基于全局正区域不一致性的快速求核算法

赵 洁[1] 梁俊杰[2] 董振宁[1] 陈 旭[1] 唐德育[2]
(广东工业大学 广州 510520)[1] (华南理工大学 广州 510006)[2]

摘 要 首先基于改进的Hash和位运算设计了快速等价类与正区域算法,将其作为求核基础;然后设计基于全局正区域不一致性的快速求核算法。区别于现有算法在求核过程中需要反复多次求正区域,深入分析了核属性 a_i 的特征,捕捉两类 $C-\{a_i\}$ 所形成的正区域与全局正区域的不一致,不需要反复求完整的 $C-\{a_i\}$ 正区域,通过3个定理证明了基于全局正区域不一致性识别核属性的正确性和有效性。使用UCI中21个数据集、超高维和海量数据集进行全面检验,结果表明无论是多/少实体、多/少属性和有/无核的决策表,本算法在大部分情况下都优于现有同类算法,尤其适用于大型决策表。

关键词 粗糙集,核属性,全局正区域,不一致性
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.8.053

Global Positive Region Inconsistency Based Attributes Core Computation

ZHAO Jie[1] LIANG Jun-jie[2] DONG Zhen-ning[1] CHEN Xu[1] TANG De-yu[2]
(Guangdong University of Technology, Guangzhou 510520, China)[1] (South China University of Technology, Guangzhou 510006, China)[2]

Abstract This paper firstly proposed basic algorithms of positive region and equivalence class based on bit vector and improved Hash algorithm. Then the core attributes computation algorithm was designed based on global positive region inconsistency. Different from current algorithms which need to compute complete positive regions repeatedly when seeking attributes core, this paper studied the characteristics of core attributes, and caught the inconsistencies between the positive region of $C-\{a_i\}$ and the global positive region. The complete positive regions of $C-\{a_i\}$ don't need to be computed repeatedly. Global positive region inconsistency based attributes core recognition was proved by 3 theories. 21 data sets of UCI, ultra-high-dimensional data sets and massive data sets were used to test the algorithms proposed by this paper. And the results show the attributes core computation algorithm of this paper owns good performance no matter when the number of entities and attributes is more or less and especially is suitable for processing large decision table.

Keywords Rough set, Attributes core, Global positive region, Inconsistency

1 引言

粗糙集理论[1,2]是处理不精确、不一致、不完整信息的数学工具。属性约简是粗糙集理论研究的核心问题之一,约简算法的低效在一定程度上限制了粗糙集理论的广泛应用,研究高效的粗糙集约简算法有着重要意义。很多约简算法设计始于求核,其效率很大程度上取决于求核的时间复杂度,尤其当数据量巨大时,如何快速求核显得尤为重要。

近年来,基于粗糙集的研究向多方向发展,包括对扩展粗糙集下的约简算法的研究,如基于覆盖粗集[3,4]和模糊粗糙集[5-8];对约简算法中细节问题进行研究,如非完整不一致决策表下的约简问题[9]和启发式约简算法加速等[10]。基于差别矩阵的方法吸引了众多学者的关注[4,5,8,11-14]。基于正区域的方法起源较早,算法效率提高的难度越来越大,但该领域仍持续有新成果产生[15-19]。

近年来,众多学者围绕求核低效性的根源,从不可区分关系和正区域两个基本操作入手,通过各种方法不断降低正区域等算法的时间复杂度,进而设计快速求核和约简算法。刘少辉等[15]基于快速排序设计的正区域算法,时间复杂度为 $O(|A||U|\log|U|)$,空间复杂度为 $O(|U|+1)$,同时给出求正区域的等价算法,但该算法通过 $POS_C(D)\neq POS_{C-\{a_i\}}(D)$ 求核,需要反复求等价类。徐章艳等[17]利用基数排序对 U/C 计算进行改进,时间复杂度为 $O(|C||U|)$,同时在求 $pos_P(D)$ 算法中不断地丢弃不影响正区域计算的实体,有效提高了算法效率。刘勇等[18]的算法充分利用了Hash的优势,把获取等价类的时间复杂度降为 $O(|C||U|)$。葛浩[20]提出一

到稿日期:2014-08-25 返修日期:2014-12-22 本文受国家自然科学基金资助项目:DS证据推理下抗信誉共谋攻击的行为信任研究(71401045),大学生创业创新训练项目:电子商务中抗信誉共谋欺诈的推理算法和模型研究资助。

赵 洁(1979-),女,博士,副教授,主要研究方向为智能算法;梁俊杰(1991-),男,硕士生,主要研究方向为数据挖掘;董振宁(1978-),男,博士,副教授,主要研究方向为供应链金融;陈 旭(1988-),男,硕士生,主要研究方向为数据分析;唐德育(1978-),男,博士,讲师,主要研究方向为智能算法。

