粗糙集理论 离散化问题 知识获取 (22)

计算机科学 2000Vol. 27№. 12

粗糙集理论中的离散化问题*)

Discretization in Rough Set Theory

89-94 TP18

侯利娟 王国胤 聂 能 吴 渝

(重庆邮电学院计算机科学与技术研究所 重庆 400065)

Abstract In this paper, the discretization problem in Rough Set is discussed. We do classification research on the discretization algorithm from different viewpoints. Several discretization algorithms are introduced and discussed. We improve the greedy algorithm in different ways. We present a strategy for discretization based on the importance of condition attributes. In the end, we present the goal of discretization based on experiments.

Keywords Rough Set, Discretization, Indiscernible relation, Cut core

一 引言

数据分析及数据挖掘，是一个重要的正在迅速发展的研究课题。波兰科学家 Z. Pawlak 于 1982 年提出的粗糙集(Rough Set)理论正是解决这一问题的新理论，它可用于处理决策信息表中的不确定知识，并用规则的形式表达，是一种有效的知识获取工具。在 Rough Set 中，数据约简是非常重要的，包括属性约简和值约简，运用粗糙集理论处理决策表时，要求决策表中各值用离散值表达。如果某些条件属性或决策属性的值域为连续值(如浮点型)，则在处理前必须经过离散化，这就是粗糙集理论中的一类重要研究课题——数据离散化问题。数据的离散化问题属于粗糙集理论中的预处理问题(决策表的补齐和数据的离散化)之一，在粗糙集理论分析的其化环节(如属性约简、值约简)之前进行。

二 离散化问题的描述

$S=(U, A\cup\{d\})$是一个决策表，$U=\{x_1,\cdots,x_n\}$是有限的对象集合即论域，$A=\{a_1,\cdots,a_k\}$为条件属性的集合，d 为决策属性。对于任意 $a\in A$，有信息函数 $U\to V_a$，V_a 是属性 a 上的值域。对 d 来说，有 $V_d=\{1,\cdots,r(d)\}$，$r(d)$为决策种类的个数。任意的(a,c)，其中 $a\in A$，$c\in R$(R 为实数集)，c 被叫做是 a 上的一个断点。对于 $a\in A$，在 $V_a=[l_a,r_a]\subset R$ 上的任意一个断点集合 $\{(a,c_1^a),(a,c_2^a),\cdots,(a,c_{d_a}^a)\}$定义了 V_a 上的一个分类 P_a，$P_a=\{[c_0^a,c_1^a),[c_1^a,c_2^a),\cdots,[c_{d_a}^a,c_{d_a+1}^a]\}$，$l_a=c_0^a<c_1^a<c_2^a<\cdots<c_{d_a}^a<c_{d_a+1}^a=r_a$，$V_a=[c_0^a,c_1^a)\cup[c_1^a,c_2^a)\cup\cdots\cup[c_{d_a}^a,c_{d_a+1}^a]$。因此，任意的 $P=\cup_{a\in A}P_a$ 定义了一个新的决策表 $S^P=(U, A^P\cup\{d\})$，$A^P=\{a^P: a^P(x)=i\Leftrightarrow a(x)\in[c_i^a,c_{i+1}^a)\}$对于 $x\in U$，$i\in\{0,\cdots,k_a\}$。即经过离散化之后，原来的信息系统被一个新的信息系统所代替。

离散化本质上可归结为利用选取的断点来对条件属性构成的空间进行划分的问题，把这个 n(n 为条件属性的个数)维空间划分成有限个区域，使得每个区域中的对象的决策值相同。假设某个属性有 m 个属性值，则在此属性上就有 $m-1$ 个断点可取，随着属性个数的增加，可取的断点数将随着属性值的个数呈几何增长，选取断点的过程也是合并属性值的过程，通过合并属性值，减少属性值的个数，减小问题的复杂度，这也有利于提高知识获取过程中所得到的规则知识的适应度。

