基于 Eclipse 的概念格构建系统的设计与实现

蒋 平
(黄石理工学院 计算机学院,湖北 黄石 435003)

摘 要:概念格作为一种很好的形式化概念分析工具,被应用于数据挖掘、人工智能和知识提取等领域。如何将海量的数据中提取出来的相关规则即概念,迅速转化成直观的概念格图,成为概念格应用首先需要解决的问题。描述了概念格的相关概念,实现了概念格的改进构造算法,在 Eclipse 平台下实现了一种半自动化的概念格构建系统,并对各个模块的实现做了详细的介绍,生成相应的概念格的 Hasse 图,最终通过实验分析验证了半自动化构建概念格系统的可行性,优化了算法的空间复杂度,使系统性能得到优化。

关键词:概念格;形式概念分析;Eclipse
中图分类号:TP311.56 **文献标识码**:A **文章编号**:1673-629X(2011)04-0136-05

Design and Realization of Structuring Concept Lattice Based on Eclipse

JIANG Ping
(School of Computer, Huangshi Institute of Technology, Huangshi 435003, China)

Abstract: As an excellent formal concept analysis tool, concept lattice has been applied in data mining, AI, and knowledge extraction. How to present to the data obtained the relevant rules of that concept into a Hasse of the concept, will be the concept of lattice applications for the first thing to solve problems. Describe the relevant concepts of the concept lattice, implement improved algorithm of constructing concept lattice, build a system for a half automation of structring concept lattice in the platform of Eclipse, and introduce various parts of the details, generate appropriately a Hasse of concept lattice , and then, analysis by experiment verify the feasibility of the system of semi-automatic structuring concept lattice, and optimize the algorithm space complex and the system.

Key words: concept lattice; formal concept analysis; Eclipse

0 引 言

形式概念分析是一种基于格结构的数据分析方法,由 Wille 教授于 1982 年首次提出[1-3],概念格模型是形式概念分析的核心数据结构。概念它起源于一种哲学含义,由外延和内涵两部分组成,在形式概念分析中,外延定义为属于此概念的所有对象集合,内涵定义为属于此概念的所有对象共有的属性集合。

Eclipse[4,5]是一种基于 Java 语言的并且属于开源性质的用户可扩展的开发平台。它本质上是一个框架和一组服务,开发人员通过插件、组件来构建开发环境。

文中探讨了 Eclipse 平台概念格构建系统的设计与实现,并对算法做了改进和优化,实现了由抽象的元素属性关系到直观、生动的格图的转换。

收稿日期:2010-09-01;**修回日期**:2010-12-08
基金项目:湖北省高校重大科研项目基金(Z200622002)
作者简介:蒋 平(1983-),男,助教,研究方向为计算机软件工程与用例建模。

1 主要内容

文中在研究形式概念分析概念格的基础上,进行概念格构建系统的设计,包括系统的界面、格构建核心算法实现和 Eclipse 插件实现等。运用 Java 编程和 Eclipse 插件技术实现系统。

1.1 概念格

定义 1 一个形式化的背景(Context) C=(O,D,R),其中 O 中的元素称为对象,D 为描述符(属性)集合,R 为 O 和 D 的二元关系。可用 oRd,或者(o,d) ∈ R 来表达对象 O 和属性 D 的关系。

定义 2 如果(a1,b1),(a2,b2)是形式化背景中的概念,由层次关系而构建的所有反映了概念间的层次关系(O,D,R)的概念记作 β(O,D,R),被叫作概念格[2](Concept Lattice)。

1.2 Eclipse 概述

Eclipse 平台是 IBM 公司开发的一种开源的集成开发环境(Integrated Development Environment,简称 IDE),它建立在发现、集成和运行插件的工作机制之

上。因此,Eclipse 可以通过集成插件来不断地扩展自身的用户开发功能,以此来满足各种不同的应用,比如测试和调试程序等。

Eclipse 是一个开源的软件开发项目[4],主要由 Eclipse 项目、工具项目和技术项目三个项目构成。Eclipse 利用用户界面作为平台的使用工具,提供标准的跨系统、跨平台的用户导航模型;其核心是动态发现、集成和运行插件的体系结构。