种分布计数的基数排序方法，时间复杂度也为 $O(|C||U|)$，空间复杂度为 $O(|U|)$，算法实现的简易程度优于文献[17]。基于上述排序方法，葛浩[20]进一步提出基于冲突域的求核算法，其基本思想仍是判断 $POS_C(D)\neq POS_{C-\{a_i\}}(D)$，需要多次求等价类，但计算时把对 $|POS_C(D)|$ 的计算转换为求解 $|U|-|POS_C(D)|$，使求核更为简洁。随后葛浩等[21]通过在决策表中对正区域进行标记来进一步改进算法，得到了快速求核算法。葛浩提出的两个求核算法的时间复杂度为 $O(|C|^2|U|)$，空间复杂度为 $O(U)$。Hu 等[22]利用分而治之方法的思想，为求等价类和核设计了递归算法，其中正区域算法的时间和空间复杂度分别为 $O(|C||U|)$ 和 $O(|U|+p|C|)$，求核的时间和空间复杂度分别为 $O(|C|^2|U|)$ 和 $O(n+p\times m)$。其中 $p=\max(|V_i|)(1\leqslant i\leqslant|C|)$，$m=|C|$，判断依据仍是 $POS_C(D)\neq POS_{C-\{a_i\}}(D)$，需要多次求完整的正区域。

基于上述研究，本文深入研究全局正区域、核属性特征以及两者之间的关系，发现当使用启发规则计算 $C-\{a_i\}$ 的正区域时，若 a_i 是核属性，会使得同属于正区域的 x_i 和 x_j 无法辨识(决策值 $D(x_i)\neq D(y_i)$)，或者使属于正区域的 x_i 与不属于正区域的 x_j 无法辨识，从而产生不一致。通过上述分析，本文证明 3 个识别核属性的定理，并设计算法实时捕捉上述不一致情况，通过剪枝(见定义 6)，避免了反复求正区域判断 $POS_C(D)\neq POS_{C-\{a_i\}}(D)$ 的计算。本文首先借助位操作的快速特性并结合 Hash 优势设计正区域算法，并设计判断实体 x 是否属于正区域的算法，结合所证明的定理设计核属性算法，最差情况下时间复杂度为 $O(|C||U|^2)$，平均计算时间为 $O(|U||C|^2)-O(\frac{|C||U||Core(A)|}{2})$，空间复杂度为 $O(U)$。最后通过 UCI 中的多个数据集，并增加超高维和海量数据集来进行验证，结果表明本文求核算法是正确、高效的，适用于多种情况。

本文第 2 节简要介绍 Rough 集的基本概念；第 3 节首先基于 Hash 设计等价类和正区域算法；第 4 节给出并证明基于核属性特征识别的 3 个定理；第 5 节给出求核算法；第 6 节使用 UCI 中 21 个数据集、超高维和海量数据集进行验证，并分析实验结果；最后对全文进行总结。

2 Rough 集基本概念

本节给出粗糙集的基本概念，详见文献[1]。

定义 1(信息系统) 信息系统 $IS=(U,A,V,f)$，其中 U 是论域，是实体的集合，$U=\{x_1,x_2,x_3,\cdots\}$。A 是属性集，$A=\{a_1,a_2,\cdots,a_m\}$，$A=C\cup D$ 且 $C\cap D=\emptyset$，C 称为条件属性，D 称为决策属性。V 是属性的值域，$V=\{V_{a1},V_{a2},\cdots,V_{am}\}$。$f$ 是一个信息函数 $f:U\times A\in V$，$\forall a\in A,x\in U,f(x,a)\in V$，$f(x,a)$ 通常也记为 $a(x)$。假设 $B=\{a_1,a_2,\cdots,a_k\}\subseteq A$，$(a_1(x),a_2(x),\cdots,a_k(x))$ 记为 $B(x)$。

定义 2(不可区分关系) 对 $\forall B\subseteq A$，$Ind(B)=\{(x_i,x_j)|B(x_i)=B(x_j)\}$ 称为不可区分关系或等价关系，表示 (x_i,x_j) 关于属性集 B 是不可区分的。根据 $Ind(B)$ 可导出一个等价划分 U/B，该划分中包含对象 x 的等价类记为 $[x]_B$。

定义 3(全局正区域) 信息系统 $IS=(U,A,V,f)$，令 $P,Q\subseteq A$，$pos_P(Q)=\{x|[x]_P\subseteq[x]_Q\}$，$pos_P(Q)$ 称为 P 相对于 Q 的正区域，$pos_C(D)$ 称为全局正区域。

定义 4(约简) 在信息系统 $IS=(U,A,V,f)$ 中，若 $\forall B\subseteq C$，$POS_B(D)=POS_C(D)$，且对于 $\forall B'\subset B$，都有 $POS_{B'}(D)\neq POS_C(D)$，则称 B 是 C 相对于 D 的属性约简，记为 $B=red(D)$。

定义 5(核) C 中所有不可省略属性的集合称为 C 的核，即 $core(D)=\bigcap red(D)$。

3 基于位运算及改进 Hash 的正区域算法

基于正区域的求核和约简算法的效率在很大程度上取决于正区域算法，因此设计高效的正区域算法可有效提高求核和约简算法的效率。本节将给出基于位运算和改进 Hash 的正区域算法。本文例子使用的决策表如表 1 所列。

