可变精度粗糙集β值的增量计算*)

吉阳生　商　琳

(南京大学计算机软件新技术国家重点实验室　南京大学计算机科学与技术系　南京 210093)

摘　要　目前对于可变精度粗糙集中变精度参数β计算的研究,主要集中在非增量方面。当处理大量数据时,需要能够动态计算的方法,本文提出了一种增量计算β值的方法 ICObeta。该方法以分类质量作为确定性度量的标准,以最大确定性度量为目标,来选取合适的β值。ICObeta 相比于非增量的方法,具有动态增量和计算开销显著降低的优点,并通过实验证实了增量计算的优点。

关键词　可变精度粗糙集,分类质量,属性模式

Incremental Computation of Variable Precision Value in Variable Precision Rough Set

JI Yang-Sheng　SHANG Lin

(National Key Laboratory for Novel Software Technology, Nanjing University, Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract　The present research on computation of variable precision value β in variable precision rough set mainly focuses on non-incremental algorithms. This paper proposes an incremental algorithm-ICObeta which performs better on dynamic computation with the low computational cost. Theoretical analysis and experiments are shown in the paper.

Keywords　Variable precision rough set, Quality of classification, Attribute pattern

1　引言

粗糙集理论 RST(Rough Set Theory)20 世纪 80 年代由 Pawlak[1]提出。1990 年前后,粗糙集在模式识别、机器学习等领域已成功应用,产生了很多新的理论扩展和应用[2]。Ziarko[3]提出了一种称为可变精度粗糙集 VPRS (Variable Precision Rough Set)的模型,引入了一个参数 $\beta(0\leqslant\beta<0.5)$,将原来严格的集合包含关系,放宽到β所定义的程度。

VPRS 在应用中,参数β的处理一般是基于领域知识赋值。一方面,某些特定场合,甚至领域专家都很难给β确定一个合适的值;另一方面,基于领域知识的赋值,是否比从信息表计算更符合应用要求,这个问题有很多学者在研究。Beynon[4]研究了β的不同取值对于最终的约简结果和分类质量的影响。在规则生成的应用中,Su[5]把 VPRS 模型和一种神经网络模型相对照,得到的结果和神经网络相当;并且提出了一种新的计算β值的方法,该方法完全基于信息表,不需要额外的先验知识。Cheng[6]阐述了确定性度量的概念,提出了一种估算和计算β值的方法,并且用试验说明了β的选取对于约简结果和信息表确定性度量的影响。这些研究表明,通过满足信息表最大确定性度量的要求,能够得到一个符合实际应用的β,这样可以最大程度地减小因先验知识不足和选择β不合适所带来的影响。

目前对于β值计算的研究主要集中在以上几个方面,但是处理大量数据的动态增量方法尚未出现。本文提出了一种动态增量计算β值的方法:以分类质量作为确定性度量的标准,以最大分类质量为目标,来选取合适的β值,基于此提出了一种增量计算的方法 ICObeta;本文提出的定理 3.1 能够保证 ICObeta 选取的β值满足最大确定性度量的要求;实验说明了 ICObeta 是一种正确、高效的β值选取方法,并且体现出了 ICObeta 的优点:(1)ICObeta 是一种增量的方法。当前关于参数β计算的研究主要还是非增量的,但是当海量数据不断产生的时候,将所有历史数据都读入内存进行计算的方法就不合适了,这时候 ICObeta 增量特性可以解决这些问题;(2)ICObeta 的计算开销比较小。通过分析,可以看出它的时间复杂度是 $O(n\log n)$;在实验中也可以看出,相比于 Su[5]中的方法,ICOBeta 的累积计算开销只有不到 50%.