三 离散化问题的分类

关于连续数据的离散化并不是一个新课题，早在粗糙集理论出现前，由于计算机对数值计算的要求，人们就对离散化(或称量化)问题进行了广泛研究，取得了大量研究成果。但是，离散化问题并不是一个各学科可以完全通用的研究课题，实际上它在不同领域中有自己独特的要求和处理方式；比如图像压缩中的量化问题，要求量化后的信息熵最小，这样在熵编码时可以得到最小的压缩图像长度；而运用粗糙集理论对决策

*)本项目得到国家自然科学基金和重庆市应用基础研究基金资助。侯利娟 硕士研究生，主要研究方向为 Rough Set 理论及应用；王国胤 博士，副教授，主要研究方向为 Rough Set 理论，知识获取，神经网络等；聂 能 教授，主要研究领域为计算机应用、管理信息系统等；吴 渝 博士，主要研究方向为 Rough Set 理论，小波分析等。

表进行分析和知识获取，是在决策表表达的不可分辨关系上进行的，对离散化的信息熵没有明确要求。

目前国际上针对粗糙集理论中的离散化问题也提出了一些有价值的研究成果，大致可以分为两类，其中一类基本上是很少或不考虑粗糙集理论的特殊性，而是把其它学科中的离散化方法借用到粗糙集理论上来，因此离散化效果并不突出；另一类就是注意到了粗糙集理论对决策表的特殊要求，采取了结合方法来解决离散化问题。我们主要考虑后一种方法。

粗糙集理论和决策表相结合的离散化算法中，根据进行离散化处理时是否考虑到信息系统的具体的属性值，可把离散化算法分为两类："非参照性的离散化算法"和"参照性的离散化算法"。"非参照性的离散化算法"在离散化过程中很少考虑或不考虑信息系统中具体的属性值。而"参照性的离散化算法"是参照信息系统中具体的属性值来进行的。根据离散化过程是否改变信息系统原有的不可分辨关系（我们主要考虑离散化后的信息系统在原信息系统的基础上是否引入新的冲突），可以把离散化算法分为"改变不可分辨关系的离散化算法"和"不改变不可分辨关系的离散化算法"。根据选取断点的过程是从包含所有可能断点的断点集中逐步删除不必要的断点得到离散化结果，还是一开始设断点集为空集，逐步增加候选断点得到离散化结果，又可以把离散化过程分为"逐步删除断点"和"逐步增加断点"的离散化算法。

四　离散化算法的分析

1. 等距离划分、等频率划分[5]

这两种算法需要人为地规定划分的维数，或者需要用户预先给定一个参数。根据给定的参数将各属性的值域按等距离或者等频率（属性值出现的频率）划分为几个离散的区间。离散化过程中几乎不考虑信息系统的具体的属性值，一次得到所有的断点值，不考虑信息系统的不可分辨关系，利用得到的断点离散原有的信息系统往往会改变信息系统原有的不可分辨关系，离散化处理结果的质量没有保障。

2. NaiveScaler 算法

Rough Set 理论的优势在于它不需要先验知识，可完全从数据或经验中获取知识，生成决策规则。NaiveScaler 算法不需要额外的参数，能够根据信息系统或数据库本身进行离散化处理，并在波兰华沙大学与挪威科技大学联合开发的 Rosetta 软件[5]中实现。NaiveScaler 算法对每个属性，根据属性值由小到大的顺序对决策表中的实例进行排序，然后进行判断，对于两个相邻实例，在属性值和决策值都不同的情况下，选取两个属性值的平均值作为断点值。该算法根据具体的条件属性值和决策属性值选取断点值而不是盲目地选取断点值。不考虑信息系统的不可分辨关系，随着数据库本身的数据的排列情况不同，得到的断点也不相同。更坏的情况，有些根据不可分辨关系应该选取的断点极可能被忽略掉。NaiveScaler 算法选取断点的过程是一开始设断点集为空集，逐步增加断点得到离散化结果，得到的断点数目很多。

3. SemiNaiveScaler 算法

在 NaiveScaler 算法的基础上，Rosetta 软件实现的 SemiNaiveScaler 算法是通过对每个由 NaiveScaler 算法得到的断点的两个相邻属性值所属的等价类所对应的决策中出现频率最多的决策值的集合是否存在包含关系进行判定而决定是否选取此断点，若两个集合存在包含关系，则不选取此断点，否则选取此断点。候选断点的选取和判定是根据具体的属性值进行的，得到的断点是 NaiveScaler 算法得到的断点的子集，可能改变信息系统的不可分辨关系。