表 1 决策表

U	a_1	a_2	a_3	D	U	a_1	a_2	a_3	D
x_1	1	1	1	1	x_6	1	1	1	2
x_2	1	3	2	2	x_7	1	2	2	1
x_3	2	1	1	1	x_8	1	2	2	1
x_4	3	1	1	2	x_9	1	3	2	2
x_5	1	3	2	2					

3.1 基于改进 Hash 的等价类和正区域算法

现有算法把 Hash 应用于约简算法，一般将属性值字符串作为 key[18]。而本文采用数组存储 key，进一步改进 Hash，从而设计等价类及正区域算法。

算法 1 计算属性集 B 的等价类 U/B 及正区域

输入：IS=(U,A,V,f)，B(B⊆C)

输出：Ind(B)和|pos$_B$(D)|

1. 为 B 初始化一个 Hash 表 H，pos=0
2. 对每一个 $x_j\in U$，key=B(x_j)
   - 2.1 若 H 中无当前 key，则创建 h_k，h_k.count=0，$h_k=h_k\cup\{x_j\}$，h_k.cons=true，h_k.count++，h_k.dec=D(x_j)；
   - 2.2 若 H 中存在 h_k，h_k.count++，
     若 h_k.cons=true 且 D(x_j)≠h_k.dec，则 h_k.cons=false
3. 遍历对 $h_j\in H$，若 h_j.cons==true，pos=pos+ h_j.count
4. 返回计算后的 H 即 Ind(B)和|pos$_B$(D)|

Hash 表 H 中，对于每个分项 h_k，属性 $h_k.cons$ 标识 h_k 中实体是否属于正区域，$h_k.count$ 统计 h_k 中的实体数，$h_k.dec$ 记录 h_k 的 D 值。对每个 a_i，步骤 2 遍历所有实体，时间复杂度为 $O(|U|)$。本文采用数组存储 key，执行 put 和 get 方法的时间复杂度均为 $O(|key|)$，故步骤 2 中 2.1 和 2.2 的时间复杂度均为 $O(|key|)$，步骤 2 的时间复杂度为 $O(|U|)\times O(|key|)=O(|B||U|)$，最差情况下与文献[18]相同。数组键首次形成 key 的时间复杂度为 $O(|key|)$，与字符串方法相同，但数组键可利用原有键构造出新 key，复杂度为 $O(1)$。而字符串方法每次构造新 key 的复杂度均为 $O(|key|)$。使用算法 1 对表 1 计算等价类，结果如表 2 所列。

表 2 利用算法 1 对决策表 1 进行计算的结果

等价类	key	cons	count	dec	实体
h_1	[1 1 1]	F	2	*	x_1,x_6
h_2	[1 3 2]	T	3	2	x_2,x_5,x_9
h_3	[2 1 1]	T	1	1	x_3
h_4	[3 1 1]	T	1	2	x_4
h_5	[1 2 2]	T	2	1	x_7,x_8

3.2 正区域位向量算法

在求核过程中，需要多次检测实体是否属于正区域，因此

在正区域算法基础上，设计正区域位向量算法来判断实体 x 是否属于正区域。对于 $B\subseteq C$，若 $x_i\in pos_B(D)$，则令 $b_s^B(i)=1$，否则令 $b_s^B(i)=0$，则向量 $(b_s^B(n),b_s^B(n-1),\cdots,b_s^B(1))$ 称为位向量，用于表示 $pos_B(D)$。

算法 2 求给定属性集 $B\subseteq C$ 下的位向量 b_s^B

输入：Ind(B)，B⊆C//等价类用 Hash 表保存

输出：b_s^B

1. 初始化位向量 b_s^B(b_s^B(i)=0)

2. 遍历对 $h_k\in H$，若 h_k.cons==true，对每一个 $x_j\in h_k$，使 b_s^B(j)=1

上述算法步骤 2 搜索分项的时间复杂度是 $O(|U/B|)$，遍历正区域分项实体的时间复杂度为 $O(|pos_B(D)|)$，对位向量置 1 的时间复杂度为 O(1)。因此算法 2 总的时间复杂度是 $O(|U/C|)+O|pos_C(D)|$。在实际运行中，位向量的操作消耗时间非常少，这已在实验中得到验证。位向量使用字节数组存储位向量，从右边起存储，下标从 0 开始。实体 x_k 在 b_s^B 中的位置计算公式如下：

$$m=\lfloor(k-1)/8\rfloor,p=(k-1)\%8 \quad (1)$$

这表示 x_k 在数组的第 m 个字节，在该字节右边起第 p 位。

例如，利用算法 1 对表 1 进行计算，得到的 Hash 表 H 如表 2 所列，其中分项 $h_1=\{x_1,x_6\}$，$h_1.cons$=false，$h_1.count$=1，$h_1.desc$=*，$b_s^C(1)=0$，对 H 遍历完成后，结果如图 1 所示。

