文章编号: 1001-0920 (2016) 11-1921-15 DOI: 10.13195/j.kzyjc.2015.1164

粗等价类双边剪枝策略下多次 Hash 的约简算法

赵 洁, 张恺航, 董振宁

(广东工业大学 管理学院，广州 510520)

摘 要: 提出一种新的约简算法. 首先以全局等价类为最小计算粒度, 提出粗等价类概念, 深入研究其性质并证明粗等价类下求核和约简与原决策系统等价; 剖析 3 类粗等价类与正区域间的内在关联, 设计针对 1 和 −1 两类粗等价类双边删减下正区域的渐增式等价计算方法, 从而设计双向剪枝策略以及多次 Hash 的属性增量划分算法, 基于此给出高效完备的约简算法. 最后用 UCI 中 20 个决策集、海量、超高维 3 类数据集从多个角度进行验证, 结果表明, 所提出的约简算法的完备性和高效性在绝大多数情况下优于现有算法, 尤其适用于海量数据和超高维数据集.

关键词: 粗糙约简；粗等价类；Hash；双边剪枝

中图分类号: TP311 **文献标志码:** A

Rough equivalence class based attribute reduction algorithm with bilateral-pruning strategies and multiple Hashing

ZHAO Jie, *ZHANG Kai-hang*, *DONG Zhen-ning*

(School of Management，Guangdong University of Technology，Guangzhou 510520，China. Correspondent：ZHAO Jie，E-mail：zhaojie@gdut.edu.cn)

Abstract: A new attribute reduction algorithm is proposed. Firstly, the rough equivalence class(REC) is proposed based on the smallest computational granularity of global equivalences, and the character of REC is analyzed, under which core and reduction computation are proved to be the same with those in the original decision system. Then the relationship between positive region and the 3 types of RECs are studied, and an incremental equal method of positive region based on bilateral deleting of 1-REC and −1-REC is designed. Two directional pruning strategies and the incremental attribute partitioning algorithm with multiple Hashing are designed, based on which the efficient and complete attribution reduction algorithm is proposed. Finally, 20 decision sets of UCI, massive and ultra-high dimension data sets are used to verify the algorithms, and the results show that the attribution reduction algorithm proposed is efficient and superior to current algorithms in most conditions, and is fit for massive and ultra-high dimensional decision tables especially.

Keywords: attribute reduction using rough set；rough equivalence class；Hash；bilateral-pruning

0 引 言

粗糙集理论[1-2]是处理不精确、不完整数据的有效工具, 应用领域广泛. 其中, 高效属性约简算法是该理论的核心研究问题之一. 求最小约简是典型 NP-Hard 问题, 一般与优化方法结合求解, 但计算成本很高. 对于一般应用而言, 求某一个约简即可满足需求, 因此, 启发式约简算法一直是研究热点. 其中, 基于正区域的约简方法其时间和空间复杂度均较低, 因而备受关注. 近年来, 粗糙约简呈多元化发展, 对粗糙集扩展[3-5], 融入包括模糊集[6-8]、证据理论[9-10]在内的多种不确定性理论等. 约简算法研究日趋精细和深化, 例如在动态数据环境下约简的研究[11-12]、最大增益属性的挑选[13]等.

约简算法的一般思路如下:

1) 设计等价类算法;

2) 基于等价类算法设计正区域算法;

3) 基于前两个算法求核;

4) 以核为起点, 设计最大增益属性算法;

5) 约简检验;

6) 基于前述算法得到完备的属性约简算法.

众多学者围绕上述算法不断优化, 所提出的改进方法总结如下:

收稿日期: 2015-09-18；**修回日期:** 2015-12-17.

基金项目: 国家自然科学基金项目(71401045)；教育部人文社会科学基金项目(12YJCZH129).

作者简介: 赵洁 (1979−), 女, 副教授, 博士, 从事数据挖掘、商务智能等研究；张恺航 (1986−), 男, 硕士生, 从事信息安全、数据挖掘的研究.

1) 等价类算法改进. 由上述算法思路可见, 等价类是粗糙约简中的重要概念, 求等价类算法是约简最为基础的算法. 各种排序算法被用于改进等价类算法: 基于快速排序的等价类算法时间和空间复杂度为$O(|C||U|\log|U|)$和$O(|U|+1)$[14-15]; 基于基数排序方法的时间和空间复杂度为$O(|C||U|)$和$O(|U|)$[16-18], 但在计算中需要多次遍历U; 基于Hash方法[19]的时间复杂度同为$O(|C||U|)$, 但仅需遍历2次U.

2) 正区域算法改进. 求核、最大属性增益和约简算法均需要多次计算正区域, 正区域算法的效率对约简算法的高效性起关键性作用. 早期正区域算法计算成本很高[20], 通过简化正区域计算[14-15], 可使计算域从$|U|^2$降至$|U|$. 另外, 计算过程中可通过删减不影响结果的实体来提高效率[16, 21]. 上述两种方法自从提出以来, 一直被沿用至今. 后期多位学者均证明并使用非正区域替代正区域计算的方法, 可使属性重要度计算[19]以及缩减计算域操作更为便捷[17-18,22].

3) 求核算法优化. 等价类算法和正区域算法时间复杂度的降低可直接提高求核算法的效率, 多种改进方法可以减少求核实际运行时间, 但求核时间复杂度始终维持在$O(|C|^2|U|)$上, 是约简中最为复杂的计算步骤之一. 使用全局等价类不一致性检测的停止搜索策略, 一旦搜索到引至正区域不一致的实体, 立即停止计算[23-24], 使求核时间复杂度降至$O(|U||C|^2)-O(|C||U||\mathrm{Core}(A)|/2)$.

4) 计算域的横向压缩. 求正区域过程中删减无用实体可使计算量下降, 但不能直接降低时间复杂度. 计算域为$|U|$, 以全局等价类取代实体作为基本计算粒度可使计算域从$|U|$降至$|U/C|$[16,18-19], 且在压缩决策系统上的求核和约简与原始决策系统等价[18].

上述改进措施极大地提高了约简算法效率, 但仍有改进空间. 例如: 对于计算域的剪枝策略, 目前剪枝策略仅限于单边横向删减[16-18]; 现有研究中没有纵向删减计算域的策略, 若设计纵向删减策略, 则可进一步提高算法效率; 通常, 求核是约简的核心步骤, 但并非必要步骤[16,19], 求核对约简整体效率有何影响, 含求核的约简算法更优或是反之, 尚需较为深入的分析; 约简检验是必要步骤, 否则会使算法完备性缺失[16], 对检验算法改进也可以提高约简算法的整体效率.

基于现有研究, 针对上述问题, 本文提出一种新的约简算法. 首先以全局等价类为基本计算粒度, 提出粗等价类概念, 可使计算域从$|U|$降低为$|U/C|$. 深入分析其性质及在粗等价类下的求核与约简, 把粗等价类细分为3类, 设计1和-1-粗等价类双边删减下的正区域渐增式等价类计算方法, 从而设计双向剪枝策略, 其中横向剪枝策略是对1和-1-粗等价类的双边删减, 纵向剪枝策略是对冗余属性的删减. 设计多次Hash的属性增量划分算法, 并对约简应否包含求核步骤进行探讨. 最后给出不含求核的约简完备算法, 并通过大量实验进行验证.

1 粗等价类性质及基于粗等价类的约简

1.1 粗糙集基本概念

本节给出粗糙集基本概念, 详见文献[1].

定义 1 决策系统

$$S=(U,A,V,f). \tag{1}$$

其中: $U=\{x_1,x_2,\cdots\}$是实体的集合, 称为论域; $A=\{a_1,a_2,\cdots,a_n\}$是属性集, 可分为两个子集C和D, C称为条件属性, D称为决策属性, $A=C\bigcup D$且$C\bigcap D=\varnothing$; V是属性的值域; f是一个信息函数$f: U\times A\in V$, $f(x,a)$通常也记为$a(x)$. 假设$P=\{a_1,a_2,\cdots,a_k\}\subseteq A$, $(a_1(x),a_2(x),\cdots,a_k(x))$记为$P(x)$.

定义 2 (不可区分关系) 对于$\forall P\subseteq A$, 称

$$\mathrm{Ind}(P)=\{(x_i,x_j)|P(x_i)=P(x_j)\} \tag{2}$$

为不可区分关系, 或等价关系, 表示(x_i,x_j)关于属性集P是不可区分的. 根据$\mathrm{Ind}(P)$可以导出一个等价划分U/P, 该划分中包含对象x的等价类, 记为$[x]_P$.

定义 3 (正区域) 决策系统$S=(U,A,V,f)$, 令$P,Q\subseteq C$, 称

$$\mathrm{POS}_P(Q)=\{x|[x]_P\subseteq[x]_Q\} \tag{3}$$

为P相对于Q的正区域, $\mathrm{POS}_C(D)$称为全局正区域.