本文第 2 部分介绍 VPRS 模型;第 3 部分介绍 Su[5]计算β的基本思想,并且提出定理;第 4 部分介绍 ICObeta 算法,分析时间复杂度;第 5 部分通过实验的对比,说明 ICObeta 算法的正确性和高效性。

2　VPRS 模型

2.1　信息表 S

VPRS 的操作对象是知识表(亦称信息表),一个信息表 S 通常定义为:$S=\langle U, A, V, f\rangle$。其中 U 代表非空对象集合;A 代表对象的所有属性的集合,并且 $A=C\cup D, C\cap D=\varnothing$,,其中 C 是非空条件属性集合,D 是非空决策属性集合;V 代表所有属性取值集合,定义为:$V=\bigcup_{a\in A}V_a$,f 是作用于信息表的函数,定义为:$f(u_i,a)\in V_a$,其中对于任意的 $u_i\in U, a\in A$ 成立。

2.2　β 上近似和 β 下近似

在 Pawlak 的粗糙集理论(RST)中,对于一个信息表,可以定义对象集合 U 上面的等价关系。

定义 2.1　等价关系 R_A:

*)本文得到国家自然科学基金(No. 60503022)的资助。**吉阳生**　硕士,研究方向是 Rough 集和数据挖掘;**商琳**　副教授,研究方向是人工智能,软计算和 Rough 集。

$R_A=\{(u_i,u_j)\mid \forall a\in A, f(u_i,a)=f(u_j,a)\}$

基于等价关系 R_A，可以得到等价类集合 $E(A)=U/R_A=\{E_1,E_2,\cdots,E_n\}$。

可变精度粗糙集理论(VPRS)对于上、下近似的定义引入了参数 β。在 Ziarko [3]中，取值范围定义在[0,0.5)；而 An[7]和 Beynon[4]认为，取值的范围定义在(0.5,1]。本文中的 β 定义为对象正确分类的比例，取值在(0.5,1]。Slezak[8]定义了 VPRS 理论下，β 下近似和 β 上近似。

定义 2.2 集合 $X\subseteq U$ 基于属性集 $B\subseteq C$ 的 β 下近似和 β 上近似分别为：

$$\underline{B}_\beta(X):\ \underline{B}_\beta(X)=\bigcup_{P(X\mid E_i)\geqslant\beta}\{E_i\in E(B)\},$$

$$\overline{B}_\beta(X):\ \overline{B}_\beta(X)=\bigcup_{P(X\mid E_i)>1-\beta}\{E_i\in E(B)\}。$$

3 基于信息表 β 值非增量算法

Su[5]提出了一种不需要基于先验知识的计算 β 值非增量算法的思想，通过计算所有 C_i 对于 D_j 的分类错误率(其中 $C_i\in E(C), D_j\in E(C)$)，在这些分类错误率中选取最接近阈值 0.5 的错误率作为 β 的取值(Su[5]中 β 定义在[0,0.5))。为了与上文的假设一致，我们在下面的叙述中，将分类错误率用分类正确率重新定义，使得 β 定义在(0.5,1]。

定义 3.1 X 对于 Y 分类正确率 $C(X,Y)$：

$$C(X,Y)=\begin{cases}\dfrac{card(X\cap Y)}{card(X)} & \text{if } card(X)>0\\ 0 & \text{if } card(X)=0\end{cases}$$

其中，集合 $X\subseteq U, Y\subseteq U$。

基于定义 3.1 和 Su[5]中的 3.1，我们给出定理 3.1：

定理 3.1 当 β 取值为 $\xi(C,D)$ 时，信息表 S 条件属性集合 C 对于决策属性集合 D 分类质量达到最大。证明从略。

综上所述，当 β 取值为 $\xi(C,D)$ 时，信息表 S 条件属性集合 C 对于决策属性集合 D 分类质量达到最大。这就证明了 Beynon[4]中，β 取值和信息表分类质量关系的正确性。

4 β 值增量计算

4.1 属性模式

对于一个信息表 $S=\langle U, A, V, f\rangle$，每个 E_i 表示在属性集合 A 上取值都相同的对象的集合。和等价类概念相对应，属性模式同样描述了对于对象的划分能力。在 Inuiguchi 和 Miyajima[9]中形式化的定义属性模式：