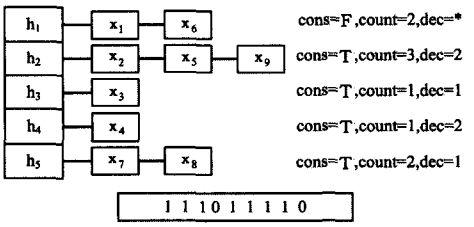

图 1 表 2 的 b_s^C

4 基于全局正区域不一致性的核属性识别定理

现有研究一般通过 $POS_C(D)\neq POS_{C-\{a_i\}}(D)$ 判断属性 a_i 是否为核属性，需要反复计算正区域。区别于上述方法，本文求核算法不需要多次计算正区域，通过深入分析核属性 a_i 的特征，捕捉两类 $C-\{a_i\}$ 所形成的正区域与全局正区域的不一致，一旦发现不一致的出现，即可停止搜索即剪枝，因此不需要反复求完整的 $C-\{a_i\}$ 正区域。下面给出剪枝的定义和定理证明。

定义 6(剪枝) 在求核属性中未遍历完所有实体，根据识别特征终止遍历的过程称为剪枝。

下面通过证明 3 个定理和 1 个引理来研究核属性的识别特征。

定理 1 设 $a_i\in C$，令 $B=C-\{a_i\}$。若 $\exists x,y$ 满足 $B(x)=B(y)$，$x\in pos_C(D)$，且 $D(x)\neq D(y)$，则 $a_i\in core(A)$。

证明：假设 $R\subseteq B=C-\{a_i\}$ 是一个约简，则由约简定义知 $pos_R(D)=pos_C(D)$。因为 $B(x)=B(y)$，所以 $R(x)=R(y)$，即 $y\in[x]_R$。又因为 $D(x)\neq D(y)$，故 $y\notin[x]_D$，从而 $[x]_R$ 不包含于 $[x]_D$，因此由正区域定义知 $x\notin pos_R(D)$。而由条件知 $x\in pos_C(D)$，则 $pos_R(D)\neq pos_C(D)$，因此假设 $R\subseteq B=C-\{a_i\}$ 是一个约简不成立。因此对于任一约简 R，均有 $a_i\in R$，即 $a_i=\bigcap red(A)$，因此 $a_i\in core(A)$。定理得证。

引理 1 若 $B\subset B'\subseteq C$，则 $pos_B(D)\subseteq pos_{B'}(D)$。

证明：任取 $x\in pos_B(D)$，则由正区域的定义知，$[x]_B\subseteq[x]_D$。任取 $y\in[x]_{B'}$，即 $B'(x)=B'(y)$。因为 $B\subset B'$，所以 $B(x)=B(y)$，故 $y\in[x]_B$，所以 $[x]_{B'}\subseteq[x]_B\subseteq[x]_D$，所以 $x\in pos_{B'}(D)$，因此 $pos_B(D)\subseteq pos_{B'}(D)$。

定理 2 设 $a_i\in C$，令 $B=C-\{a_i\}$。若 $\exists B(x)=B(y)$，$x\in pos_C(D)$，$y\notin pos_C(D)$，且 $D(x)=D(y)$，则 $a_i\in core(A)$。

证明：假设 $R\subseteq B=C-\{a_i\}$ 是一个约简，则由约简定义知 $pos_R(D)=pos_C(D)$。因 $B(x)=B(y)$，而 $R\subseteq B$，所以 $R(x)=R(y)$，即 $[y]_R=[x]_R$。因为 $x\in pos_C(D)=pos_R(D)$，所以由正区域定义知，$[x]_R\subseteq[x]_D$；已知 $D(x)=D(y)$，即 $[x]_D=[y]_D$，所以 $[y]_R=[x]_R\subseteq[x]_D=[y]_D$，所以由正区域定义知，$y\in pos_R(D)$。因为 $R\subseteq B$，所以由引理 1 知 $pos_R(D)\subseteq pos_B(D)$，所以 $y\in pos_B(D)$，这与已知 $y\notin pos_B(D)$ 矛盾，因此假设 $R\subseteq B=C-\{a_i\}$ 为一约简不成立。因此对于任一约简 R，均有 $a_i\in R$，即 $a_i=\bigcap red(A)$，所以 $a_i\in core(A)$。

由定理 1 和定理 2 得到定理 3。

定理 3 设 $a_i\in C$，令 $B=C-\{a_i\}$。若 $\exists B(x)=B(y)$，$x\in pos_C(D)$，$y\notin pos_C(D)$，则 $a_i\in core(A)$。

分析上述定理，其实质是识别忽略核属性时，全局正区域产生的两种不一致性：(1)同属于正区域的 x_i 和 x_j 无法辨识(决策值 $D(x_i)\neq D(y_i)$)；(2)属于正区域的 x_i 与不属于正区域的 x_j 无法辨识。本文根据上述定理，不需要反复求完整的正区域，从而快速求核。

5 基于全局正区域不一致性的求核算法

在上述定理基础上，给出求核算法。

算法 3 求 $core(A)$

输入：IS=(U,A,V,f)

输出：core(A)