4. 布尔逻辑和粗糙集理论相结合的离散化算法[1]

此算法可以根据给出的信息系统求出所有可能的断点集，而且采用任意的一种断点集，得到的新的信息系统不会改变原信息系统的不可分辨关系。在求取实际的断点集时，采取的是贪心算法。根据断点值的重要性逐步把断点加入到断点集中。贪心算法[1]如下：

首先构造一个新的信息表 $S^*=(U^*,A^*)$：

· $U^*=\{(x_i,x_j)\in U\times U: d(x_i)\neq d(x_j)\}$；

· $A^*=\{P_r^a: a\in A, r$ 是属性 a 的第 r 个断点$[c_r^a, c_{r+1}^a)\}$；

对于任意 $P_r^a\in A^*$，如果$[c_r^a,c_{r+1}^a)\subseteq[\min(a(x_i),a(x_j)),\max(a(x_i),a(x_j)))$

那么 $P_r^a((x_i,x_j))=1$；

否则 $P_r^a((x_i,x_j))=0$；

第 1 步　根据原来的信息系统 S 构造新的信息系统 S^*。

第 2 步　初始化断点集 CUT=Φ。

第 3 步　选取所有列中 1 的个数最多的断点加入到 CUT 中，去掉此断点所在的列和在此断点上值为 1 的行。

第 4 步　如果信息系统 S^* 中的元素不为空，则转第 3 步。否则停止。此时 CUT 即是所求的断点集。

此算法时间复杂度 $k*n^3$，空间复杂度为 n^2*k（n 为实例总数，k 为属性个数）。在信息系统的数据最大的情况下是不可取的。Nguyen S. H.，Nguyen H. S. 提出了一种有效的能够减小空间复杂度和时间复杂度的 Nguyen S. H 改进的贪心算法[2]，通过计算能够被给定的断点区分开的实例对的个数来代替贪心算法[1]的决

策表中各列的1的数目来衡量各个断点的重要性，再根据断点值的重要性逐步把断点加入到断点集中。Nguyen S. H 改进的算法[5]选择断点的本质和贪心算法[3]是一样的，因此得到的断点集和贪心算法[3]是相同的，其优势在于空间复杂度为 $k^2*n*\log n$，时间复杂度为 $k*n$（n 为实例总数，k 为属性个数），适合离散数据量大的信息系统。

布尔逻辑和粗糙集理论相结合的离散化算法是粗糙集理论中离散化算法在思想上的重大突破，是让其中一个断点或几个断点去区分两个实例的不同的不可分辨关系。此种算法的思想是首先在保持信息系统的不可分辨关系不变的前提下，尽量以最小数目的断点能够把所有实例间的分辨关系区分开。

五　贪心算法的改进

为了求得最小数目的断点集，贪心算法[3]每次取重要性最高的断点。断点的重要性是以各列中1的数目来衡量的，1的个数多，则断点的重要性高。当两列1的个数相同时，没有提出解决的办法。如文[1]中的 S* 表：

U*	P_1^1	P_2^1	P_3^1	P_4^1	P_1^2	P_2^2	P_3^2	各行1的数目
x_1,x_4	1				1	1		3
x_1,x_2	1	1					1	3
x_1,x_5	1	1	1					3
x_2,x_4		1	1		1			3
x_2,x_6		1	1	1	1	1	1	6
x_2,x_7		1			1			2
x_3,x_4			1			1	1	3
x_3,x_6			1	1				2
x_3,x_7						1	1	2
x_4,x_5						1		1
x_5,x_6				1			1	2
x_5,x_7			1			1		2
各列1的数目	3	5	6	3	4	6	5	