定义 4.1

$Inf_A(u)=\bigcup_{a\in C\cup D}\{\langle a,f(u,a)\rangle\}$，其中 $u\in U$

同样，条件属性模式(condition attribute pattern)：$Inf_C(u)=\bigcup_{a\in C}\{\langle a,f(u,a)\rangle\}$，决策属性模式(decision attribute pattern)：$Inf_D(u)=\bigcup_{a\in D}\{\langle a,f(u,a)\rangle\}$。实际上，属性模式和等价类是一一对应的。

4.2 属性模式的排序

在定义属性模式的序之前，介绍构造可排序的属性模式(Ordering attribute pattern)的算法。假设信息表的属性是有顺序的：$A_i\in A, A_1=c_1,\cdots,A_{|C|}=c_{|C|}, A_{|C|+1}=d_1,\cdots, A_{|C|+|D|}=d_{|D|}$。

下面给出构造可排序属性模式的算法 OrderAttribute：

算法：OrderAttribute

输入：信息表 $S=\langle U, A, V, f\rangle$
输出：信息表 $S'=\langle U', A, V', f'\rangle$
步骤：
Step0：初始化 $S'=S$；
Step1：对于所有属性，重复 Step2 到 Step4；
Step2：如果，$S.A_i$ 取值是连续型的数值，$S'=discrete(S)$，转 Step3；如果 $S.A_i$ 取值是离散型数值，则转 Step3；
Step3：将 $S.f(u_j,A_i)$ 排序，把排名存入数组 order[]中(属性值相同则排名也相同)，然后 $S'.f(u_j,A_i)=order[j]$；
Step4：如果 $S.A_i=d_{|D|}$，则转 Step5；否则，转 Step1；
Step5：结束，输出 $S'=\langle U', A, V', f'\rangle$。

算法的 Step2 和 Step3，在我们的实验中，由于具体数据集的情况并没有执行。

定义 4.2 可排序的属性模式：$OInf_A(u)=\bigcup_{a\in C\cup D}\{\langle a,f'(u,a)\rangle\}$。

属性模式的排序，是指可排序的属性模式的排序。$OInf_c(u_i)$ 和 $OInf_c(u_j)$ 之间的顺序通过如下算法 OrderPattern[1]来确定。现在给出 OrderPattern 算法。

算法：OrderPattern

输入：$S'=\langle U', A, V', f'\rangle$，
$OCAPTable(OCAP, O_{cap})$，
$ODAPTable(ODAP, O_{dap})$；
输出：$S_{op}=\langle U_{op}, A_{op}, V_{op}, f_{op}\rangle$；
步骤：
Step0：扫描 S' 得到每个属性取值的个数，存入数组 $Num[]$；初始化 $S_{op}=\langle U_{op}, A_{op}, V_{op}, f_{op}\rangle$；第一次运行此算法时，初始化 OCAPTable(OCAP, O_{cap})，ODAPTable(ODAP, O_{dap})；
Step1：遍历每一个 $u_i\in U'$，重复 Step2 到 Step4；
Step2：计算 S_{op} 中：
$f_{op}(u_{opi},OCAP_i)=\sum_{A_j\in C}f'(u_i,A_j)Num[j]$
$f_{op}(u_{opi},DCAP_i)=\sum_{A_j\in D}f'(u_i,A_j)Num[j]$；
Step3：将 $f_{op}(u_{opi},OCAP)$ 和 $f_{op}(u_{opi},ODAP)$ 折半插入排序到 OCAPTable 和 ODAPTable 中；
Step4：如果所有的对象处理完毕，遍历 OCAPTable 和 ODAPTable，并且按顺序赋值给 O_{capi} 和 O_{dapi}，转 Step5；否则，转 Step1；
Step5：遍历 S_{op}：
$f_{op}(u_{opi},O_{capi})=$ OCAPTable.O_{capi}，
$f_{op}(u_{opi},O_{dapi})=$ OCAPTable.O_{dapi}；
Step6：算法结束，输出 S_{op}.