1. 利用算法 1 和算法 2 获得 b_s^C

2. 对每一个 $a_i\in C$，B=C−{a_i}，初始化哈希表 H

3. 对每一个 $x_j\in U$，key= B(x_j)，有以下操作

//以当前属性外的所有属性值构造 key

4. 通过 b_s^C 判断，若 $x_j\notin pos_C(D)$

4.1 若 H 无当前 key 记录，创建分项 h_k 且令 $h_k=h_k\cup\{x_j\}$，h_k.cons=false

4.2 若 H 已有对应 h_k

若 h_k.cons=true，则 core(A)=core(A)∪{a_i}，i++，跳至步骤 2//按定理 3 剪枝

否则，$h_k=h_k\cup\{x_j\}$

5. 通过 b_s^C 判断，若 $x_j\in pos_C(D)$

5.1 若 H 无当前 key 记录，则创建分项 h_k 且令 $h_k=h_k\cup\{x_j\}$，h_k.cons=true

5.2 若 H 已有对应 h_k

若 $D(x_j)\neq h_k$.dec，则 core(A)=core(A)∪{a_i}，i++，跳至步骤 2;//按定理 1 剪枝

若 h_k.cons=false，则 core(A)=core(A)∪{a_i}，i++，跳至步骤 2;//按定理 3 剪枝

否则，$h_k=h_k\cup\{x_j\}$

6. 返回 core(A)

算法 3 中的 $h_k.cons$ 的含义同上。步骤 2 遍历所有属性一次，时间复杂度为 $O(|C|)$。步骤 3 忽略属性 a_i 和 D 构造 key，并遍历所有实体一次，其时间复杂度为 $O(|C||U|)$。步骤 4 和步骤 5 利用位向量判断实体 x 是否为 $pos_C(D)$ 的成员的时间复杂度为 O(1)，步骤 4 和步骤 5 均有 Hash 的 $put()$

操作，其时间复杂度为 $O(|C|)$。故最差情况下，算法 3 的时间复杂度为 $O(|C|)\times O(|U|)\times O(|C|)=O(|U||C|^2)$。本文依据定理 1 和定理 3 剪枝，遍历实体个数大大减少，在核属性平均分布情况下，计算中产生的不一致性实体出现在决策表中第 i 位的可能性是 $\frac{1}{|U|}$，则第 $i(2\leqslant i\leqslant|U|)$ 次进行剪枝的可能性是 $p=\frac{(|U|+2)(|U|-1)}{2|U|}$，故采用上述定理进行剪枝后的平均计算时间为 $O(|U||C|^2)-O(\frac{|C||U||Core(A)|}{2})$。

对表 1 求核过程举例如下，计算过程中需使用 b_s^C。

第一步忽略 a_1：

$h_1:\{\{x_1,x_3\},cons=\mathrm{F},dec=*\}$

$h_2:\{\{x_2\},cons=\mathrm{T},dec=2\}$

遍历至实体 x_3 时出现不一致，符合定理 3 的剪枝条件，故 $a_1\in core(A)$。

第二步忽略 a_2：

$h_1:\{\{x_1,x_6\},cons=\mathrm{F},dec=*\}$

$h_2:\{\{x_2,x_5,x_7\},cons=\mathrm{T},dec=2,1\}$

$h_3:\{\{x_3\},cons=\mathrm{T},dec=1\}$

$h_4:\{\{x_4\},cons=\mathrm{T},dec=2\}$

遍历至实体 x_7 时出现不一致，符合定理 1 的剪枝条件，故 $a_2\in core(A)$。

第三步忽略 a_3：

$h_1:\{\{x_1,x_6\},cons=\mathrm{F},dec=*\}$

$h_2:\{\{x_2,x_5,x_9\},cons=\mathrm{T},dec=2\}$

$h_3:\{\{x_3\},cons=\mathrm{T},dec=1\}$

$h_4:\{\{x_4\},cons=\mathrm{T},dec=2\}$

$h_5:\{\{x_7,x_8\},cons=\mathrm{T},dec=1\}$

第三步中不发生剪枝，故 $a_3\notin core(A)$。

因此对表 1 求核的结果为：$core(A)=\{a_1,a_2\}$。

6 实验与结果分析

本文使用 3 类数据集进行实验：1)UCI 中的 21 个数据集(见表 3)，对全部数据进行离散化处理；2)超高维数据集；3)海量数据集。用 Java 实现本文算法与文献中的求核算法(包括算法 a[15]、算法 b[20]、算法 c[21]、算法 d[22])，并进行比较，在 PC(AMD Phenom(tm) II X4 B32 Processor 2.90GHz，内存 4GB，Windows 8)上进行对比分析，每次实验均对决策集进行随机排序。全部求核的算法均在本文提出的 Hash 和位运算的基础上实现，以证明本文求核算法的高效性。

1. 使用 UCI 数据集的实验

使用 UCI 中的 21 个数据集对本文算法、算法 a[15]、算法 b[20]、算法 c[21]和算法 d[22]进行测试，运行时间如表 3 所列。需要指出的是算法 d[22]存在问题，即在关键的求核算法中缺少跳出递归的条件，本文为该算法补充了条件，但不一定能保证修改后的算法符合原算法思想。