定理 1

$$\mathrm{POS}_P=\bigcup_{Y\in U/P\text{ and }|Y/Q|=1}Y. \tag{4}$$

证明 1) 先证$\forall x\in\mathrm{POS}_P(Q)$, 均有

$$x\in\bigcup_{Y\in U/P\text{ and }|Y/Q|=1}Y.$$

假设$U/P=\{P_1,P_2,\cdots,P_n\}$, $0<n\leqslant|U|$, $U/Q=\{Q_1,Q_2,\cdots,Q_n\}$, $0<m\leqslant|U|$. $\forall x\in\mathrm{POS}_P(Q)$, 有$P_i$, 使$x\in[x_i]_P$. 根据定义3, 必有$Q_j$, $[x_i]_P\subseteq[x_i]_Q$, 故$x\in[x_j]_Q$, 即$\forall x,y\in[x_i]_P$, 均有$Q(x)=Q(y)$. 因此$P_i/Q=P_i$, 即$|P_i/Q|=1$, 则

$$P_i\in\bigcup_{Y\in U/P\text{ and }|Y/Q|=1}Y,\ x\in\bigcup_{Y\in U/P\text{ and }|Y/Q|=1}Y.$$

2) 再证$\forall x\in\bigcup_{Y\in U/P\text{ and }|Y/Q|=1}Y$, 有$x\in\mathrm{POS}_P(Q)$. $\forall x\in\bigcup_{Y\in U/P\text{ and }|Y/Q|=1}Y$, 存在$P_i\in U/P$, 使$x\in[x_i]_P$. 已知$|P_i/Q|=1$, 则$P_i/Q=P_i$, 即$\forall x,y\in[x_i]_P$, $Q(x)=Q(y)$成立, 必有Q_j, 使$x\in[x_j]_Q$, 即$[x_i]_P\subseteq[x_j]_Q$. 由定义3知, $x\in\mathrm{POS}_P(Q)$. □

定理1是求正区域的等价算法, 对于$\forall x, y\in[x]_P$, 若$Q(x)=Q(y)$, 则$[x]_P\subseteq \mathrm{POS}_P(Q)$.

定义 4 (约简)　决策系统$S=(U,A,V,f)$中, 若$\forall R\subseteq C, \mathrm{POS}_R(D)=\mathrm{POS}_C(D)$, 且对于$\forall R'\subset R$, 均有$\mathrm{POS}'_R(D)\neq \mathrm{POS}_C(D)$, 则称$R$是$C$相对于$D$的属性约简, 记为

$$R=\mathrm{Red}(C). \tag{5}$$

定义 5 (核)　在决策系统$S=(U,A,V,f)$中, C中所有不可省略属性的集合称为C的核, 即

$$\mathrm{core}(C)=\bigcap \mathrm{Red}(C). \tag{6}$$

1.2　粗等价类概念及性质

定义 6 (全局等价类及其特征)　基于定义2, $\mathrm{Ind}(C)$可导出一个等价划分$U/C=\{e_1,e_2,\cdots,e_k\}$, 其中$e$是基于条件属性$C$的等价类, 称为全局等价类. 对于$e$, 定义其3个特征:

1) e.count. 等价类e中实体x的个数被记作e.count.

2) cons和dec. 对于$\forall x,y\in e$, 显然$C(x)=C(y)$. 若同时$D(x)=D(y)$, 则e.cons = true, e.dec $=D(x)$; 否则, 若存在$x,y\in e$, 但$D(x)\neq D(y)$, 则e.cons = false, e.dec $=\,'/'$, 表示e的D值不存在或无意义.

定义 7 (基于S全局等价类决策系统)　在$S^G=(U^G,A,V,f^G)$中, $U^G/C=\{e_1,e_2,\cdots,e_k\}$. f^G是$U^G\times(C\bigcup D)$到V的映射, 设$x\in e\in U^G$, 其中$C(e)=C(x)$, $D(e)=e$.dec. 称S^G是基于S全局等价类的决策系统.

同时, 对于任意的属性集$P\subseteq C$, 若$x\in e$, 则定义$P(e)=P(x)$.

定义 8 (粗等价类)　给定属性集$P\subseteq C, e\in U^G$, 则$E^P=[e]_P=\{e'|P(e')=P(e)\}$被称为$P$的粗等价类RIND. $U^G/P=\{E_1^P,E_2^P,\cdots,E_n^P\}$, E^P称为P的粗等价类.

定义8表明, 决策系统S^G中, 属性集$P\subseteq C$, U^C中的元素$\{e_1,e_2,\cdots,e_k\}$可以被合并为几个大的集合$\{E_1^P,E_2^P,\cdots,E_n^P\}$, e是S^G中粒度最小的单位. 若$P\subseteq C$, 则根据e.cons, P的粗等价类可分为3类, 同时定义粗等价类的2个特征cons和dec:

1) 若$\forall e,e'\in E^P$, 均有e.cons = e'.cons = true, 且e.dec = e'.dec, 记E^P.cons = 1, E^P.dec = e.dec = e'.dec, 则称E^P为P的1-粗等价类.

2) 若$\forall e\in E^P$, 均有e.cons = false, 记E^P.cons $=-1$, E^P.dec $=\,'/'$, 表示$D(E^P)$不存在或无意义, 则称E^P为P的-1-粗等价类.

3) 如果E^P不满足上述两种情况, 则称E^P为P的0-粗等价类.

定义 9 (S^G中D的C-正区域)　在$S^G=(U^G, C\bigcup D, V, f^G)$中, D的C-正区域为

$$\mathrm{POS}_C^G(D)=\{e|[e]_C\subseteq[e]_D\}. \tag{7}$$

定理 2　在S^G中, 令$P\subseteq C$, 对于任意$x\in e\in E_i^P$, 若e.cons = true, 则

$$x\in\mathrm{POS}_P(D)\Leftrightarrow e\in\mathrm{POS}_P^G(D). \tag{8}$$

证明　1) 先证若$x\in\mathrm{POS}_P(D)$, 则$e\in\mathrm{POS}_P^G(D)$. 任取$e'\in[e]_P$, 满足$P(e)=P(e')$, 从e'中任取y, 必有$P(y)=P(e')=P(e)=P(x)$, 即$y\in[x]_P$. 因为$x\in\mathrm{POS}_P(D)$, 由定义3知, $[x]_P\subseteq[x]_D$, 所以$y\in[x]_D$, 即$D(x)=D(y)$. 因为e.cons = true, 由定义6, 有$D(e)=D(x)=D(y)=D(e')$, 即$e'\in[e]_D$. 所以对于$\forall e'\in[e]_P$, 都有$e'\in[e]_D$, 故$[e]_P\subseteq[e]_D$, 从而由定义9知, $e\in\mathrm{POS}_P^G(D)$.

2) 再证如果$e\in\mathrm{POS}_P^G(D)$, 则$x\in\mathrm{POS}_P(D)$. 取$\forall y\in e$, 则$P(x)=P(y)$. 因为e.cons = true, 所以$D(x)=D(y)$, 即$\forall x,y\in e$, 有$D(x)=D(y)$, 由定理1知, $x\in\mathrm{POS}_P(D)$. □

定理2表明, 决策系统S中的正区域与基于S全局等价类的决策系统S^G的正区域存在映射关系. 对于任意$x\in e$, 若$x\in\mathrm{POS}_P(D)$, 则必有$e\in\mathrm{POS}_P^G(D)$. 反之亦然.

定理 3　在$S^G=(U^G,C\bigcup D,V,f^G)$中, 令$P\subseteq C$, 则

$$\mathrm{POS}_P^G(D)=\bigcup_{|E^P/D|=1}E^P\ . \tag{9}$$

证明　1) 先证$\forall e\in\bigcup_{|E^P/D|=1}E^P$, 有$e\in\mathrm{POS}_P^G(D)$. 取$\forall e,e'\in E^P\subseteq\bigcup_{|E^P/D|=1}E^P$, 同时取$\forall x\in e,\forall y\in e'$, 有$P(x)=P(e)=P(e')=P(y)$. 已知$|E^P/D|=1$, 有$D(x)=D(e)=D(e')=D(y)$, 即$\forall x,y$, 均有$P(x)=P(y)$, 且$D(x)=D(y)$. 由定理1知, $x,y\in\mathrm{POS}_P(D)$; 由定理2知, $e,e'\in\mathrm{POS}_P^G(D)$.

2) 再证$\forall e\in\mathrm{POS}_P^G(D)$, 有$e\in\bigcup_{|E^P/D|=1}E^P$. $\forall e\in\mathrm{POS}_P^G(D)$, 必存在$E^P$, 使$e\in E^P$. 由定义9知$[e]_P\subseteq[e]_D$, 取$\forall e'\in[e]_P$, 有$D(e)=D(e')$, 即$E^P/D=E^P$, 故$|E^P/D|=1$. 因此$e\in\bigcup_{|E^P/D|=1}E^P$. □

定理3与定理1作用相似, 是S^G中求正区域的等价算法. 对于$\forall e,e'\in E^P$, 若e.dec = e'.dec, 则有$E^P\in\mathrm{POS}_P^G(D)$.