（注：上表中未填值的表格单元均取值为0，最后一行和最后一列是附加的。）

当 P_1^1 和 P_2^1 两列1的个数相同时，究竟应该选择哪一个呢？

对断点重要性的度量，还可以有一些辅助方法，S* 表中每一行1的数目表示了能够区分该行相应的两个实例的候选断点数目，若该数目较小，则说明相应的断点重要。

当有两列1的个数相同时，把断点所在的列值为1的行的1的数目相加，和值越小，则说明断点重要性越高。本例 P_1^1 所在的列值为1的行的1的数目加起来为（3+3+6+3+2+2）=19，P_2^1 所在的列值为1的行的1的数目加起来为（3+6+3+2+1+2）=17。因此我们可以认为 P_2^1 比 P_1^1 重要。由此，在原来贪心算法的基础上，得到改进算法1。

改进算法1

第1步　根据原来的信息系统S构造新的信息系统 S*。

第2步　初始化断点集 CUT=Φ。

第3步　选取所有列中1的个数最多的断点加入到CUT中，去掉此断点所在的列和在此断点上值为1的行；当有一个以上的断点的1的个数相同时，把列对应的断点所在的列值为1的对应的行的1的数目相加，取和最小的断点。

第4步　如果信息系统 S* 中的元素不为空，则转第3步；否则停止。此时 CUT 即是得到的断点集。

上面算法中，衡量断点的重要性是以列的1的个数多少作为主要标准的。

S* 表中每一行1的数目表示了能够区分该行相应的两个实例的候选断点数目，若该数目较小，则说明相应的断点重要。最特殊情况是该值取值为1，这表明此列对应的断点是唯一能区分该行对应的两个对象的断点，因而在保持不可分辨关系不变的大前提下进行离散化处理，这个断点是必不可少的。借鉴属性约简中对属性核的定义概念[4]，我们在这里特别定义断点核为所有能唯一区分两个对象的必不可少的断点所组成的集合，记为 CUT_{core}。

下面的算法是从取行为1的个数最少的断点开始。

改进算法2

第1步　根据原来的信息系统S构造新的信息系统 S*。

第2步　初始化断点集 CUT=Φ。

第3步　首先取断点核加入到 CUT 中，去掉断点核所在的列和在此断点上值为1的行。

第4步　重新计算各行列为1的个数，然后把行为1的数目最少的行的列值为1的断点对应的行的1的数目相加，取总和最小的断点加到断点集 CUT 中。去掉此断点所在的列和在此断点上值为1的行。

第5步　如果信息系统 S* 中的元素不为空，则转第4步；否则停止。此时 CUT 即是得到的断点集。

这两种算法分别从不同的角度出发对贪心算法进行改进，来对信息系统进行离散化。

在信息系统中，每个条件属性所占的地位不一定相同，一些条件属性占的地位较高，而另外的条件属性占的地位较低。如果我们能对信息系统的属性重要性作出评价，就可以充分利用重要性高的属性取值来优

化决策，在离散化处理过程中也可以借助这一信息来指导断点的选择过程。

六 基于属性重要性的离散化算法

·属性的重要性

信息系统的属性重要性是建立在分类能力上的。为了衡量条件属性的重要性程度，我们可从表中删除一些属性，再来考察信息系统的分类会产生怎样的变化：如果去掉某属性会相应地改变分类，则说明该属性的重要性高；反之说明该属性的重要性低。分类能力可通过分类质量 r_R 来衡量[3]，$r_R \overset{def}{=} card(pos_R(D))/card(U)$。属性 a 的重要性为：

$$gain = r_P - r_{R-\{a\}}$$

表 1 信息系统 1

U	a	b	d
x_1	0.8	2	1
x_2	1	0.5	0
x_3	1.3	3	0
x_4	1.4	1	1
x_5	1.4	2	0
x_6	1.3	1	1
x_7	1.6	3	1
x_8	4	3	1

表 2 信息系统 2

U	a	b	d
x_1	0.8	2	1
x_2	1	1	0
x_3	1.3	3	0
x_4	1.4	1	1
x_5	1.4	2	0
x_6	1.3	1	1
x_7	1.6	3	1
x_8	4	3	1