2　设计及实现

2.1　概念格构建系统界面的实现

界面设计主要考虑的是用户和系统的交互[6],即用户方便输入数据,系统直观反映如图1所示。因此,对于一个二维的元素,属性即(oRd),采用表格的形式。如,m 个对象 n 个属性,利用一个 m 行 n 列的表格 T 实现,如图2所示。

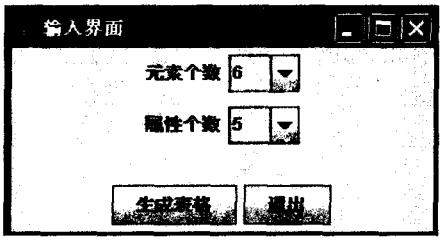

图1　输入界面一

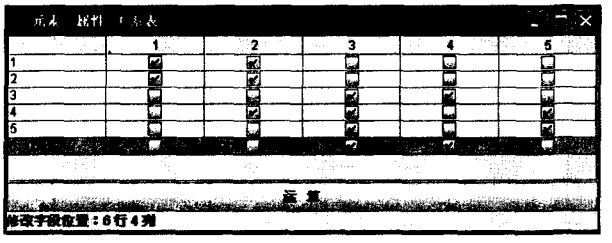

图2　输入界面二

2.2　概念格构建算法的实现

比较著名的概念格构造算法有 Ganter's Next-Concept Algorithm[3], Christian Linding 的 Fast Concept Analysis[7]等,文中使用一种 Bottom-Up[8]的计算方式计算出概念格。

算法描述:

给定形式化背景(O,D,R),求出概念格的概念并生成相应的概念格图:

第1步:建立一维的数据表 M,存放算法产生的中间节点,用节点 (I,J) 抽象表示一个(对象,属性)概念;

第2步:统计并计算出 $Max=|J|_{max}$,即节点 (I,J) 所对应的单一对象所具备的属性个数的最大值;

第3步:检查数据表 M 中的属性个数大于或等于1的节点 (I,J),并将其插入到数据表 M 中,其存放位置由 $|J|$ 决定;

第4步:检查 M 中相同属性的节点,同时将 M 中 Max 所对应的节点作为第一层(即 UP)节点;

第5步:从第 $n(n\geq 1)$ 层开始,循环1~4步,同时,如果对第 $n+1$ 层的每个节点 (i,j),满足 $i\cap I=i$,则将他们之间连接起来;如果对第 $n-x(1\leq x\leq n)$ 层的每个节点 (l,m),满足 $l\cap I=l$ 且 $|I|-|l|=1$,则连接。

第6步:算法结束。

2.2.1　生成原子概念模块

将数组 pro 各行元素进行逐位比较,并且变换与之相对应的 ele_temp,得到原子概念集,它是由数组 pro 和数组 ele_temp 各自对应的行的元素组成。算法如图3所示。

2.2.2　消除重复的原子概念模块

由2.2.1得到的 ele_temp 和 pro 中可能存在相同的行,为了降低系统运行过程中的数据冗余,逐行比较数组 ele_temp,若存在行相同,则将下面的行中的每一个数都赋为0。同时把数组 pro 中与之相对应的行也全赋为0。算法如图4所示。

2.2.3　Bottom_Up 算法模块

计算数组 pro_all_k:按行进行按位求与运算,将计算结果存到预先设置的一个临时数组中,如果临时数组中某 m 行与数组 pro_all_k 中的某 i 行相等,同时,设置二维输出数组 out[ ][ ],使 out[ ][0] = m +1、out[ ][1] = i +1,设置计数器输出 counter_out 自动加1;否则,将所有数据存放到数组 pro_all_k 的 counter+1 行,使 out[ ][0] = m +1、out[ ][1] = counter+1,设计输出计数器 counter_out 加1,如图5所示。

2.2.4　概念格构建系统的实现

按照上一步得到的结果进行画图,点 B 在最底部,点 T 在最顶部,然后从点 B 开始逐次向上连线,$B->1$ 表示从点 B 到点1连线,完成所有连线即得到图6。