定理 4 在 $S^G=(U^G,C\bigcup D,V,f^G)$ 中, 令 $P\subseteq C$, 则

$$\mathrm{POS}_P^G(D)=\bigcup_{E^P.\mathrm{cons}=1}E^P \quad . \tag{10}$$

证明 已知 $E^P.\mathrm{cons}=1$, 根据定义 8, $\forall e,e'\in E^P$, 均有 $e.\mathrm{dec}=e'.\mathrm{dec}$, 即 $D(e)=D(e')$, 所以 $|E^P/D|=1$. 即当 $E^P.\mathrm{cons}=1$ 时, 必有 $|E^P/D|=1$. 根据定理 3, 式 (10) 成立. □

定理 4 说明, 粗等价类定义简化了正区域计算, 若任意 $E^P.\mathrm{cons}=1$, 则 E^P 包含在 P 的正区域中.

推论 1

$$\mathrm{POS}_C^G(D)=\{e|e.\mathrm{cons}=\mathrm{true}\}. \tag{11}$$

由定理 4 知式 (11) 成立. 推论 1 说明, S^G 的 C 正区域由 cons 为 true 的全局等价类构成, 即 cons 为 true 的全局等价类均包含于 S^G 的全局正区域中.

1.3 粗等价类下的求核与约简

定理 5 在 $S^G=(U^G,A,V,f^G)$ 中, 令 $P,Q\subseteq A$, 则

$$\mathrm{POS}_P(D)=\mathrm{POS}_Q(D)\Leftrightarrow\mathrm{POS}_P^G(D)=\mathrm{POS}_Q^G(D). \tag{12}$$

证明 1) 先证 $\mathrm{POS}_P(D)=\mathrm{POS}_Q(D)\Rightarrow\mathrm{POS}_P^G(D)=\mathrm{POS}_Q^G(D)$. 设 $x\in e\in\mathrm{POS}_P^G(D)$, 已知 $\forall x\in\mathrm{POS}_P(D)$, $\mathrm{POS}_P(D)=\mathrm{POS}_Q(D)$, 有 $x\in\mathrm{POS}_Q(D)$, 根据定理 2, $e\in\mathrm{POS}_Q^G(D)$, 即 $\mathrm{POS}_P^G(D)\subseteq\mathrm{POS}_Q^G(D)$.

同理, $\mathrm{POS}_Q^G(D)\subseteq\mathrm{POS}_P^G(D)$, 故定理成立.

2) 再证 $\mathrm{POS}_P^G(D)=\mathrm{POS}_Q^G(D)\Rightarrow\mathrm{POS}_P(D)=\mathrm{POS}_Q(D)$. $\forall e\in\mathrm{POS}_P^G(D)$, 对于 $\forall x\in e$, 根据定理 2, 均有 $x\in\mathrm{POS}_P(D)$, 已知 $\mathrm{POS}_P^G(D)=\mathrm{POS}_Q^G(D)$, 故 $e\in\mathrm{POS}_Q^G(D)$, 因此 $x\in\mathrm{POS}_Q(D)$, 即 $\mathrm{POS}_P(D)\subseteq\mathrm{POS}_Q(D)$.

同理, $\mathrm{POS}_Q(D)\subseteq\mathrm{POS}_P(D)$, 定理成立. □

根据定理 5, 对 S 求正区域等价于对 S^G 求正区域, 给出以下定理.

定理 6 在 $S^G=(U^G,C\bigcup D,V,f^G)$ 中, $\forall R\subseteq C$, 如果 $\mathrm{POS}_R^G(D)=\mathrm{POS}_C^G(D)$, 且对于 $\forall R'\subset R$, 都有 $\mathrm{POS}_{R'}^G(D)\neq\mathrm{POS}_C^G(D)$, 则 R 是 S 中 C 相对于 D 的属性约简, 记为

$$R=\mathrm{Red}(C). \tag{13}$$

证明 由于 $\mathrm{POS}_R(D)=\mathrm{POS}_C(D)\Leftrightarrow\mathrm{POS}_R^G(D)=\mathrm{POS}_C^G(D)$, 根据定理 5, 定理得证. □

由上述定义和定理可知, 在 S^G 上求得的约简即为 S 的约简, 从而求核、约简等计算均可基于 S^G 进行, 所得结果与在 S 上所求的结果等价.

2 基于 Hash 的粗等价类算法

等价类算法是正区域、约简算法的基础, 本节首先设计基于 Hash 的等价类算法, 然后给出粗等价类算法. 本文算例使用的决策表如表 1 所示[16].

表 1 决 策 集

x	a_1	a_2	a_3	a_4	D
x_1	1	1	1	1	0
x_2	2	2	2	1	1
x_3	1	1	1	1	0
x_4	2	3	2	3	0
x_5	2	2	2	1	1
x_6	3	1	2	1	0
x_7	1	2	3	2	2
x_8	2	3	1	2	3
x_9	3	1	2	1	1
x_{10}	1	2	3	2	2
x_{11}	3	1	2	1	1
x_{12}	2	3	1	2	3
x_{13}	4	3	4	2	1
x_{14}	1	2	3	2	3
x_{15}	4	3	4	2	2

算法 1 计算属性集 P 的等价类 U/P.

输入: $S=(U,C\bigcup D,V,f)$, $P(P\subseteq C)$;

输出: $\mathrm{Ind}(P)$.

1) 为 P 初始化一个 Hash 表 H.

2) 对于 $\forall x\in U$, $\mathrm{key}=P(x)$, 执行以下步骤:

2.1) 若 H 中无当前 key, 则创建 h, $h.\mathrm{cons}=1$, $h.\mathrm{cons}=\mathrm{true}$, $h.\mathrm{dec}=D(x)$;

2.2) 若 H 中存在 key, 则

2.2.1) 获取对应 h, $h.\mathrm{cons}$++;

2.2.2) 若 $h.\mathrm{cons}=\mathrm{true}$ 且 $D(x)\neq h.\mathrm{dec}$, 则 $h.\mathrm{cons}=\mathrm{false}$.

3) 返回 H.

给定属性集 P, 使用算法 1 可得到相应等价类的 Hash. 当 $P=C$ 时, 可求得全局等价类 Hash H_C, 全局等价类是后续多种算法的基本计算单位, 只计算一次, 常驻内存. h 是 H 中的分项, 对应一个全局等价类, 3 个特征如定义 6 含义. $h.\mathrm{cons}$ 记录 h 中的实体数, $h.\mathrm{cons}$ 标识 h 中实体是否属于正区域, $h.\mathrm{dec}$ 记录 D 值. 对于每个 x, 步骤 2) 遍历 U, 时间复杂度为 $O(|U|)$. Hash 执行 put 和get 的时间复杂度均为 $O(|\mathrm{key}|)$, 故步骤 2.1) 和 2.2) 时间复杂度均为 $O(|\mathrm{key}|)$, 算法 1 时间复杂度为 $O(|P||U|)$, 与多个文献[19, 25-27]相同, 优于文献 [15] 的 $O(|A||U|\log|U|)$. 空间复杂度也与多个文献相同[15, 26-27], 优于文献 [25] 的 $O(|U|+p|C|)$. 但本文算法仅需遍历 U 两次, 基于基数排序的等价类算法需要多次遍历 U, 本文算法在实际运行效率上更优. 在表 1 上使用算法 1 计算的结果如表 2 所示.

表 2　决策表 1 的全局等价类在算法 1 下的结果

IND(C)	e.key	e.cons	e.dec	e.count	对应实体
e_1	[1111]	true	0	2	x_1, x_3
e_2	[2221]	true	1	2	x_2, x_5
e_3	[2323]	true	0	1	x_4
e_4	[3121]	false	/	3	x_6, x_9, x_{11}
e_5	[2312]	true	3	2	x_8, x_{12}
e_6	[1232]	false	/	3	x_7, x_{10}, x_{14}
e_7	[4342]	false	/	2	x_{13}, x_{15}

算法 2　计算属性集 P 的粗等价类 U^G/P.

输入: $H_C, P(P \subset C)$ // H_C 是 S 的全局等价类 Hash;

输出: U^G/P.

1) 为 P 初始化一个 Hash 表 H_P.

2) 对每一个 $e \in H_C$, key $= P(e)$, 执行以下操作:

2.1) 若 H_P 中无当前 key, 则:

2.1.1) 创建 $h, h = h \bigcup \{e\}$, h.dec $= e$.dec, h.cons++;

2.1.2) 若 e.cons $=$ true, 则 h.cons $= 1$, 否则 h.cons $= -1$.

2.2) 若 H_P 中存在对应 key, 则:

2.2.1) 获取对应 h, $h = h \bigcup \{e\}$, h.cons++;

2.2.2) 若 h.cons $= 1$ && e.cons $=$ true && h.dec $= e$.dec, 转步骤 2);

2.2.3) 若 h.cons $= -1$ && e.cons $=$ false, 则转步骤 2);

2.2.4) 否则, 令 h.cons $= 0, h$.dec $= '/'$.

3) 返回 H_P.

给定属性集合 $P(P \subset C)$, 使用算法 2 可以得到粗等价类 Hash H_P. H_P 中的子项对应一个粗等价类, 特征 cons 和 dec 如定义 8. 步骤 2) 遍历 U^G, 需要构造 P 的 key, 时间复杂度为 $O(|P||U/C|)$. 设 $P = \{a_1, a_2\}$, 对表 2 使用算法 2 得到的粗等价类如表 3 所示.