因此，$OInf_c(u_i)$ 和 $OInf_c(u_j)$ 之间的顺序，可以用 S_{op} 的 OCAP 值的大小来比较。

4.3 基于信息表的 β 增量计算

基于 4.2 节中的 OrderAttribute 和 Order-Pattern 算法，我们给出增量计算 β 的算法 ICObeta。

算法：ICObeta (Incremental Computation of β)

输入：信息表 S
输出：β
步骤：
Step0：如果是增量计算，转 Step6；
Step1：$S'=OrderAttribute(S)$；
Step2：$S_{op}=OrderPattern(S')$；
Step3：遍历 S_{op}：将 $count(OCAP_i\wedge ODAP_j)$ 填入 PatternIntersectionTable[i][j](|E(C)|*|E(D)|大小的矩阵)；将 $count(OCAP_i)$ 填入 OCAPNumberTable(|E(C)|*1 大小的矩阵)；
Step4：计算 Su[5]算法中的 m_1 和 m_2，得到 $\beta=min(m_1, m_2)$；
Step5：如果没有增量数据，转 Step10；否则，转 Step6；
Step6：$S'=OrderAttribute(S)$；
Step7：$S_{op}=OrderPattern(S')$；
Step8：遍历 S_{op}：将 $count(OCAP_i\wedge ODAP_j)$ 增量填入的 PatternIntersectionTable[i][j]；将 $count(OCAP_i)$ 增量填入的 OCAPNumberTable；
Step9：计算 Su[5]中的 m_1 和 m_2，得到 $\beta=\min(m_1, m_2)$；
Step10：结束，输出 β。

4.4 β 增量计算的时间复杂度分析

首先做一些说明和假设：(1)属性值是非负的离散型数值；(2)条件属性个数为 c，决策属性个数为 $d=1$，条件属性模

---

1) $A_{op}=\{OCAP, ODAP, O_{cap}, O_{dap}\}$
OCAP：Ordered Condition Attribute Pattern；ODAP：Ordered Decision Attribute Pattern；O_{cap}：Order of Condition Attribute Pattren；O_{dap}：Order of Decision Attribute Pattern；

式个数 p_c(与 c 有关的一个变量),决策属性模式个数 p_d(p_d 是一个常量);(3)非增量的信息表对象个数 n,增量的信息表对象个数 n',增量之后条件属性模式个数为 p'_c,决策属性模式个数为 p'_d;(4)c 远小于 n,看作于 n 无关的常量;(5)属性模式的排序使用折半插入排序的方法。

下面分析其中关键步骤的时间复杂度:

表 1 ICObeta 时间复杂度分析

算法步骤	时间复杂度	化简的时间复杂度
Step2	$O(cn+n\log n+n\log p_c+n)$	$O(n\log n)$
Step3	$O(n)$	$O(n)$
Step4	$O(p_c p_d)$	$O(p_c)$
Step7	$O(cn'+n'\log n'+n'\log p'_c+n')$	$O(n'\log n')$
Step8	$O(n')$	$O(n')$
Step9	$O(p'_c p'_d)$	$O(p'_c)$

因此,在增量阶段,ICObeta 方法总时间复杂度为 $O(n'\log n')$,比非增量方法的时间复杂度 $O((n+n')\log(n+n'))$ 低。

5 ICObeta 实验

实验采用 UCI 数据集的 1990 US Census 数据。1990 US Census 数据集,共有 69 个属性,26040 个对象。其中第一个属性是对象标识,采用第 2 个到第 68 个属性作为条件属性,第 69 个属性作为决策属性。实验中,选取前 20500 个对象作为输入计算 β 值。实验环境中,JDK 版本是 1.5.0_01,Matlab 版本是 6.5,操作系统是 WindowsXP。

5.1 实验目的

(1)通过非增量方法和 ICObeta 方法计算 β 值,比较两种方法得到的 β 值是一致的,证明 ICObeta 算法的正确性;