例：计算表 1 所示信息系统 1 的条件属性的重要性。其中：$C=\{a,b\}$，$D=\{d\}$，易得：

$U/ind(a,b)=\{\{1\},\{2\},\{3\},\{4\},\{6\},\{7\},\{8\}\}$

$U/ind(a)=\{\{1\},\{2\},\{3,6\},\{4,5\},\{7\},\{8\}\}$

$U/ind(b)=\{\{1,5\},\{2\},\{3,7,8\},\{4,6\}\}$

因此

$$r_C(D)=card(pos_C(D))/card(U)=card(\{1,2,3,4,5,6,7,8\})/card(U)=8/8=1$$

$$r_{C-a}(D)=card(pos_{C-a}(D))/card(U)=card(\{2,4,6\})/card(U)=3/8=0.375$$

$$r_{C-b}(D)=card(pos_{C-b}(D))/card(U)=card(\{1,2,4,7,8\})/card(U)=5/8=0.625$$

属性 a，b 的重要性分别为：

$$r_C(D)-r_{C-a}(D)=1-0.375=0.625$$

$$r_C(D)-r_{C-b}(D)=1-0.625=0.375$$

由此可见，属性 a 比属性 b 重要。

有了属性重要性程度的度量，我们再来介绍基于属性重要性的离散化算法。

·基于属性重要性的离散化算法

第 1 步 初始化断点集，CUT＝全部候选断点的集合。

第 2 步 首先根据条件属性的重要性由小到大对条件属性 $V_i(i=1\cdots n)$ 进行排序。在属性重要性相同的情况下，按条件属性断点个数由多到少进行排序。

第 3 步 对属性 $V_i\in A$，进行下面的过程。

第 4 步 对属性 V_i 中的每一个断点 $C_{ij}(j=1,\cdots,N_i)$，N_i 为属性 V_i 上的断点数目。考虑它的存在性：

把信息系统中与 C_{ij} 相邻两个属性值较小值改为较大值；

如果信息系统不引入冲突，则 $CUT=CUT/C_{ij}$，信息系统发生改变；

否则，$CUT=CUT$，把信息系统还原。

对属性 V_i 的每个断点经过上述的处理后，转向下一个属性。直至处理完所有属性的所有断点。

第 5 步 此时 CUT 即是所求的断点集。

该算法通过对每一个断点进行判定，去掉冗余的断点，从而得到简化的信息系统。

下面以信息系统 1 为例来说明该算法。

信息系统 1 的候选的断点集为：

$C_a=\{(a,0.9),(a,1.15),(a,1.35),(a,1.5),(a,2.8)\}$；

$C_b=\{(b,0.75),(b,1.5),(b,2.5)\}$。

经过计算，属性 a 比 b 重要。

我们从属性 b 的断点开始判定：

首先考虑断点值 0.75，它的相邻属性值为 0.5 和 1，把属性值 0.5 改为 1 后，不引起信息系统的冲突，则断点值 0.5 是多余的，可以被去掉，得到如表 2 所示的信息系统 2。

在信息系统 2 的基础上，接着考虑断点 1.5，1.5 的相邻属性值为 1 和 2，当把 1 改为 2 后实例 x_4 和 x_5 将会冲突，所以断点 1.5 是为保持信息系统的不可分辨关系所必须具有的断点值，信息系统 2 不变。

表 3 信息系统 3

U	a	b	d
x_1	0.8	3	1
x_2	1	1	0
x_3	1.3	3	0
x_4	1.4	1	1
x_5	1.4	3	0
x_6	1.3	1	1
x_7	1.6	3	1
x_8	4	3	1

对于断点 2.5 来说，相邻的属性值为 2 和 3，把信息系统 2 中属性 b 的属性值 2 改为 3，不引起信息系统的冲突，断点值 2.5 是多余的，可以被去掉，由信息系

统 2 得到如表 3 所示的信息系统 3。

同样，对属性 a 的断点 0.9、1.15、1.35、1.5、2.8 进行判定，求得最终的断点集 $CUT=\{(a,1.15),(a,1.5),(5,1.5)\}$。得到如表 4 所示的信息系统 4。

表 4 信息系统 4

U	a	b	d
x_1	1	3	1
x_2	1	1	0
x_3	1.4	3	0
x_4	1.4	1	1
x_5	1.4	3	0
x_6	1.4	1	1
x_7	4	3	1
x_8	4	3	1