表 3 各算法求核运行时间总表

序号	决策集	实体数	属性数(|CUD|)	核数	算法耗时(ms)					
					算法 a[15]	算法 b[20]	算法 c[21]	算法 d[22]	本文算法	最快算法
1	arrhythmia	452	280	0	10639	1497	1388	167	218	算法 d
2	breast1	569	31	0	141	31	16	102	16	本文算法，算法 c
3	breast2	198	34	0	47	16	31	45	0	本文算法
4	chess1	28056	7	6	639	94	125	109	49	本文算法
5	connect	67557	43	0	106175	25148	25288	6897	1640	本文算法
6	credit	690	16	1	65	0	18	76	0	本文算法，算法 b
7	dermatology	366	35	0	110	31	16	102	0	本文算法
8	fac	2000	217	0	42401	5943	5101	450	406	本文算法
9	letter	20000	17	3	3011	561	562	264	125	本文算法
10	mushroom	8124	23	0	1841	375	374	434	78	本文算法
11	nursery	12960	9	8	406	78	78	78	16	本文算法
12	pendigit	7494	17	0	1077	219	218	190	31	本文算法
13	poker	25010	11	5	1544	296	265	233	94	本文算法
14	satimage	4435	37	0	2621	421	374	221	62	本文算法
15	shuttle	43500	10	1	2621	562	531	282	156	本文算法
16	ticmld	5822	86	9	19625	3620	3682	1288	281	本文算法
17	waveform1	5000	22	0	1092	250	328	176	46	本文算法
18	waveform2	5000	41	0	4088	983	951	214	78	本文算法
19	wine	178	14	0	15	15	16	19	0	本文算法
20	kddcup	494021	42	2	975398	192709	192834	8764	9532	算法 d
21	covtype	581012	55	0	543389	514603	534435	15595	29037	算法 d

2. 使用超高维数据集的实验

本实验测试各算法处理超高维数据的性能，采用数据集 advertise，属性个数为 1559，实体个数为 3279，核数为 30，对各算法进行多次测试取平均值，结果如表 4 所列。

表 4 超高维数据集上各算法求核运行时间

决策集	实体数	属性数(|CUD|)	核数	算法耗时(s)					
				算法 a[15]	算法 b[20]	算法 c[21]	算法 d[22]	本文算法	最快算法
advertise	3279	1559	30	3728.075	851.208333	557.542667	35.145333	27.773667	本文算法

由表 3 和表 4 可看出：

(1)本文算法和算法 d 均能适应多属性情况，在属性个数超过 200 的情况下，仍保持良好性能；算法 a、算法 b 和算法 c 适用于少属性情况，在超高维属性决策集上耗时都明显多于

其他决策集，受属性个数的影响较大。

(2)本文算法和算法 d 均能适应多实体情况，并优于算法 a、算法 b 和算法 c，在若干情况下，本文算法的耗时接近于 0。而在大多数情况下，本文算法优于算法 d。

(3)本文算法在大部分情况下优于上述 4 种算法，无论针对多/少实体还是多/少属性的决策表，均能表现出良好的性能。

3. 使用海量数据集的实验

本实验测试各算法处理海量数据的性能。使用 kddcupbig 数据集，从其中抽取 50 万、100 万、150 万、200 万数据分别进行测试，结果如表 5 和图 2 所示。

表 5　海量数据上各算法求核运行时间

决策集	实体数	属性数(|CUD|)	核数	算法耗时(s)					
				算法 a[15]	算法 b[20]	算法 c[21]	算法 d[22]	本文算法	最快算法
kddcupbig	500000	42	1	1295.778	238.246	243.455	33.995	15.899	本文算法
kddcupbig	1000000	42	4	3101.281	512.312	509.556	85.876	38.741	本文算法
kddcupbig	1500000	42	4	4651.608	788.283	815.352	137.708	76.217	本文算法
kddcupbig	2000000	42	4	7177.138	1095.984	溢出	201.483	124.616	本文算法

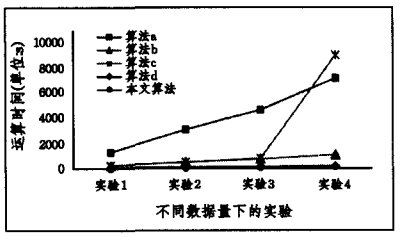

图 2　海量数据集下各算法求核时间对比

由表 5 可看出：

(1)除算法 c 外，其余算法在处理海量数据集时基本上均呈线形趋势。其中算法 a 耗时最多。

(2)算法 b 和 c 比算法 a 更适用于海量数据集，但不能因此判断算法 c 在处理海量数据方面优于算法 a，因为算法 c 在 200 万数据量时溢出。

(3)算法 d 和本文算法适用于大型数据集，但在绝大多数情况下，尤其是在决策表有核情况下，本文算法优于算法 d。

根据上述 3 类实验结果的分析，对各算法的适用情景作出补充，如表 6 所列。

表 6　各求核算法适用情景分析表-补充

算法	适用情况
本文算法	多/少实体、多/少属性、有核、无核均可适应
算法 a	对多属性、大数据集效果不理想
算法 b	对多属性、大数据集效果不理想
算法 c	对多属性、大数据集效果不理想
算法 d	适用于多/少实体、多/少属性，无核情况