(2)通过统计程序中关键步骤的执行次数,拟合曲线,证明上文对于算法时间复杂度的分析是正确的;

(3)通过非增量方法和 ICObeta 关键步骤执行次数的比较,证明 ICObeta 方法计算的高效性。

5.2 实验结果

在实验结果图中,实线表示非增量方法得到的值,圆点表示 ICObeta 得到的值,横坐标表示参加计算的对象个数。

(1)图 1 所示,纵坐标表示计算得到的 β 值。图 1 表明,ICObeta 方法计算所得的 β 值,与非增量方法得到的 β 值是一样的,这说明 ICObeta 方法是正确的。

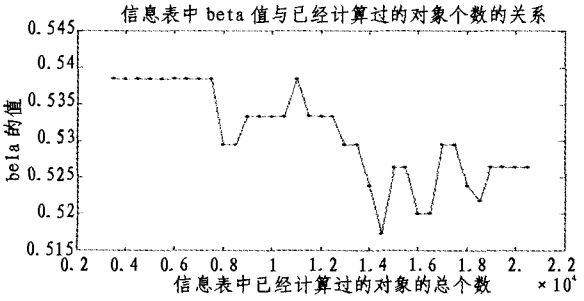

图 1 beta 值

(2)如图 4 所示,纵坐标表示计算 β 过程中关键步骤的执行次数。如表 2 所示,给出了对计算时间复杂度进行曲线拟合,用麦夸特法拟合函数得到的结果:$f(x)=ax\ln(x)+bx+c$。