用最终得到的断点集对原始信息系统进行离散化处理，得到离散化后的信息系统 5 如表 5 所示（属性 a 上，0 表示原始信息系统在此实例上的属性值在区间 $(-\infty,1.15)$ 上，1 表示原始值在区间 $[1.15,1.5)$ 上，2 表示原始值在区间 $[1.5,\infty)$ 上，属性 b 上，0 表示原始值在区间 $(-\infty,1.5)$ 上，1 表示原始值在区间 $[1.5,\infty)$ 上）。

本算法特点在于离散化过程始终不改变信息系统的不可分辨关系，由于断点的判定是从它所属的属性的重要性由小到大的顺序进行的，重要性较小的属性先被判定，其被淘汰的可能性更大些，同时离散化的过程又是属性约简的过程，之所以从属性重要性小的那些属性开始，是为了避免把属性重要性较大的那些属性上的断点去掉，该离散化算法从候选的断点集开始，一步步地筛选掉不必要的属性值断点，断点数目由多到少，直到剩下的断点都是必须的，即每个断点都不能被去掉，如果去掉，那么得到的新的信息系统的不可分辨关系将发生改变。

表 5 信息系统 5

U	a	b	d
x_1	0	1	1
x_2	0	0	0
x_3	1	1	0
x_4	1	0	1
x_5	1	1	0
x_6	1	0	1
x_7	2	1	1
x_8	2	1	1

·算法的复杂度分析

假设每个条件属性平均有 $m+1$ 个不同的属性值，分别为 $V_1,V_2,\cdots,V_{m+1}$，所对应的属性值的个数为 $n_1,n_2,\cdots,n_{m+1}$，则此属性有 m 个候选断点，实例总数 $n=n_1+n_2+\cdots+n_{m+1}$，假设被判定的断点的两个相邻属性值为 v_i,v_{i+1}，算法第 4 步要判定此断点所需的冲突检测次数为 n_i*n_{i+1}，远小于 n^2。所以算法的时间复杂度远小于 n^2*m*k，k 为属性个数。存储空间只需要一个辅助的信息系统，其空间复杂度为 $n*k$。该算法的优点在于离散化后的信息系统中的属性个数少，离散化的过程同时又是属性约简的过程。

七 实验结果

为了说明这些离散化方法的有效性和性能，我们进行了规则知识获取实验。首先用不同的离散化方法将原始数据进行离散化处理，然后用 Rosetta 软件的 Johnson algorithm 算法进行约简和规则获取，最后对所获得的知识进行测试。实验中，采用了 5 组不同的实验数据，每组数据中，随机选一半用于学习，利用所得到的断点对全部数据进行测试，识别结果如表 6，断点结果如表 7 所示。

表 6 识别结果

数据集	样本数	贪心算法			属性重要性算法			改进算法 1			改进算法 2		
		识别率	误识率	拒识率	识别率	误识率	拒识率	识别率	误识率	拒识率	识别率	误识率	拒识率
Iris	150	100%	0%	0%	94%	5%	1%	99%	1%	0%	84%	0%	16%
Whole	150	96%	4%	0%	96%	4%	0%	96%	4%	0%	59%	0%	41%
ecoli	336	73%	5%	22%	73%	9%	18%	81%	7%	12%	56%	3%	41%
Glass	214	69%	6%	25%	63%	15%	22%	68%	10%	22%	60%	7%	33%

表 7 断点结果

数据集	原条件属性数	贪心算法		属性重要性算法		改进算法 1		改进算法 2	
		剩余条件属性数	断点	剩余条件属性数	断点数	剩余条件属性数	断点数	剩余条件属性数	断点数
Iris	4	3	5	3	8	2	5	4	17
Whole	4	2	2	1	2	2	2	4	15
ecoli	7	5	16	4	26	5	13	6	45
Glass	9	7	10	5	22	8	11	9	32