表 3　属性集 $P = \{a_1, a_2\}$ 在算法 2 下的结果

RIND	E.key	E.cons	E.dec	对应e
E_1	[11**]	1	0	e_1
E_2	[22**]	1	1	e_2
E_3	[23**]	0	/	e_3, e_5
E_4	[31**]	-1	/	e_4
E_5	[12**]	-1	/	e_6
E_6	[43**]	-1	/	e_7

3　粗等价类下约简的等价计算

3.1　粗等价类下正区域的等价计算

3.1.1　正区域的等价计算

在 S^G 中, $P \subseteq C$, 由粗等价类定义可得下式:

$$U^G/P = \bigcup_{E_i^P.\text{cons}=1} E^P \ \cup \ \bigcup_{E_j^P.\text{cons}=-1} E_j^P \ \cup \ \bigcup_{E_k^P.\text{cons}=0} E_k^P \ . \quad (14)$$

定理 7

$$U^G/C = \bigcup_{E_i^C.\text{cons}=1} E_i^C \ \cup \ \bigcup_{E_j^C.\text{cons}=-1} E_j^C \ ,$$

即

$$\Big| \bigcup_{E_k^C.\text{cons}=0} E_k^C \Big| = 0. \quad (15)$$

证明　根据定义 8, $E^C = \{e\}$, 即 $|E^C| = 1$. 因为任意 e 的 cons 值只有 true 和 false 两种, 所以 E^C 的 cons 值只有 1 和 -1, 从而 $|\bigcup_{E_k^C.\text{cons}=0} E_k^C| = 0$ 成立. □

由式 (14) 可以得到下式:

$$\bigcup_{E_k^P.\text{cons}=0} E_k^P = U^G/P - \bigcup_{E_i^P.\text{cons}=1} E_i^P - \bigcup_{E_j^P.\text{cons}=1} E_j^P \ . \quad (16)$$

由粗等价类性质和式 (16) 可知, 当属性渐增时, 新增的属性对 U^G 划分会使原有的 0-粗等价类双边递减, 从而使 1-粗等价类和 -1-粗等价类递增. 通过 1 和 -1-粗等价类可以计算 0-粗等价类的大小. 下面讨论 $\bigcup_{E_i^P.\text{cons}=1} E_i^P$ 和 $\bigcup_{E_j^P.\text{cons}=-1} E_j^P$ 的计算.

3.1.2　属性增量划分下 1-粗等价类的增量计算

令 $P \subset Q \subseteq C$, 下面分析 P 和 Q 的正区域关系.

定理 8　在 S^G 中, 令 $P \subset Q \subseteq C$, $E^P \in U^G/P$, 若 $E^P \subseteq \text{POS}_P^G(D)$, 则有 $E^P \subseteq \text{POS}_Q^G(D)$.

证明　已知 $E^P \subseteq \text{POS}_P^G(D)$, 先要证明 $\forall e \in E^P$, 均有 $e \in \text{POS}_Q^G(D)$.

已知 $P \subset Q$, 对于 $\forall E^P \subseteq \text{POS}_P^G(D)$, 必有一个或多个 $E^Q \in U^G/Q$, 使 $E^P = \bigcup_{|E^Q/P|=1} E^Q$, 即 $E^Q \subseteq E^P$. 因此 $\forall e \in E^P$, 必有 $e \in E^Q$.

在同一 E^Q 中任取 e', 有 $Q(e) = Q(e')$. 因为 $E^Q \subseteq E^P$, 所以必有 $e, e' \in E^P$. 已知 $E^P \subseteq \text{POS}_P^G(D)$, 根据定义 8 有 e.dec $= e'$.dec, 再根据定理 3, $E^Q \in \text{POS}_Q^G(D)$. 即 $\forall e \in E^P$, 均有 $e \in \text{POS}_Q^G(D)$, 因此 $E^P \subseteq \text{POS}_Q^G(D)$. □

基于上述分析, 当 $P \subset Q \subseteq C$ 时, P 的正区域即为 Q 的正区域, Q 比 P 多出的 $Q - P$ 属性通过对 P 的 0-粗等价类划分, 得到新的 1 和 -1-粗等价类, 从而使 Q 的正区域大于 P 正区域. 因此, 求 Q 正区域可在 P 对 U^G 划分基础上进行, 求 Q-P 对 P 的 0-粗等价类的增量划分即可, 从而避免重复计算, 方法如下.

定理 9　在 S^G 中, 令 $P \subset Q \subseteq C$, 则

$$\text{POS}_Q^G(D) = \text{POS}_P^G \bigcup I^{Q-P}, \quad (17)$$

其中

$$I^{Q-P}=\bigcup_{E^{Q-P}\subseteq E^P \text{ and } E^{Q-P}.\text{cons}=1 \text{ and } E^P.\text{cons}=0} E^{Q-P}.$$

证明 1) 先证$\forall e\in \text{POS}_Q^G(D)$, 均有$e\in \text{POS}_P^G(D)\bigcup I^{Q-P}$.

已知$P\subset Q$, 必有

$$\text{POS}_P^G(D)\subseteq \text{POS}_Q^G(D). \tag{18}$$

分两种情况讨论:

① $e\in \text{POS}_Q^G(D)$且$e\in \text{POS}_P^G(D)$, 式(18)成立;

② $e\in \text{POS}_Q^G(D)$且$e\notin \text{POS}_P^G(D)$, 此时对于$\forall e, e'\in \text{POS}_Q^G(D)$, 由于$P\subset Q$, 有$(Q-P)e=(Q-P)e'$, 必有$\exists E^{Q-P}$, 使$e\in E^{Q-P}$. 同时取$\forall e''\in E^{Q-P}$, 必有$(Q-P)(e)=(Q-P)(e'')$. 由已知再结合定义8, 可得$e.\text{cons}=e''.\text{cons}=\text{true}$, $e.\text{dec}=e''.\text{dec}$. 再根据粗等价类分类有$E^{Q-P}.\text{cons}=1$.

同理, 因为$P\subset Q$, 必有$P(e)=P(e'')$, 所以在条件$e\in \text{POS}_Q^G(D)$下, 当$e\in E^{Q-P}$时, 均有$e\in E^P$, 因此$E^{Q-P}\subseteq E^P$.

由上面证明可知$e\in E^{Q-P}\subseteq E^P\notin \text{POS}_P^G(D)$. 假设$E^P.\text{cons}=1$, 根据定理4, 必有$E^P\subseteq \text{POS}_P^G(D)$, 与已知矛盾.

若$E^P.\text{cons}=-1$, 则必有$e.\text{cons}=\text{false}$, 与已知$e\in \text{POS}_Q^G(D)$矛盾, 因此必有$E^P.\text{cons}=0$. 从而可得$e\in I^{Q-P}$.

2) 再证$\forall e\in \text{POS}_P^G(D)\bigcup I^{Q-P}$, 均有$e\in \text{POS}_Q^G(D)$.

已知$\forall e\in \text{POS}_P^G(D)\bigcup I^{Q-P}$, 分两种情况:

① 若$e\in \text{POS}_P^G(D)$, 根据定理8, 则必有$e\in \text{POS}_Q^G(D)$.

② 若$e\in I^{Q-P}$, 由已知$\forall e, e'\in E^{Q-P}\in I^{Q-P}$, 且$E^{Q-P}\subseteq E^P$, 则$e, e'\in E^P$. 因$P(e)=P(e')$且$(Q-P)(e)=(Q-P)(e')$, 有$Q(e)=Q(e')$, 故$E^{Q-P}\subseteq U^G/Q$. 已知$E^{Q-P}.\text{cons}=1$, 由定理4, $E^{Q-P}\subseteq \text{POS}_Q^G(D)$, 因此$e\in \text{POS}_Q^G(D)$. □

根据定理8和定理9, 若$A_1\subset A_2\cdots\subset A_i\subset A_n\subseteq C$, 则$\bigcup_{E^A.\text{cons}=1}E^A$的计算可转化为以下增量式计算:

$$\begin{cases}\bigcup\limits_{E^{A_{i+1}}.\text{cons}=1}E^{A_{i+1}}=\bigcup\limits_{E^{A_i}.\text{cons}=1}E^{A_i}+I^{A_{i+1}-A_i},\ i\geqslant 1;\\ I^{A_1-A_0}=\varnothing,\ A_0=\varnothing.\end{cases} \tag{19}$$

其中

$$I^{A_{i+1}-A_i}=\bigcup_{E^{A_{i+1}-A_i}\subseteq E^{A_i} \text{ and } E^{A_{i+1}-A_i}.\text{cons}=1 \text{ and } E^{A_i}.\text{cons}=0} E^{A_{i+1}-A_i}.$$

3.1.3 属性增量划分下 −1-粗等价类的增量式计算

与1-粗等价类的增量式计算相似, −1-粗等价类增量式计算的相关证明如下. 首先给出如下定理.

定理 10 在S^G中, 令$P\subset Q\subseteq C$, 若$E^P.\text{cons}=-1$, 则$E^P\nsubseteq \text{POS}_Q^G(D)$.

