的算法总时间和为每个节点生成的它的所有子节点函数所花时间分别如图4，图5所示。第2组是由对象数目×属性数目分别是50×20，100×20，150×20，…，3000×20的50组不同的形式背景组成。形式背景填充率都为30%。这组数据我们让属性的数目不变，对象数目不断增加。第2组数据的算法总时间和为每个节点生成的它的所有子节点函数所花时间分别如图6，图7所示。

比较这几幅图，我们可以发现当属性的个数不变时,随着概念个数的增加 CMCG 算法比和 Lattice 算法越来越快。从图 8 中我们也可以发现用 Trie 树来代替搜索树也能一定的提高算法的时间性能。

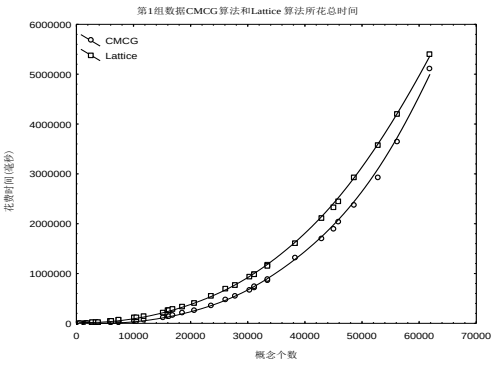

图 4. 第 1 组数据 CMCG 算法和 Lattice 算法所花总时间

Figure 4 Total Time Cost by CMCG and Lattice in Processing 1st Data

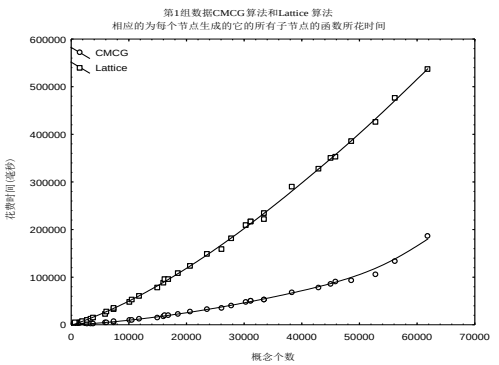

图 5. 第 1 组数据 CMCG 算法和 Lattice 算法相应的为每个节点生成的它的所有子节点的函数所花时间

Figure 5 Time Cost by CMCG and Lattice in Generating All Sub-nodes of Each Node in 1st Data

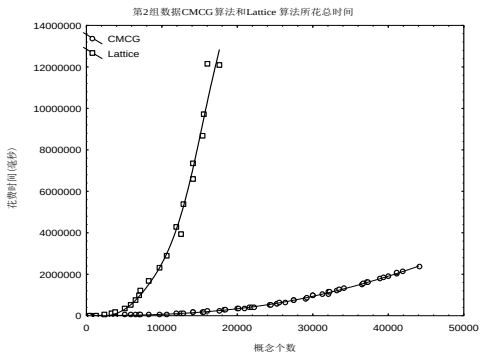

图 6. 第 2 组数据 CMCG 算法和 Lattice 算法所花总时间

Figure 6 Total time Cost by CMCG and Lattice in Processing 2nd Data

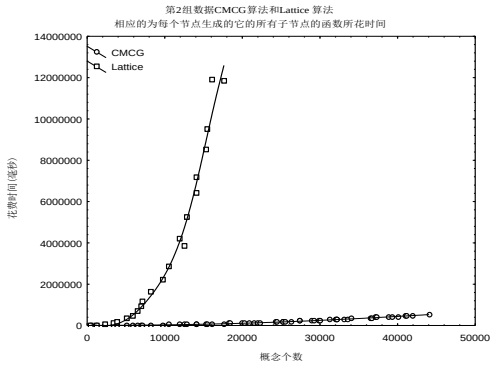

图7. 第2组数据CMCG算法和Lattice算法相应的为每个节点生成的它的所有子节点的函数所花时间

Figure 7 Time Cost by CMCG and Lattice in Generating All Sub-nodes of Each Node in 2nd Data

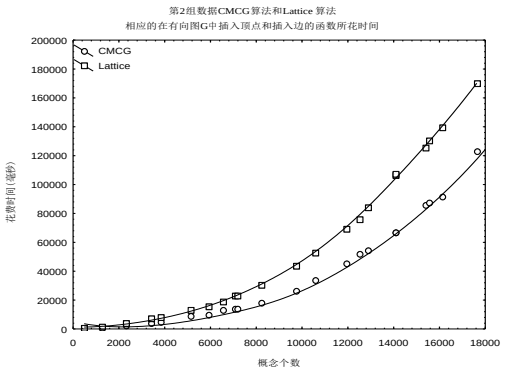

图 8. 第 2 组数据 CMCG 算法和 Lattice 算法相应的在有向图 G 中插入顶点和插入边的函数所花时间

Figure 8 Time Cost by the Function of Inserting Vertex and Edges into a Digraph G in 2nd Data by CMCG and Lattice

6 结论和将来的工作

目前，构造概念格的算法很多，本文引入了概念矩阵，基于该矩阵，利用秩把这个概念的所有后继结点都找出来。同时用实验证明了 CMCG 算法比 Lattice 算法要快。实验中还发现搜索一个概念是否已经在图中存在的操作也花了大量的时间，如何减少这个搜索时间也是值得进一步去研究的。

参 考 文 献

[1] Wille R.Restructuring lattice theory：An approach based on hierarchies of concepts[C]//Rival I. Ordered Sets. Dordrecht-Boston：Reidel， 1982：445-470.

[2] Oosthulzen G D.The application of concept lattice to machine learning[R].University of Pretoria，South Africa，1996.

[3] Godin R. Incremental concept formation algorithm based on Galois (concept) lattices[J].Computational Intelligence:1995, 11 (2) : 246～267.

[4] Carpineto C, Romano G. Galois: an order-theoretic approach to conceptual clustering. In: Proceedings of ICML-93, Utgoff Ped, 1993, 33～40

[5] Ho T B. Incremental conceptual clustering in the framework of Galois lattice. In: LU Hong-Jun, Motoda H, LIU Huan. KDD: Techniques and Applications. Singapore: World Scientific, 1997, 49～64

[6] Ganter B. Two basic algorithms in concept analysis, FB4-Preprint No.831, TH Darmstadt.

[7] Stumme G, Taouil R, Bastide Y, Pasquier N, and Lakhal L. Fast computation of concept lattices using data mining techniques. In Proceedings of 7th International Workshop on Knowledge Representation Meets Databases (KRDB 2000), Berlin, 2000, 129～139.

[8] Nourine L , Raynaud O. A fast algorithm for building lattices. In: Workshop on Computational Graph Theory and Combinatorics, Victoria, 1999, 199～204

[9] Bordat J P (1986). Calcul pratique du treillis de Galois d'une correspondance. Math. Sci.Hum., 1996: 31～47.

[10] Lindig C. Fast concept analysis. In Stumme G (Eds.), Working with Conceptual Structures - Contributions to ICCS 2000, Shaker Verlag, Aachen, Germany.

[11] Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations. Berlin: Springer, 1999

[12] 翟岩慧,曲开社,曹桃云. 基于矩阵秩的概念格生成算法. 电脑开发与应用, 2006, 19(5): 11～12

作者联系方式：
张　娜　Email：nazhang08@mails.jlu.edu.cn
13604365360
长春市前进大街 2699 号计算机大楼 B237　130012