结束语　现有基于正区域的约简算法的时间复杂度和空间复杂度不断下降，但仍有提高空间，尤其是在求等价类、正区域和求核算法上。基于此，本文使用改进的 Hash 和位运算获得快速等价类和正区域算法，并将其作为求核的基础。在全局正区域基础上，深入研究其与核属性间的关系。区别于现有求核方法，本文求核算法不需要多次计算完整的正区域，通过深入分析核属性 a_i 的特征，捕捉两类 $C-\{a_i\}$ 所形成的正区域与全局正区域的不一致：(1)同属于正区域的 x_i 和 x_j 无法辨识(决策值 $D(x_i)\neq D(y_i)$)；(2)属于正区域的 x_i 与不属于正区域的 x_j 无法辨识，一旦发现不一致的出现，即可停止搜索，因此不需要反复求完整的 $C-\{a_i\}$ 正区域。通过 3 个定理证明通过不一致性识别核属性的正确性，求核的平均计算时间为 $O(|U||C|^2)-O(\frac{|C||U||Core(A)|}{2})$；通过 UCI 中 21 个数据集、超高维数据集、海量数据集进行全面检验，表明无论是多/少实体、多/少属性和有/无核下的决策表，本算法在大部分情况下均优于现有同类算法，尤其适用于大型决策表。

参 考 文 献

[1] Pawlak Z. Rough sets[J]. International Journal of Computer & Information Sciences, 1982, 11(5): 341-356

[2] 王珏，王任，苗夺谦，等. 基于RoughSet理论的"数据浓缩"[J]. 计算机学报，1998，21(5)：393-400
Wang Jue, Wang Ren, Miao Duo-qian, et al. Data enriching based on rough set theory[J]. Chinese Journal of Computers, 1998, 21(5): 393-400

[3] Wang C, He Q, Chen D, et al. A novel method for attribute reduction of covering decision systems[J]. Information Sciences. 2014, 254: 181-196

[4] Yang T, Li Q, Zhou B. Related family: A new method for attribute reduction of covering information systems[J]. Information Sciences, 2013, 228: 175-191

[5] Zhang Z, Tian J. On Attribute Reduction with Intuitionistic Fuzzy Rough Sets [J]. International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, 2012, 20(1): 59-76

[6] Chen D, Hu Q, Yang Y. Parameterized attribute reduction with Gaussian kernel based fuzzy rough sets[J]. Information Sciences, 2011, 181(23): 5169-5179

[7] Bedi P, Chawla S. Use of fuzzy rough set attribute reduction in high scent web page recommendations[M]// Rough Sets, Fuzzy Sets, Data Mining and Granular Computing. Springer, 2009: 192-200

[8] Tsang E C, Chen D, Yeung D S, et al. Attributes reduction using fuzzy rough sets [J]. IEEE Transactions on Fuzzy Systems, 2008, 16(5): 1130-1141

[9] Meng Z, Shi Z. Extended rough set-based attribute reduction in inconsistent incomplete decision systems[J]. Information Sciences, 2012, 204: 44-69

[10] Qian Y, Liang J, Pedrycz W, et al. Positive approximation: An accelerator for attribute reduction in rough set theory[J]. Artificial Intelligence, 2010, 174(9): 597-618

[11] 葛浩，李龙澍，杨传健. 基于冲突的增量式核属性更新算法[J]. 控制与决策，2011，26(7)：984-990
Ge Hao, Li Long-shu, Yang Chuan-jian. Incremental updating algorithm of the computation of core based on the collision[J]. Control and Decision, 2011, 26(7): 984-990

[12] 冯少荣，张东站. 一种高效的增量式属性约简算法[J]. 控制与决策，2011，26(4)：495-500
Feng Shao-rong, Zhang Dong-zhan. Effective increment algo-

rithm for attribute reduction[J]. Control and Decision, 2011, 26(4): 495-500

[13] 杨传健,葛浩,李龙澍. 垂直划分二进制可分辨矩阵的属性约简[J]. 控制与决策,2013,28(4):563-568
Yang Chuan-jian, Ge Hao, Li Long-shu. Attribute reduction of vertically partitioned binary discernibility matrix[J]. Control and Decision, 2013, 28(4): 563-568

[14] 杨明. 一种基于改进差别矩阵的属性约简增量式更新算法[J]. 计算机学报,2007,30(5):815-822
Yang Ming. An incremental updating algorithm for attribute reduction based on improved discernibility matrix[J]. Chinese Journal of Computers, 2007, 30(5): 815-822

[15] 刘少辉,盛秋戬,吴斌,等. Rough 集高效算法的研究[J]. 计算机学报,2003,26(5):524-529
Liu Shao-hui, Sheng Qiu-jian, Wu Bin, et al. Research on efficient algorithms for rough set methods[J]. Chinese Journal of Computers, 2003, 26(5): 524-529

[16] 刘少辉,盛秋戬,史忠植. 一种新的快速计算正区域的方法[J]. 计算机研究与发展,2003,40(5):637-642
Liu Shao-hui, Sheng Qiu-jian, Shi Zhong-zhi. A new method for fast computing positive region[J]. Journal of Computer Research and Development, 2003, 40(5): 637-642