证明 已知$E^P.\text{cons}=-1$, 根据定义8, $\forall e\in E^P$, 有$e.\text{cons}=\text{false}$, 对于任意$E^P$, 均存在一个或多个$E^Q$, 使$E^P=\bigcup_{|E^Q/P|=1}E^Q$. 取$\forall e, e'\in E^Q$, 均有$e.\text{cons}=e'.\text{cons}=\text{false}$, 则$D(e)=D(e')$不成立. 根据定理3, $E^Q\nsubseteq \text{POS}_Q^G(D)$. 因此$\bigcup_{|E^Q/P|=1}E^Q=E^P\nsubseteq \text{POS}_Q^G(D)$. □

定理10说明, 所有被P划分出的−1-粗等价类都不可能成为Q的正区域, 计算公式及证明如下.

定理 11 在S^G中, 令$P\subset Q\subseteq C$, 则

$$\bigcup_{E^Q.\text{cons}=-1}E^Q=\bigcup_{E^P.\text{cons}=-1}E^P\bigcup N^{Q-P}, \tag{20}$$

其中

$$N^{Q-P}=\bigcup_{E^{Q-P}\subseteq E^P \text{ and } E^{Q-P}.\text{cons}=-1 \text{ and } E^P.\text{cons}=0} E^{Q_P}.$$

证明与定理9的相似, 此略.

根据定理11, $\left|\bigcup_{E^A.\text{cons}=-1}E^A\right|$的增量式计算公式如下:

$$\begin{cases}\bigcup\limits_{E^{A_{i+1}}.\text{cons}=-1}E^{A_{i+1}}=\bigcup\limits_{E^{A_i}.\text{cons}=-1}E^{A_i}+N^{A_{i+1}-A_i},\ i\geqslant 1;\\ |N^{A_1-A_0}|=\varnothing,\ A_0=\varnothing.\end{cases} \tag{21}$$

其中

$$N^{A_{i+1}-A_i}=\bigcup_{E^{A_{i+1}-A_i}\subseteq E^{A_i} \text{ and } E^{A_{i+1}-A_i}.\text{cons}=-1 \text{ and } E^{A_i}.\text{cons}=0} E^{A_{i+1}-A_i}.$$

3.1.4 属性增量划分下 0-粗等价类的增量式计算

由式(19)和(21)可得

$$\begin{cases}\bigcup\limits_{E^{A_0}.\text{cons}=0}E^{A_0}=U^G,\ A_0=\varnothing;\\ \bigcup\limits_{E^{A_1}.\text{cons}=0}E^{A_1}=U^G-\bigcup\limits_{E^{A_1}.\text{cons}=1}E^{A_1}-\bigcup\limits_{E^{A_1}.\text{cons}=-1}E^{A_1};\\ \bigcup\limits_{E^{A_{i+1}}.\text{cons}=0}E^{A_{i+1}}=U^G-I^{A_{i+1}-A_n}-N^{A_{i+1}-A_i},\ i\geqslant 1.\end{cases} \tag{22}$$

约简中依据正区域的计算可转化为0-粗等价类的增量式计算, 最后通过判断0-粗等价类大小, 即可判断约简或冗余属性.

3.2 粗等价类下约简的等价计算

根据定理7和式(16), 在S^G中, 若$R\subseteq C$, 则可根据R对U^G划分所得的0-粗等价类大小, 判断R是否包含约简.

定理 12 在S^G中, 若

$$U^G/R=\bigcup_{E_i^R.\mathrm{cons}=1}E_i^R\ \cup\ \bigcup_{E_j^R.\mathrm{cons}=-1}E_j^R\ ,$$

即$\left|\bigcup_{E_k^R.\mathrm{cons}=0}E_k^R\right|=0$, 则 R 为包含约简的一组属性.

证明　假设 R 不包含约简. 由定理 7 可知 $\left|\bigcup_{E^C.\mathrm{cons}=0}E^C\right|=0$, 即 $\forall e.\mathrm{cons}=1$, 则必有 $e\in\mathrm{POS}_C^G(D)$.

由于假设 R 不包含约简, $\mathrm{POS}_C^G(D)\neq\mathrm{POS}_R^S(D)$, 必有 $\exists e.\mathrm{cons}=\mathrm{true}$, 且 $e\notin\mathrm{POS}_R^G$, 必存在 $E_k^R.\mathrm{cons}=0$, 所以$\left|\bigcup_{E_k^R.\mathrm{cons}=0}E_k^R\right|\neq0$, 这与已知$\left|\bigcup_{E_k^R.\mathrm{cons}=0}E_k^R\right|=0$矛盾, 因此 R 必为包含约简的一组属性. □

基于定理 12, 渐增式逐个增加属性形成属性组 R, 对 U^G 划分所得$\left|\bigcup_{E_k^R.\mathrm{cons}=0}E_k^R\right|=0$, 必有 $\mathrm{POS}_C^G(D)=\mathrm{POS}_R^S(D)$, 则 R 必为包含约简的一组属性.

3.3　粗等价类下判断冗余属性的等价计算

由定理 7 和式 (16), 可根据 0-粗等价类大小判断属性 a 是否为冗余属性, 其中 $a\in R$ (R 包含约简).

定理 13　在 S^G 中, 若 R 为包含约简的属性组, 则

$$U^G/(R-\{a\})=\bigcup_{E_i^{R-\{a\}}.\mathrm{cons}=1}E_i^{R-\{a\}}\ \cup\ \bigcup_{E_j^{R-\{a\}}.\mathrm{cons}=-1}E_j^{R-\{a\}}\ \cup\ \bigcup_{E_k^{R-\{a\}}.\mathrm{cons}=0}E_k^{R-\{a\}}\ .\quad(23)$$

若$\left|\bigcup_{E_k^{R-\{a\}}.\mathrm{cons}=0}E_k^{R-\{a\}}\right|\neq0$, 则 a 必是 R 包含约简中的不可省略属性.

证明　已知 R 是包含约简的一组属性, 且$\left|\bigcup_{E_k^{R-\{a\}}.\mathrm{cons}=0}E_k^{R-\{a\}}\right|\neq0$, 必有 $\exists E_k^{R-\{a\}}.\mathrm{cons}=0$, 且 $\exists e.\mathrm{cons}=\mathrm{true}$, $e\in E_k^{R-\{a\}}$. 根据定理 4, $e\in E_k^{R-\{a\}}\not\subseteq\mathrm{POS}_{R-\{a\}}^G$, 但根据推论 1, $e\in\mathrm{POS}_C^G(D)$, 则必有 $\mathrm{POS}_{R-\{a\}}^G\neq\mathrm{POS}_C^G(D)$, 因此, a 在 R 包含的约简中不可省略. □

定理 14　在 S^G 中, 若 R 为包含约简的属性组, 则

$$U^G/(R-\{a\})=\bigcup_{E_i^{R-\{a\}}.\mathrm{cons}=1}E_i^{R-\{a\}}\ \cup\ \bigcup_{E_j^{R-\{a\}}.\mathrm{cons}=-1}E_j^{R-\{a\}}\ \cup\ \bigcup_{E_k^{R-\{a\}}.\mathrm{cons}=0}E_k^{R-\{a\}}\ .\quad(24)$$

若$\left|\bigcup_{E_k^{R-\{a\}}.\mathrm{cons}=0}E_k^{R-\{a\}}\right|=0$, 则 a 必是 R 包含约简中的冗余属性.

定理 14 的证明过程与定理 13 的相似.

根据定理 13 和定理 14, 可得到冗余属性的等价计算方法, 即通过计算 $R-\{a\}$ 对 U^G 划分所得的 0-粗等价类的大小, 便可判断 a 是否为冗余属性. 相关引理可以应用于求核和约简检验计算中, 在求核中可识别核属性, 因为核属性一定是非冗余属性, 在约简检验中可以判断多余属性.

3.4　粗等价类下不一致信息系统的处理

定义 10 (不一致决策系统)　在决策系统 S 中, 若 $\exists x,y\in U$, $[x]_C=[y]_C$ 且 $D(x)\neq D(y)$, 则称 S 是不一致的, 否则 S 是一致的.

根据定义 6, 当 S 不一致时, 必有 $e.\mathrm{cons}=\mathrm{false}$. 由定理 8, 任意约简 $\mathrm{Red}(C)$ 或者 C 均可把所有不一致的对象完整划分到 -1-粗等价类中, 因此不需要增加额外处理. 本文基于粗等价类的算法即可用于处理不一致信息系统. 与现有约简算法研究不同, 本文基于粗等价类, 把论域细分成 3 类, 求约简过程中, 渐增式把属性逐个加入, 新增属性在原属性对论域划分得到的 0-粗等价类上进一步划分, 划分出新的 1 和 -1-粗等价类, 对其进行删减可不断缩小计算域, 不影响结果. 基于此设计 1 和 -1-粗等价类双边删减下的正区域、约简的等价计算方法, 从而设计双向剪枝策略, 其中横向剪枝策略对 1和 -1-粗等价类进行双边删减, 纵向剪枝策略在约简检验中对遍历属性进行删减.