2.3　Eclipse 插件的实现

Eclipse 插件体系结构中,其声明利用 XML 语言,而其实现则是在 Java 类文件中,即 PdePlugin. java 文件,一个比较典型的 Eclipse 插件它包含具体 JAR 库中的代码、只读文件和其他资源。

Eclipse 插件都会有一个相应的扩展点,而此扩展点又可能对应一个 API 接口,主要负责管理 Eclipse 插件的生存期的新插件的主类(如同有 main( )函数的 Java 类)。

较为常用的开发实现 Eclipse 插件的方法是通过添加开发者预先定义好的功能扩展点,然后将其程序代码转变成功能插件,步骤如下:

第1步:考察功能插件的性能,提供与平台集成的可行性解决方案;

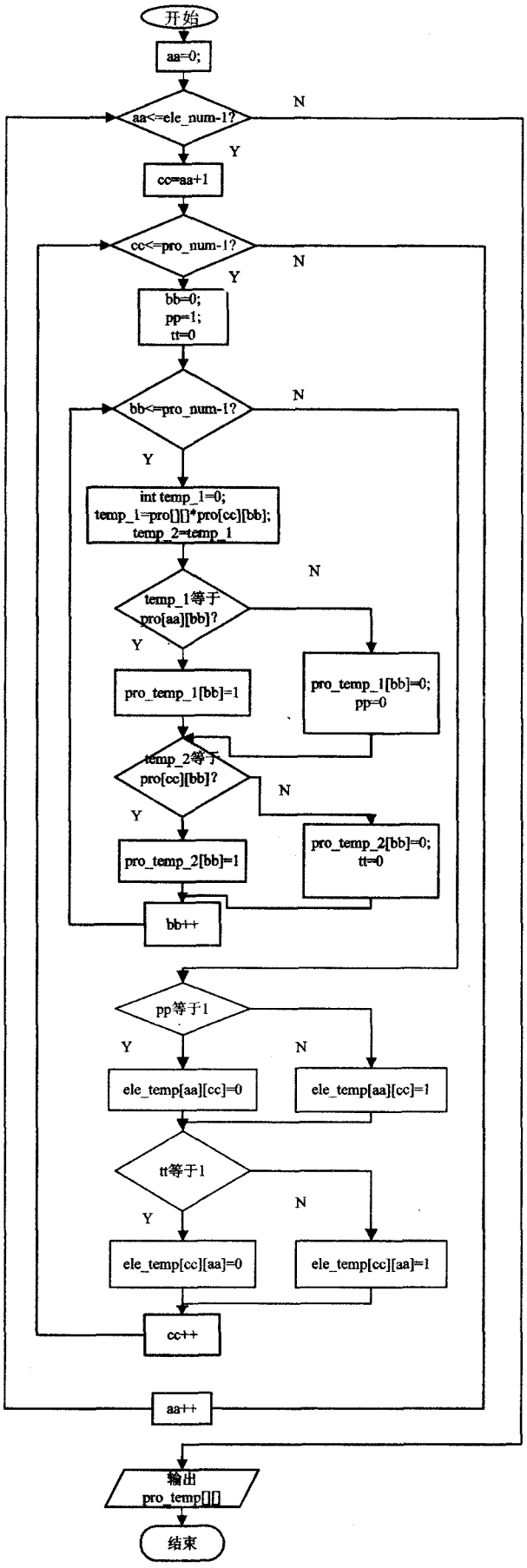

图 3 生成原子概念集流程

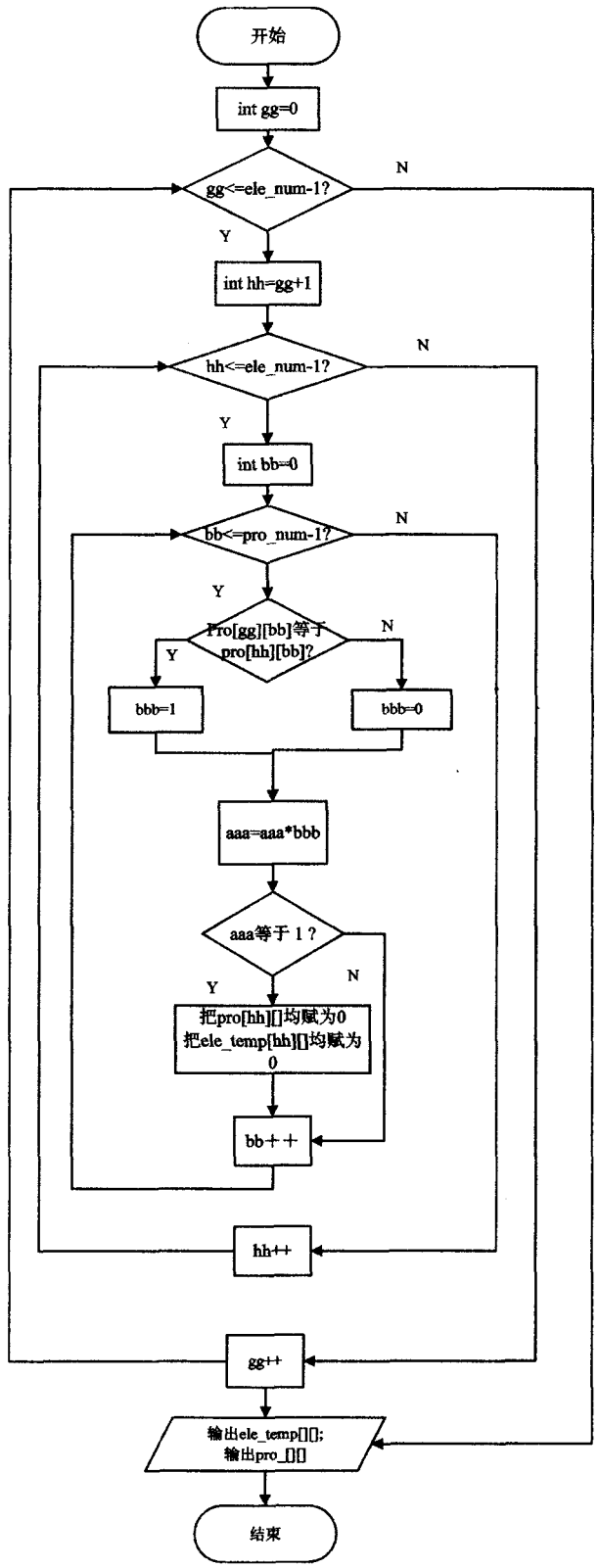

图 4 消除原子概念集中重复的项流程

第 2 步:根据插件的具体性能标识出功能扩展点,以便后续集成;

第 3 步:实现扩展点开发;

第 4 步:提供 xml 语言描述的声明文件(一般为 plugin. xml);