[17] 徐章艳,刘作鹏,杨炳儒,等. 一个复杂度为 max(O(|C||U|), O(|C^2|U/C|)) 的快速属性约简算法[J]. 计算机学报,2006,29(3):391-399
Xu Zhang-yan, Liu Zuo-ping, Yang Bing-ru, et al. Quick attribute reduction algorithm with complexity of max(O(|C||U|), O(|C|^2)|U/C|))[J]. Chinese Journal of Computers, 2006, 29(3): 391-399

[18] 刘勇,熊蓉,褚健. Hash 快速属性约简算法[J]. 计算机学报,2009(8):1493-1499
Liu Yong, Xiong Rong, Chu Jian. Quick attribute reduction algorithm with hash[J]. Chinese Journal of Computers, 2009, 32(8): 1493-1499

[19] 葛浩,李龙澍,杨传健. 基于冲突域的高效属性约简算法[J]. 计算机学报,2012,35(2):342-350
Ge Hao, Li Long-shu, Yang Chuan-jian. An efficient attribute reduction algorithm based on conflict region[J]. Chinese Journal of Computers, 2012, 35(2): 342-350

[20] 葛浩,李龙澍,杨传健. 一种核属性快速求解算法[J]. 控制与决策,2009,24(5):738-742
Ge Hao, Li Long-shu, Yang Chuan-jian. Quick algorithm for computing core attribute[J]. Control and Decision, 2009, 24(5): 738-742

[21] 葛浩,杨传健,李龙澍. 一种高效的核属性求解算法[J]. 计算机工程与应用,2010,46(26):138-141
Ge Hao, Li Long-shu, Yang Chuan-jian. Efficient algorithm for computing core attributes[J]. Computer Engineering and Applications, 2010, 46(26): 138-141

[22] Hu F, Wang G, Xia Y. Attribute core computation based on divide and conquer method[M]// Rough Sets and Intelligent Systems Paradigms. Springer, 2007: 310-319

(上接第 230 页)

[9] Morse M, Patel J, Grosky W. Efficient continuous skyline computation [J]. Information Sciences, 2007, 177(17): 3411-3437

[10] Zhang Z, Cheng R, Papadias D, et al. Minimizing the communication cost for continuous skyline maintenance [C]// Proceedings of the ACM International Conference on Management of Data (SIGMOD). 2009: 495-508

[11] Xin J, Wang G, Chen L, et al. Continuously maintaining sliding window skylines in a sensor network [C]// Proceedings of the International Conference on Database Systems for Advanced Applications(DASFAA). 2007: 509-521

[12] Kontaki M, Papadopoulos A N, Manolopoulos Y. Continuous Top-k Dominating Queries [J]. IEEE Transactions on Knowledge and Data Engineering(TKDE), 2012, 24(5): 840-853

[13] 孙圣力,戴东波,黄震华,等. 概率数据流上 Skyline 查询处理算法 [J]. 电子学报,2009,37(2):285-293
Sun Sheng-li, Dai Dong-bo, Huang Zhen-hua, et al. Algorithm on computing Skyline over probabilistic data stream [J]. Acta Electronica Sinica, 2009, 37(2): 285-293

[14] Zhang W, Lin X, Zhang Y, et al. Probabilistic skyline operator over sliding windows [C]// Proceedings of the IEEE International Conference on Data Engineering (ICDE). 2009: 1060-1071

[15] Lian X, Chen L. Monochromatic and bichromatic reverse skyline search over uncertain data [C]// Proceedings of the ACM International Conference on Management of Data(SIGMOD). 2008: 213-226

[16] Ding X, Lian X, Chen L, et al. Continuous monitoring of skylines over uncertain data streams [J]. Information Sciences, 2012, 184(1): 196-214

[17] Wang Y, Li X, Li X, et al. A survey of queries over uncertain data [J]. Knowledge and Information Systems(KAIS), 2013, 37(3): 485-530

[18] Li X, Wang Y, Li X, et al. Parallel skyline queries over uncertain data streams in cloud computing environments [J]. International Journal of Web and Grid Services(IJWGS), 2014, 10(1): 24-53

[19] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters [C]// Proceedings of the USENIX Symposium on Operating System Design and Implementation (OSDI). 2004: 137-150

[20] MacCormick J, Murphy N, Najork M, et al. Boxwood: Abstractions as the Foundation for Storage Infrastrucure [C]// Proceedings of the USENIX Symposium on Operating System Design and Implementation(OSDI). 2004: 105-120

[21] Wu P, Zhang C, Feng Y, et al. Parallelizing skyline queries for scalable distribution [C]// Advances in Database Technology (EDBT): Proceedings of the International Conference on Extending Database Technology. Springer, 2006: 112-130

[22] Wang S, Ooi B, Tung A, et al. Efficient skyline query processing on peer-to-peer networks [C]// Proceedings of the IEEE International Conference on Data Engineering (ICDE). 2007: 1126-1135

[23] 王媛,王意洁,邓瑞鹏,等. 云计算环境下的容错并行 Skyline 查询算法研究 [J]. 计算机科学与探索,2011,5(9):804-814
Wang Yuan, Wang Yi-jie, Deng Rui-peng, et al. Fault-tolerant parallel skyline computation in cloud computing environment [J]. Journal of Frontiers of Computer Science and Technology, 2011, 5(9): 804-814