4　粗等价类双边剪枝下多次 Hash 的约简算法

基于前述分析, 下面给出剪枝策略及相关算法.

4.1　横向和纵向剪枝策略

根据定理 8 和定理 10, 当 $P\subset Q\subseteq C$ 时, 属性集 P 对 U^G 划分形成的 1 和 -1-粗等价类必会成为 Q 的 1 和 -1-粗等价类, 因此, 对其删减不影响计算结果, 可有效缩减计算域, 基于此给出双向剪枝策略, 下面首先给出剪枝的定义.

定义 11 (剪枝)　求核、约简和冗余属性检验中, 根据定理 8、定理 10 和式 (22), 不断删减不影响计算结果的实体和属性的方法称为剪枝. 剪枝策略分两种, 其中对计算实体的删减称为横向剪枝, 对遍历属性的删减称为纵向剪枝.

1) 双边删减的横向剪枝策略. 横向剪枝策略用于对 1 和 -1-粗等价类双边删减, 可在约简与求核、约简检验中横向删减计算实体.

定义 12 (横向剪枝策略)　增量式求正区域的等价计算中, 根据定理 8 和定理 10, 删除所有 1 和 -1-粗等价类.

2) 删减属性的纵向剪枝策略. 根据定理 13 和定理 14 可得到纵向剪枝策略, 应用于求核与约简检验中, 对遍历属性进行纵向删减. 判断冗余属性的思路是: 若 R 为包含约简的一组属性, 则忽略 R 中属性 a_i,

把$a_j{\in}R-\{a_i\}$逐个并入属性集R'; 若R'对U^G划分所得0-粗等价类为$\varnothing$, 则剩余属性$\{a_j,a_{j+1},\cdots,a_{|R|}\}$及$a_i$均为冗余属性, 此时可使用纵向剪枝策略.

根据上述思路可以得到纵向剪枝策略. 根据纵向剪枝策略, 无需遍历全部属性即可判断冗余属性.

定义 13 (纵向剪枝策略) 在检验冗余属性中, 基于定理13和定理14将冗余的属性删除.

4.2 多次Hash的属性增量划分算法

基于前述定理和式(22), 给出多次Hash属性增量划分算法, 实现属性组对论域划分的增量式计算.

算法3 属性集Q相对于属性集P粗等价类的增量划分.

输入: $U^G/P=\bigcup\limits_{E^P.\mathrm{cons}=0}E^P$, 属性集$Q$($P{\subset}Q$, 若$P=\varnothing$, 则$U^G/P=U^G$)//$U^G/P$只保留0-粗等价类, 是一个Hash;

输出: U^G/Q.

1) 创建一个Hash H, $U'=U^G/P$.

2) 依次对$\forall E^P\subseteq U'$, 执行以下操作//遍历P的0-粗等价类.

2.1) $T=Q-P$, 为E^P初始化一个Hash H^{T} // 用$Q-P$对P的粗等价类进行增量划分.

2.2) 对于$\forall e\in E^P$, 执行下列操作:

2.2.1) 令key $=T(e)$, 在H^{T}中查找.

2.2.2) 若H^{T}中无对应key, 则:

2.2.2.1) 创建h^{T}, 令$h^{\mathrm{T}}.\mathrm{key}=T(e)$, $h^{\mathrm{T}}.\mathrm{dec}=e.\mathrm{dec}$;

2.2.2.2) 若$e.\mathrm{cons}=\mathrm{true}$, 则$h^{\mathrm{T}}.\mathrm{cons}=1$, 否则$h^{\mathrm{T}}.\mathrm{cons}=-1$.

2.2.3) 若H^{T}中存在对应key, 则:

2.2.3.1) 获取对应h^{T}, 若$h^{\mathrm{T}}.\mathrm{cons}=1$ && $e.\mathrm{cons}=\mathrm{true}$ && $h^{\mathrm{T}}.\mathrm{cons.dec}=e.\mathrm{dec}$, 则转步骤2.2);

2.2.3.2) 否则, 若$h^{\mathrm{T}}.\mathrm{cons}=-1$ && $e.\mathrm{cons}=\mathrm{false}$, 则转2.2);

2.2.3.3) 否则, 令$h^{\mathrm{T}}.\mathrm{cons}=0$, $h^{\mathrm{T}}.\mathrm{dec}='/'$.

2.3) 遍历H^{T}, 把H^{T}中的粗等价类放入H中.

3) 返回H.

算法中H^{T}的子项h^{T}的cons和count如定义8. 算法3是Q在P对计算域划分结果基础上的进一步划分, 其中, 基于定理8~定理11和式(22), 计算域为P对U^G划分所得的0-粗等价类, 通过横向剪枝策略删减1和-1-粗等价类, 可避免大量重复计算, 时间复杂度为$O\Big|\bigcup\limits_{E^P.\mathrm{cons}=0}E^P\Big|$, 大多数情况下$\Big|\bigcup\limits_{E^P.\mathrm{cons}=0}E^P\Big|\ll|U/C|$. 执行put和get的时间复杂度为$O(|\mathrm{key}|)$, 步骤2.2.2)和2.2.3)的时间复杂度均为$O(|Q-P|)$, 因此算法的时间复杂度为$O\Big(|Q-P|\Big|\bigcup\limits_{E^P.\mathrm{cons}=0}E^P\Big|\Big)$, 最差情况下为$O(|Q-P||U/C|)$.

4.3 计算0-粗等价类大小的算法

基于0-粗等价类的正区域等价计算中, 需要多次计算0-粗等价类大小, 下面给出算法及分析.

算法4 判断属性组P对U'划分所得0-粗等价类是否为$\varnothing$及计算冗余属性.

输入: U', $P(P\subseteq C)$ // U'仅保留0-粗等价类, 是一个Hash;

输出: true/false(对U'划分所得0-粗等价类是否为空), 冗余属性集NR.

1) 初始化NR $=\varnothing$, 对于$\forall a_i{\in}P$执行以下操作:

1.1) 调用算法3(U', a_i), 获取属性a_i对U'划分所得到的哈希表$U/\{a_i\}$, $U'=U/\{a_i\}$.

1.2) 将U'中所有1和-1-粗等价类删除//横向剪枝.

1.3) 若$U'=\varnothing$, 则:

1.3.1) NR $=$ NR$\bigcup\{a_{i+1}\cdots a_{|P|}\}$//冗余属性;

1.3.2) 返回true, NR//$|$0-粗等价类$|=0$, 结束;

1.4) 否则, 转步骤1.1).

2) 返回false & NULL //$|$0-粗等价类$|\neq0$.

算法4的输入U'为计算域, $|U'|\leqslant|U^G|$, 即$|U'|\leqslant|U/C|$. 步骤1) 把P中的属性逐个加入, 由算法3分析$|Q-P|=1$. 步骤1.2)删减1和-1-粗等价类, 因此每一次循环计算中删减的计算域为

$$n_1=\Big|\bigcup_{E_i^P.\mathrm{cons}=1}E_i^P\Big|+\Big|\bigcup_{E_j^P.\mathrm{cons}=-1}E_j^P\Big|,$$

$$n_2=|U^G|-n_1-\Big|\bigcup_{E_k^P.\mathrm{cons}=0}E_k^P\Big|,$$

其时间复杂度为$O(|U^G|-n_1-n_2\cdots-n_m)$. 当循环至属性$a_j$时, 若$U'=\varnothing$, 则算法终止, 剩余属性均为冗余属性. 步骤2)总共执行了$|P|-|\mathrm{NR}|$次, NR为P中的冗余属性, 数量不确定且与次序相关. 最理想情况下前$|P|-|\mathrm{NR}|$个属性恰好是一个约简, 最坏情况下是$|\mathrm{NR}|=0$, 此时需循环$|P|$次, 时间复杂度为$O((|P|-|\mathrm{NR}|)(|U^G|-n_1-n_2\cdots-n_{|\mathrm{Red}(C)|}))\ll O(|P||U/C|)$. 绝大多数情况下, 横向剪枝策略生效, 可双边缩减计算域; 只要P中存在冗余属性, 纵向剪枝策略可能生效, 但依赖于属性次序.

4.4 约简算法是否包含求核的探讨

经典的约简算法以求核为起点, 在核属性基础上逐个增加最大增益属性, 直到算法结束条件成立. 贪心算法则忽略求核步骤. 本节分析求核对约简算法效率的影响, 确定后续的约简算法设计是否包含求核.

现有求核的经典实现方法是: 每次忽略一个属性 $a(a \in C)$, 然后对 $C-\{a\}$ 求正区域, 通过判断 $\mathrm{POS}_C(D) \neq \mathrm{POS}_{C-\{a\}}(D)$ 识别 a 为核属性, 计算需要遍历 $|C|$ 个属性, 每次求 $|C|-1$ 个属性的正区域, 时间复杂度是 $|C|^2|U|$.

在本文算法 4 的基础上, 易得粗等价类下双向剪枝的多次 Hash 的求核算法.

算法 5　求核算法.

输入: $S^G=(U^G, C\bigcup D, V, f^G)$;

输出: $\mathrm{Core}(C)$.