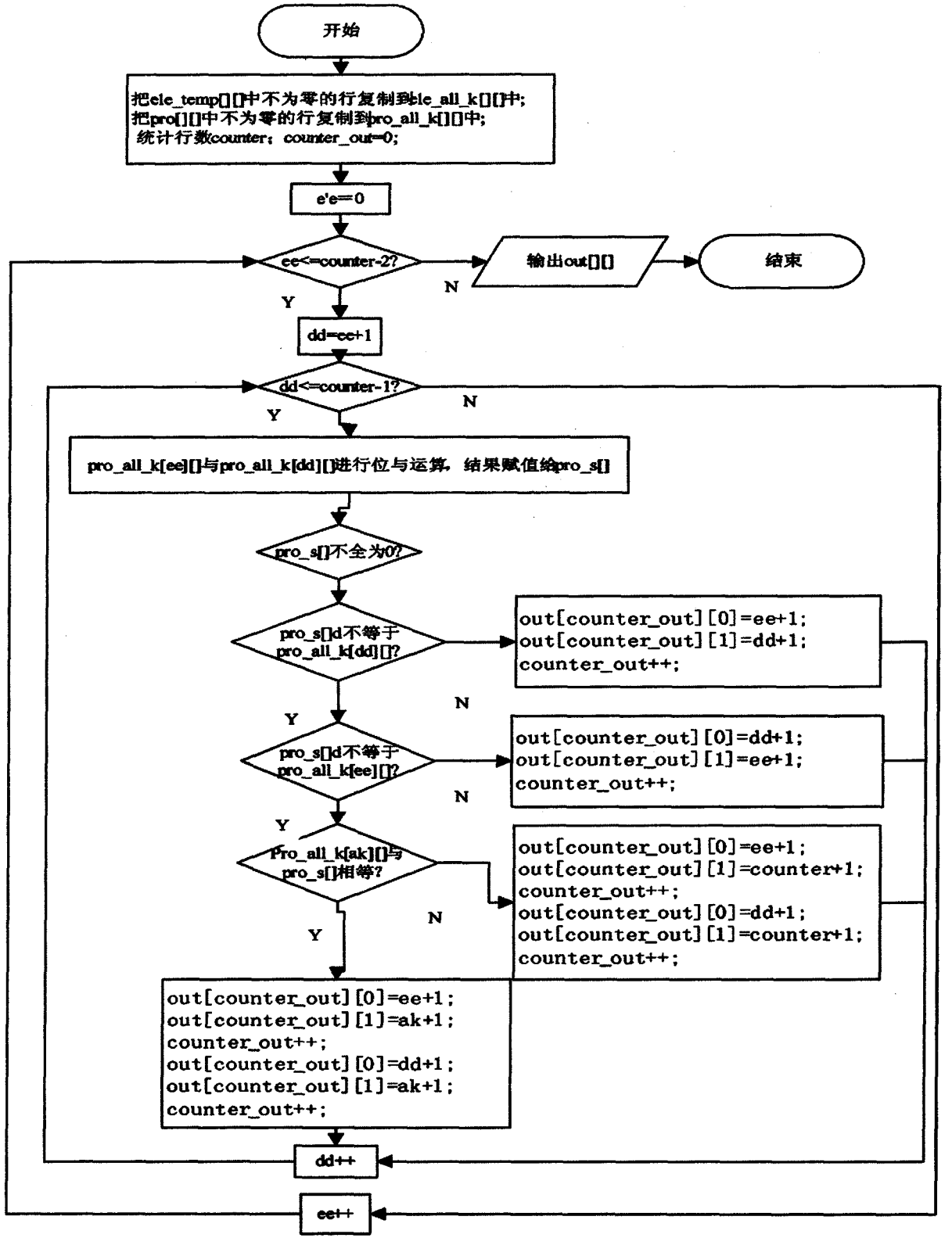

图 5　自底向上算法流程

第 5 步:测试并封装插件。

2.4　系统性能优化

Bottom-up 算法在处理容量十分庞大的数据时,会产生部分冗余数据,因此,需要进行算法的优化设计。具体实现有以下几点:

1、输入设计。采用了下拉框的形式,这个方法可以保证用户输入准确的数据;元素_属性关系表采用复选框的形式。

2、临时数组。设置并生成动态的临时数组,避免系统内存空间浪费,降低了系统的空间复杂度[9]。同

时,合理设置临时数组的大小,提高了系统性能。例如:格构建系统中用于存放中间元素属性关系的二维数组 pro_all_k,列数 n 即属性个数,行数 m 即元素的个数,m 的理论最小值为 elel_num,理论最大值 $C^2_{ele\_num}$ + ele_num 即(1/8) * (ele_num^2-ele_num)。

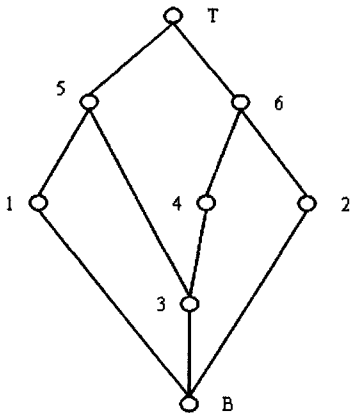

图 6 概念格图

3、数据类型。系统中的数组都采用 int 类型。

4、临时计数器。在系统中设置临时计数器降低了系统的时间复杂度。

3 结束语

文中借助概念格这一数学理论的指导,在 Eclipse 平台下设计一个概念格构造算法的插件,半自动地实现并生成概念格图。为以后其他各种运用形式概念分析方法的应用[10-12]提供了一种较为直观的基础保障。

参考文献:

[1] Wille R Restructuring lattice theory: an approach base on hierarchies of concepts[M]. Orered Sets. Reidel, Dordrecht-Boston:[s. n.],1982.