从上面的识别结果中，我们可以看出这四种离散化算法都是有效的。前三种算法效果大致相当，改进算法2效果最差，根据对离散化问题的分析，我们再把识别结果和断点结果结合起来看，改进算法2得到的断点数目和剩余的属性个数较多，把学习实例所属的属性空间划分得太细，以致得到的规则的适应度较小，用得到的规则去测试时，导致测试数据中拒识或误识的比例较大（这说明以 S^+ 中行的1的个数为主来度量断点的重要性不合适）。而基于属性重要性的算法虽然得到的断点较多，但是剩余属性较少，从而减小了学习实例的空间维数。贪心算法和改进算法1在剩余属性个数和断点个数两方面相对折中，也是有效的离散化算法。

结论 通过对离散化问题的系统研究，我们认为，为了能够最大限度地提高识别正确率，减小误识率和拒识率，在进行数据离散化处理时，应该在保持信息系统不可分辨关系不改变的前提下，尽量减小信息系统的复杂程度，使离散化后的信息系统的属性个数尽量少，每一属性上的断点尽量少，由于各信息系统蕴涵的领域知识不同，不可能提出一个离散化算法针对所有的信息系统都是最优的。前面提出的几种离散化算法是从不同的出发点出发的，有其各自的特点，在进行知识获取时，可根据具体情况选择合适的离散化算法。

参考文献

1 Nguyen H S, Skowron A. Quantization of real values attributes, rough set and boolean reasoning approaches. In: Proc. of the Second Joint Annual Conference on Information Science. Wrightsville Beach, NC, 1995. 34～37

2 Nguyen S H, Nguyen H S. Some Efficient Algorithms for Rough Set Methods. In: Proc. of the Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems. Granada, Spain, 1996. 1451～1456

3 曾黄麟. 粗集理论及其应用—关于数据推理的新方法. 修订版，重庆：重庆大学出版社，1998. 83～87

4 常犁云，王国胤，吴渝. 一种 Rough Set 理论的属性约简及规则提取方法. 软件学报，1999，10(11)：1206～1221

5 Knowledge Systems Group. Rosetta Technical Reference Manual, 1999

（上接第107页）

据结构而已。如何管理、保存和维护这一数据结构对进程而言是透明的。对进程而言，无论是文件服务器所维护的文件还是其他进程所维护的数据结构，它们都是进程可以通过进程通讯访问的被其他进程所维护的数据结构，进程感觉不到它们之间有任何差别。如果按照这种观点来看待文件，那么指令对文件直接寻址，就是指令对通过进程通讯访问的被其他进程所维护的数据结构的直接寻址。

在采用指令对文件直接寻址的操作系统中，就象在传统的操作系统中一样，仍然为进程构造逻辑空间，而且是两维的段页式逻辑空间，但逻辑空间中的数据如何存储和维护，则不再由操作系统来负责，而交给了运行于核外的其他进程。进程之间通过进程通讯交换数据，计算机系统中的内存全部用作进程访问其他进程所维护数据结构的缓冲区，通过把进程页表映射到缓冲区上为进程构造可直接访问其他进程数据的逻辑空间。

因此，指令对文件直接寻址实现了操作系统中数据存储和数据处理的分离，数据处理由一个进程实现，数据存储由另一个进程实现。如果数据存储由文件服务器管理，就是上面讨论的指令对文件的直接寻址。

结论 本文提出了指令对文件的直接寻址的概念和实现原理，并分析了和传统的操作系统相比所具有的优点，笔者认为：把逻辑空间和文件统一起来，取消逻辑空间的概念，实现指令对文件的直接寻址，使进程直接在文件之上运行，具有许多传统操作系统所没有的优点，值得在操作系统的设计中采用和推广，对操作系统的设计具有十分重要的意义。

参考文献

1 Tanenbaum A S. Distributed Operating System. Prentice Hall Inc, 1995

2 樊建平，李国杰. 并行操作系统的现状和发展趋势. 计算机研究与发展，1995(1)

3 陈华瑛. Mach 3.0 核心分析. 计算机研究与发展，1994(9)

4 陆桑璐，谢立. 进程的动态迁移技术. 计算机研究与发展，1997(9)

5 江南计算所，吉增瑞. 操作系统的并行处理. 计算机世界，1996