1) 初始化 $\mathrm{NR}=\varnothing$, $C'=C$.

2) 对 $a \in C'$ 执行以下操作//纵向剪枝下 C' 会改变:

2.1) $U'=U^G$;

2.2) 调用算法 4 $(U', C-\{a\})$, $N=$ 算法 4 返回的冗余属性;

2.3) 若算法 4 返回 false, 则转步骤 2) //a 是核属性;

2.4) 否则, 若算法 4 返回 true, 则 $\mathrm{NR}=\mathrm{NR}\bigcup N\bigcup\{a\}$, $C'=C'-\mathrm{NR}$, 转步骤 2) //纵向剪枝.

3) 返回 $\mathrm{Core}(C)=C'$.

本文求核算法调用算法 4, 算法 4 中使用横向的双边剪枝策略, 算法 5 中使用纵向剪枝策略. 最理想的情况下, 核属性是一个约简 $\mathrm{Red}(C)=\mathrm{Core}(C)$, 此时时间复杂度为 $O(|\mathrm{Red}(C)|^2(|UG|-n_1-n_2\cdots-n_{|\mathrm{Red}(C)|}))\ll O(|\mathrm{Red}(C)^2|U/C|)<O(|C|^2|U/C|)$, 其中 n_i 与算法 4 中的分析同样, 最坏情况下为 $O(|C|^2|U/C|)$, 即信息系统无核或者 $C=\mathrm{Core}(C)$. 当信息系统无核时, 求核约简算法比不求核约简算法计算成本高, 求核后步骤的运行时间相同. 当信息系统存在核时, 求核约简算法首先求核, 以核属性为起点, 搜索最大增益属性的范围是 $C-\mathrm{Core}(C)$, 而不求核约简算法搜索范围是 C. 由于约简一定包含核属性, 以核属性为起点的启发策略是较优的, 因为不求核约简算法以空集为起点, 搜索次数可能较多. 但当数据量及属性数增多时, 求核的运算时间可能高于搜索属性的运算时间. 同时, 由于约简实际上是属性的组合优化, 以核为起点有可能妨碍其他具有较强划分能力的属性入选, 不一定能得到较优解; 而以空集为起点, 核属性一定会入选, 并且可能使属性重要度高的属性优先入选. 结合约简算法分析, 不求核约简算法的时间复杂度是 $O(|R|^2|U/C|)$, 其中 R 是包含约简的一组属性, $|R|\leqslant|C|$, 在一定程度上优于含求核算法的 $O(|C|^2|U/C|)$. 因此, 基于上述分析, 本文约简算法不含求核.

4.5　最大增益属性算法

下面给出粗等价类下属性重要度 sig 的定义, 然后给出最大增益属性算法. 当多个属性重要度相同时, 本文使用属性划分能力作为指标来选取属性.

定义 14 (属性重要度)　在 S^G 中, $R\subset C$, $a\in C-R$, 属性重要度定义为

$$\mathrm{sig}(a,R,D)=\Big|\bigcup_{E^{R\bigcup\{a\}}.\mathrm{cons}=1}E^{R\bigcup\{a\}}\Big|-\Big|\bigcup_{E^R.\mathrm{cons}=1}E^R\Big|. \quad (25)$$

算法 6　最大增益属性算法.

输入: $U^G/R=\bigcup_{E^R.\mathrm{cons}=0}E^R$, R (若 $R=\varnothing$, 则 $U^G/R=U^G$) // U^G/R 只保留 0-粗等价类, 是一个 Hash;

输出: 相对 R 的增益最大属性 $a*$.

1) 初始化 $\mathrm{maxPos}=0$, $\mathrm{maxRIND}=0$.

2) 依次对 $\forall a\in C-R$ 执行下列操作:

2.1) $U'=U^G/R$, 用算法 3 (U', a), 获得 a 对 R 增量划分的 Hash $H_{R\bigcup\{a\}}$;

2.2) 遍历 $H_{R\bigcup\{a\}}$, 计算 (x,y), 其中

$$x=\Big|\bigcup_{E_i^{R\bigcup\{a\}}.\mathrm{cons}=1}E_i^{R\bigcup\{a\}}\Big|,\ y=\Big|\bigcup_{E_j^{R\bigcup\{a\}}.\mathrm{cons}=0}E_j^{R\bigcup\{a\}}\Big|;$$

2.3) 若 $x>\mathrm{maxPos}$, 则 $a*=a$;

2.4) 否则, 若 $x=\mathrm{max\,Pos}\ \&\&\ y>\mathrm{max\,RIND}$, 则 $a*=a$.

3) 返回 $H_{R\bigcup\{a*\}}$, $a*$.

算法使用二元组指标来选择属性, 其中 x 是定义 15 中的 sig. 当多个属性的 sig 值同为最大值时, 使用 y 指标, y 是当前属性对原 0-粗等价类增量划分后得到的 0-粗等价类数量. 本文认为 0-粗等价类粒度越细 (即数量越多), 表示属性 a 划分能力相对越高. 实际计算中经常出现属性增益为 0 的情况, 即 $x=0$, 此时使用 y 选择属性. 步骤 2) 需要遍历 $|C|=|R|$ 个属性; 步骤 2.1) 调用算法 3, 时间复杂度为 $O\Big(\Big|\bigcup_{E^R.\mathrm{cons}=0}E^R\Big|\Big)$; 步骤 2.2) 遍历 $H_{R\bigcup\{a\}}$, 时间复杂度为

$$O\Big(\Big|\bigcup_{E^{R\bigcup\{a\}}.\mathrm{cons}=0}E^{R\bigcup\{a\}}\Big|\Big),$$

$$O\Big(\Big|\bigcup_{E^R.\mathrm{cons}=0}E^R\Big|\Big)\leqslant O\Big(\Big|\bigcup_{E^{R\bigcup\{a\}}.\mathrm{cons}=0}E^{R\bigcup\{a\}}\Big|\Big)\leqslant|U/C|.$$

因此, 算法 6 的时间复杂度在最差情况下为

$$O\Big(\Big|\sum_{i=1}^{|C|-|R|}\bigcup_{E^{R\bigcup\{a\}}.\mathrm{cons}=0}E^{R\bigcup\{a\}}\Big|\Big).$$

4.6　约简检验算法

首先基于算法 4, 给出约简检验算法.

算法 7　约简检验算法.

输入: U^G, 待检验的一组属性 R// U^G 是一个 Hash, R 是包含约简的一组属性;

输出: 约简 $\mathrm{Red}(C)$.

1) $R'=R$, $\mathrm{NR}=\varnothing$;

2) 从最后一个 a 开始往前, 对 $\forall a\in R'$ 执行以下操作//纵向剪枝下, R' 会被改变:

2.1) $U'=U^G$;

2.2) 调用算法 4 $(U', R'-\{a\})$, NR = 算法 4 返回的冗余属性;

2.3) 若算法 4 返回 true, 则 $R'=R'-\mathrm{NR}-\{a\}$, 转步骤 2) //纵向剪枝.

3) 返回 $\mathrm{Red}(C)=R'$.

步骤 2) 对待检验属性组进行遍历, 步骤 2.2) 多次调用算法 4. 因为步骤 2.3) 的纵向剪枝策略会改变 R' 集合, 在最佳情况下调用 $|\mathrm{Red}(C)|$ 次算法 4, 其中 $\mathrm{Red}(C)$ 是一个约简, 在最坏情况下调用 $|R|$ 次, 所以算法 7 的时间复杂度为

$$O((|R|-|\mathrm{NR}|)^2(|U^G|-n_1-n_2\cdots-n_{|\mathrm{Red}(C)|}))\ll O(|R|^2|U/C|),$$

其中 n_i 的分析与算法 4 相同, 只要 R 中包含冗余属性, 纵向剪枝策略便能生效, 使遍历属性减少.

4.7 粗等价类双向剪枝下多次 Hash 的完备约简算法

基于上述分析, 下面给出完整的约简算法.

4.7.1 约简算法

首先根据算法 1 得到 $S^G=(U^G, C\bigcup D, V, f^G)$ 作为算法 8 的输入, 计算粗等价类时调用算法 2.

算法 8 完备约简算法.

输入: $S^G=(U^G, C\bigcup D, V, f^G)$;

输出: $\mathrm{Red}(C)$.

1) 初始化 $R=\varnothing$, $U'=U^G$.

2) 当 $U'\neq\varnothing$, 反复执行下面步骤:

2.1) $\forall a\in C-R$, 调用算法 6 (U', R), 获得最大增益属性 $a*$, $R=R\bigcup\{a*\}$;

2.2) 对算法 6 返回的 $H_{R\bigcup\{a*\}}$ 删除其中的所有 1 和 -1-粗等价类, $U'=H_{R\bigcup\{a*\}}$.

3) 对 R 调用算法 7, 得到检验后的约简 $\mathrm{Red}(C)$.

4) 返回 $\mathrm{Red}(C)$.