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

[3] Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundationgs[M]. Berlin:[s. n.],1999.

[4] 刘亚滨,杨 红. 精通 Eclipse[M]. 北京:电子工业出版社,2004:2-5.

[5] 刘甫迎,谢 春,徐 虹. Java 程序设计实用教程[M]. 北京:科学出版社,2005:85-280.

[6] 段友祥,郭 辉,林桂华. 一种通过交互界面实现用例抽取算法的研究与应用[J]. 计算机技术,2006,1(2):27-29.

[7] Lindig C. Fast Concept Analysis[EB/OL]. 2002. http://www. st. cs. uni-lindig. pdf.

[8] Stumme G, Maedche A. FCA-merge: bottom-up merging of ontologies[C]//17th Intel. Conf. on Artificial Intelligence(IJCAI'01). Germany:Springer,2001:225-230.

[9] 陈 玲. 浅谈算法的复杂性和常用算法[J]. 教育现代化,2004,3(2):66-67.

[10] 蒋 平,任胜兵,林 娟. 形式概念分析在软件工程中的应用[J]. 计算机技术与发展,2008,18(4):127-129.

[11] 张 涛,周爱武,谢荣传. 基于概念格和关联规则 Web 个人化系统[J]. 计算机技术与发展,2008,18(2):139-142.

[12] 胡可云,陆玉昌,石纯一. 基于概念格的分类和关联规则的集成挖掘方法[J]. 软件学报,2000,11(4):1478-1484.

(上接第 135 页)

值的信息。对文中提出的求核算法,虽然快捷有效,但在属性和数据对象增加及其对该算法的影响方面还有待研究,后期工作将继续增强算法的普适性和健壮性。

参考文献:

[1] 苗夺谦,李道国. 粗糙集理论、方法与应用[M]. 北京:清华大学出版社, 2008.

[2] 王国胤. Rough 集理论与知识获取[M]. 西安: 西安交通大学出版社, 2001.

[3] 张文修,吴伟志,梁吉业,等. 粗糙集理论与方法[M]. 北京:科学出版社, 2001.

[4] Hu Xiaohua, Cercone N. Learning in relational data-bases: a rough set approach[J]. Computational Intelligence, 1995,11(2):323-337.

[5] Wong S K M, Ziarko W. On Optimal Decision Rules in Decision Tables[J]. Bulletin of Polish Academy of Science,1985,33:693-696.

[6] Skowron A,Rauszer C. The Discerni-bility Matrics and functions in Information System[M]//Intelligent Decision Support Handbook of Applications and Advances of the Rough Sets Theory. Dordrecht: Kluwer Academic Publishers, 1992:331-362.

[7] 常犁云,王国胤,吴 渝. 一种基于 Rough Set 理论的属性约简及规则提取方法[J]. 软件学报, 1999,11(11):1206-1211.

[8] 李 侃,刘玉树,王 蕾. 一种粗糙集属性约简算法[J]. 计算机工程与应用,2002(5):15-16.

[9] 叶东毅,陈昭炯. 一个新的差别矩阵及其求核方法[J]. 电子学报, 2002, 30(7):1086-1088.

[10] 杨 明,孙志挥. 改进的差别矩阵及其求核方法[J]. 复旦学报:自然科学版, 2004, 43(5): 865-868.

[11] 乔 梅,韩文秀. 基于 Rough 集和数据库技术的属性约简算法[J]. 计算机工程,2005(6):18-19.

[12] 杨 飞,代广珍. 属性约简在高校就业决策分析中的应用[J]. 计算机技术与发展,2007,17(7):223-225.

[13] 赖桃桃,冯少荣,张东站. 基于改进差别矩阵的核增量式更新算法[J]. 计算机应用,2009(9):2477-2480.