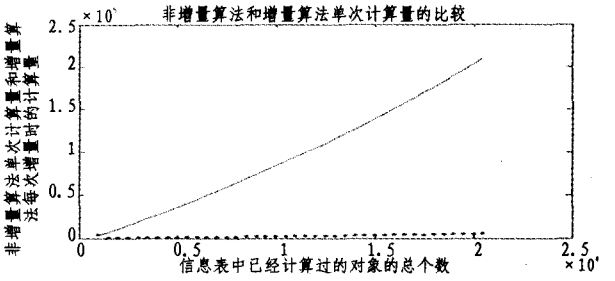

图 2 单次计算量对比

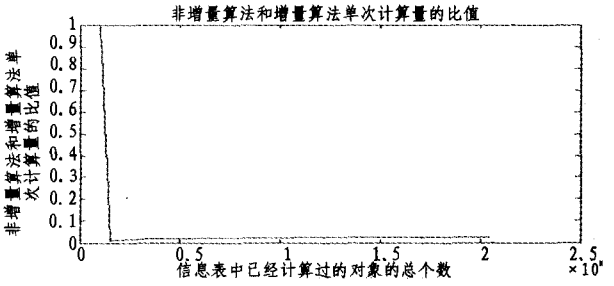

图 3 单次计算量的比值

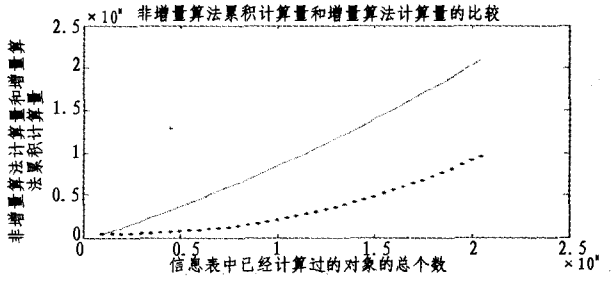

图 4 累积计算量的对比

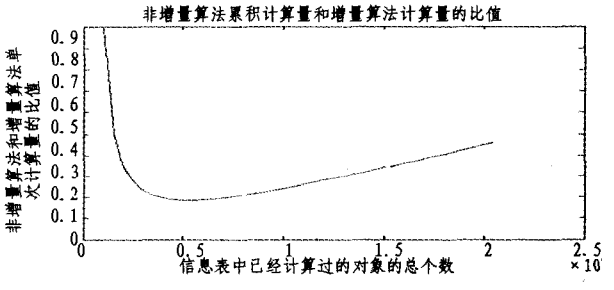

图 5 累积计算量的比值

表 2 非增量和增量计算复杂度曲线拟合

f(x)	a	b	c
非增量	2735.7	−17248.1	4838894.4
增量	4758.9	−43771.3	20299190.2

图 4 和表 2 表明,非增量方法和 ICObeta 方法拟合的曲线,与上文对于 ICObeta 计算时间复杂度的分析是一致的。

(3)图 2 和图 4 所示,纵坐标表示关键步骤的执行次数。图 3 和图 5 所示,纵坐标表示 ICObeta 与非增量方法关键步骤执行次数的比值。图 2~5 说明了,ICObeta 的计算量远比非增量方法的计算量小,证实了 ICObeta 方法的高效性。

(下转第 266 页)

需要对 key 做一下处理。我们令 $key=n+1-i$，对于所有的 $i\in[k+1,n]$，$key\in[1,n-k]$，然后同样令 $p=-\sigma(i)-b_2(\sigma(i))$，$d=\sigma(i)+b_2(\sigma(i))$，这样插入$(key,p,d)$三元组之后，对于给定 k，我们做 BoundedMin(BH2，$n-k+1$)，返回的就是我们需要的 $OPT_2(j)$。

4.3 算法的伪代码实现

伪代码的第 1 行为预处理，用于计算 $b_x(i)$，$b_x(i,j)$，$\Delta b_x(i)$，以及 $\mathrm{num}_x(\gamma)$。$\mathrm{num}_x(\gamma)$表示在序列 x 中 γ 的个数。我们将预先计算的表达式存于数组中，便于将来常数时间内存取。

第 2 行，对 $b_1(i)-b_2(i)$做一个桶排序；

第 3 行建一棵 van Emde Boas 树 T，T 的作用在于快速查询 4.1 节提到的分界点 k。

第 7～8 行，建立两个限定堆 BH1 和 BH2。

第 14～28 行是算法的主循环，这里我们对 j 遍历，对每个 j 求得$OPT(j)$。

第 29 行求对所有 j，$OPT(j)$的最大值。

5 时间和空间复杂度分析

5.1 空间复杂度

我们用到的数据结构包括线性表、限定堆和 van Emde Boas 树，空间复杂度都是 $O(n)$的，因此我们算法的总空间复杂度为线性。

5.2 时间复杂度

我们针对图 2 所示的伪代码逐行分析算法的时间复杂度。

第 1 行预处理，$b_x(i)$，$b_x(i,j)$，$\Delta b_x(i)$，$\mathrm{num}_x(\gamma)$都可以较容易地在线性时间内计算；

第 2 行，桶排序，同样是线性时间的算法；

第 4～6 行，对 van Emde Boas 树 T 作了 $O(n)$次 insert 操作，时间复杂度为 $O(n\mathrm{loglog}n)$；

第 9～12 行，在限定堆 BH1 和 BH2 上累计做了 $2n$ 个 insert 操作，时间复杂度为 $O(n\mathrm{loglog}n)$；

第 19～22 行，在限定堆 BH1 和 BH2 上累计做了 $2n$ 个 DecreasePriority 操作，时间复杂度为 $O(n\mathrm{loglog}n)$；

第 24 行，在 van Emde Boas 树 T 上做了 n 次 FindPrevious 操作，时间复杂度为 $O(n\mathrm{loglog}n)$；

第 25～26 行，在限定堆 BH1 和 BH2 上累计做了 $2n$ 次 BoundedMin 操作，时间复杂度为 $O(n\mathrm{loglog}n)$；

其余操作均为线性或常数的时间复杂度。

综上所述，我们算法的总的时间复杂度为 $O(n\mathrm{loglog}n)$。

结束语 本文针对 Brodal 等人提出的 3 字符字母表上的最长公共弱递增字串(LCWIS)问题，提出了一个新的时间复杂度为 $O(n\mathrm{loglog}n)$，空间复杂度为 $O(n)$的算法。我们算法利用了 van Emde Boas 树和限定堆作为主要的数据结构，时间和空间复杂度都是目前为止最好的。研究 LCWIS 问题，对于进一步研究 LCS 和 LIS 问题都有理论价值，并且在生物信息学上具有潜在应用价值。

尽管我们的算法时间复杂度是目前最好的，但该问题时间复杂度的上界依然未知，是否存在该问题的线性算法也未知。而对于该问题的扩展，4 字母或更多字母字符表上的 LCWIS 是否存在同样高效的算法等问题，依然需要进一步的研究。

参 考 文 献

1 Wagner R A, Fischer M J. The string-to-string correction problem. J ACM, 1974, 21(1): 168～173

2 Masek W J, Paterson M S. A faster algorithm computing string edit distances. J Comput System Sci, 1980, 20(1): 18～31

3 Schensted C. Longest increasing and decreasing subsequences. Canad J Math, 1961, 13: 179～191

4 Knuth D E. Sorting and searching. In: The Art of Computing Programming, Vol 3. Reading, MA: Addison-Wesley, 1973

5 Fredman M L. On computing the length of longest increasing subsequence. Discrete Mathematics, 1975, 11(1): 29～35

6 Yang I-H, Huang C-P, Chao K-M. A fast algorithm for computing a longest common increasing subsequence. Information Processing Letters, 2005, 93(5): 249～253

7 Chan W-T, Zhang Y, Fung S P Y, et al. Efficient algorithms for finding a longest common increasing subsequence. Journal of Combinatorial Optimization, 2007, 13(3): 277～288

8 Brodal G S, Kaligosi K, Katriel I. Faster Algorithms for computing longest common increasing subsequences. In: 16th Annual International Symposium on Algorithms and Computation (ISAAC), Henan, China, 2005

9 Van Emde Boas P, Kaas R, Zijlstra E. Design and implementation of an efficient priority queue. Theory of Computing Systems, 1977, 10(1): 99～127

(上接第 230 页)

结论 本文提出了一个满足信息表最大确定性度量标准的选择 beta 的增量算法 ICObeta，定理 3.1 保证了 ICObeta 方法满足了信息表确定性度量的标准，实验结果证实了 ICObeta 的正确性和高效性。这种 β 选取方法，避免了先验知识对于 β 选取的干扰，能够让我们摆脱先验知识不足的束缚，高效的获取合适的 β 值。进一步的工作，将主要考虑信息表约简，以及规则生成的增量方法展开。

参 考 文 献

1 Pawlak Z. Rough Sets. International Journal of Information and Computer Sciences, 1982, 11(5): 341～356

2 张文修，吴伟志，梁吉业，李德玉. 粗糙集理论与方法. 北京：科学出版社，2001

3 Ziarko W. Variable precision rough set model. Journal of Computer and System Science, 1993, 46: 39～59

4 Beynon M. Reducts within the variable precision rough sets model: a further investigation. European Journal of Operational Research, 2001, 134(3): 592～605

5 Su Chao-Ton, Hsu Jyh-Hwa. Precision parameter in the variable precision rough sets model: an application. The International Journal of Management Science, 2006, 34(2): 149～157

6 Cheng Yusheng, Zhang Yousheng, Hu Xuegang. The Relationships Between Variable Precision Value and Knowledge Reduction Based on Variable Precision Rough Set Model. RSKT2006, Berlin: Springer-Verlag, Lecture Notes in Computer Science, 2006

7 An A, Shan N, Chan C, Cercone N, Ziarko W. Discovering rules for water demand prediction: an enhanced rough-set approach. Engineering Applications in Artificial Intelligence, 1996, 9(6): 645～653

8 Slezak D, Ziarko W. Variable Precision Bayesian Rough Set Model. Lecture notes in computer science, 2003, 2639: 312～315

9 Inuiguchi M, Miyajima T. Variable Precision Rough Set Approach to Multiple Decision Tables. Lecture notes in computer science, 2005, 3641: 304～313