步骤 2.1) 调用 $|R|$ 次算法 6, 其中 $|R|$ 是包含约简的一组属性, 每次删减 1 和 -1-粗等价类, 第 1 次算法 6 需要遍历 $|C|$ 个属性, 第 2 次需要遍历 $|C|-1$ 个, 第 $|R|$ 次需要遍历 $|C|-|R|$ 个, 遍历的计算域依次递减, 分析与算法 4 相似. 因此, 算法 8 的时间复杂度为

$$O(|R|^2(|U^G|-n_1-n_2\cdots-n_{|\mathrm{Red}(C)|})).$$

步骤 3) 的时间复杂度为 $O(|R|^2|U/C|)$, 在极端情况下 $|R|=|C|$, 大多数情况下 $|R|<|C|$.

4.7.2 约简数值算例

根据表 2, 调用算法 6 和算法 8 求约简.

1) 初始化 $R=\varnothing$, 求最大增益属性. 使用算法 6 计算每个 a 的重要度和划分能力, 求最大增益属性 $a*$, $R=R\bigcup\{a*\}$.

2) 第 1 轮计算, 令 $U'=U^G$.

2.1) 调用算法 3 (U', a_1), 得到粗等价类 $\bigcup\limits_{E^R.\mathrm{cons}=0}E^R=\{\{e_1,e_6\},\{e_2,e_3,e_5\}\}$, $\bigcup\limits_{E^R.\mathrm{cons}=-1}E^R=\{\{e_4\},\{e_7\}\}$, $\mathrm{sig}(a_1)=0$, $y=2$, $\Big|\bigcup\limits_{E^R.\mathrm{cons}=0}E^R\Big|\neq\varnothing$.

2.2) 调用算法 3 (U', a_2), 得到粗等价类 $\bigcup\limits_{E^R.\mathrm{cons}=0}E^R=\{\{e_1,e_4\},\{e_2,e_6\},\{e_3,e_5,e_7\}\}$, $\mathrm{sig}(a_2)=0$, $y=3$, $\Big|\bigcup\limits_{E^R.\mathrm{cons}=0}E^R\Big|\neq\varnothing$.

2.3) 使用算法 3 (U', a_3), 得到粗等价类 $\bigcup\limits_{E^R.\mathrm{cons}=0}E^R=\{\{e_1,e_5\},\{e_2,e_3,e_4\}\}$, $\bigcup\limits_{E^R.\mathrm{cons}=-1}E^R=\{\{e_6\},\{e7\}\}$, $\mathrm{sig}(a_3)=0$, $y=2$, $\Big|\bigcup\limits_{E^R.\mathrm{cons}=0}E^R\Big|\neq\varnothing$.

2.4) 使用算法 3 (U', a_4), 得到粗等价类 $\bigcup\limits_{E^R.\mathrm{cons}=1}E^R=\{\{e_3\}\}$, $\bigcup\limits_{E^R.\mathrm{cons}=0}E^R=\{\{e_1,e_2,e_4\},\{e_5,e_6,e_7\}\}$, 1-粗等价类符合剪枝策略, $\mathrm{sig}(a_4)=1$, $y=2$, $\Big|\bigcup\limits_{E^R.\mathrm{cons}=0}E^R\Big|\neq\varnothing$.

第 1 轮计算结束, $a*=a_4$, 此时 $\Big|\bigcup\limits_{E^R.\mathrm{cons}=0}E^R\Big|\neq\varnothing$, $R=a_4$.

3) 第 2 轮计算, 令 $U'=U^G/\{a^4\}$, 删除其中的 $\bigcup\limits_{E^R.\mathrm{cons}=1}E^R$.

3.1) 使用算法 3 (U', a_1), 得到粗等价类 $\bigcup\limits_{E^R.\mathrm{cons}=1}E^R=\{\{e_1\},\{e_2\},\{e_5\}\}$, $\bigcup\limits_{E^R.\mathrm{cons}=-1}E^R=\{\{e_4\},\{e_6\},\{e_7\}\}$, 两种粗等价类符合剪枝策略, $\mathrm{sig}(a_4)=3$, $y=0$, $\Big|\bigcup\limits_{E^R.\mathrm{cons}=0}E^R\Big|=\varnothing$.

3.2) 使用算法 3 (U', a_2), 得到粗等价类 $\bigcup\limits_{E^R.\mathrm{cons}=1}E^R=\{\{e_2\}\}$, $\bigcup\limits_{E^R.\mathrm{cons}=0}E^R=\{\{e_1,e_4\},\{e_5,e_7\}\}$, $\bigcup\limits_{E^R.\mathrm{cons}=-1}E^R=\{\{e_6\}\}$, 1 和 -1-粗等价类符合剪枝策略, $\mathrm{sig}(a_2)=1$, $y=2$, $\Big|\bigcup\limits_{E^R.\mathrm{cons}=0}E^R\Big|\neq\varnothing$.

3.3) 使用算法 3 (U', a_3), 得到粗等价类 $\bigcup\limits_{E^R.\mathrm{cons}=1}E^R=\{\{e_1\},\{e_5\}\}$, $\bigcup\limits_{E^R.\mathrm{cons}=0}E^R=\{\{e_2,e_4\}\}$, $\bigcup\limits_{E^R.\mathrm{cons}=-1}E^R=\{\{e_6\},\{e_7\}\}$, 1 和 -1-粗等价类符合剪枝策略, $\mathrm{sig}(a_3)=2$, $y=1$, $\Big|\bigcup\limits_{E^R.\mathrm{cons}=0}E^R\Big|\neq\varnothing$.

第 2 轮计算结束, $a* = a_1$, 此时 $\left| \bigcup_{E^R.\mathrm{cons}=0} E^R \right| = \varnothing$, 因此, 约简 $R = \{a_1, a_4\}$. 调用算法 7 进行检验, 得到 $\mathrm{Red}(C) = \{a_1, a_4\}$.

5　实验与结果分析

5.1　实验设置

本文使用多种数据对算法进行验证. 其中: UCI 数据集见表 4, 海量和超高维数据集见表 5, 表中使用 D-ID 表示某个决策表.

表 4　UCI 决策集

D-ID	决策表	$\|U\|$	$\|U/C\|$	$\|A\|$	$\|\mathrm{Core}\|$	Red
1	Arrhythmia	452	452	280	0	2
2	Australian	690	690	15	1	3
3	breast1	198	198	35	0	1
4	breast2	569	569	32	0	1
5	Car	1 728	1 728	7	6	6
6	Credit	690	690	16	1	3
7	dermatology	366	366	35	0	6
8	german	1 000	1000	21	0	2
9	letter-recognition	20 000	18 668	17	3	11
10	nursery	12 960	5 784	9	8	8
11	poker	25 010	25 008	11	5	7
12	patient	90	74	9	8	8
13	shuttle	43 500	43 500	10	1	4
14	soybean-large	307	303	36	0	4
15	tic-tac-toe	985	958	9	0	8
16	waveform1	5 000	5 000	22	0	3
17	wine	178	178	14	0	2
18	mushroom	8 124	8 124	23	0	4
19	zoo	101	59	18	8	10
20	anneal	798	790	39	1	7

表 5　海量数据和超高维决策表

D-ID	决策表	$\|U\|$	$\|U/C\|$	$\|A\|$	$\|\mathrm{Core}\|$
21	Kddcup1	500 000	366 233	42	1
22	Kddcup2	1 000 000	582 539	42	4
23	Kddcup3	1 500 000	731 643	42	4
24	Kddcup4	2 000 000	731 643	42	4
25	Advertise	3 279	2 420	1 559	30

运行环境为 PC (Intel i7, 4 核, 2.4 GHz, 4 GB 内存), 用 Java 实现 Liu[19]、Ge1[18]、Ge2[17]、Zhao[23] 以及本文基于粗等价类的约简算法 RAR1 (包含求核) 和 RAR2 (不含求核). 各算法运行 20 次, 记录各指标值并取均值.

为便于比较, 表 4 和表 5 列出了基于全局等价类压缩以后的实体数 $|U/C|$. 决策表中包括了连续与离散数据的决策表, 由于离散化方法对约简结果有很大影响, 本文的离散化方法是: 决策表中每个数值对应一个编码, 最大程度地保留原决策表的决策能力. 由表 4 中 $|U/C|$ 属性可见, UCI 中决策表的可压缩程度较低, 大部分决策表压缩后的实体数与原决策表相等, 而表 5 中各决策表的压缩程度较高.

5.2　剪枝策略验证实验

本节对本文提出的横向及纵向剪枝策略进行验证. 剪枝策略可在求核与约简检验中使用, 由于求核算法中不删减属性, 剪枝策略效果更为明显, 本实验用各算法中的求核算法进行验证, 记录遍历的属性和实体数, 每次实验随机改变属性次序. 遍历属性数的对比用于验证纵向剪枝策略, 遍历实体数的对比用于验证横向剪枝策略. 在约简检验中, 该策略同样生效. 随机挑选多个 UCI 决策表进行该实验, 运算结果见表 6.

表 6　各约简算法在多个决策表遍历指标比较 (AP 和 UP 越小, 算法效率越高)

D-ID	$\|U\|$	$\|U/C\|$	$\|C\|$	$a*\|U\|$	$a*\|U/C\|$	核数	Liu		Ge1		Ge2		Zhao		RAR1 & RAR2	
							AP	UP	AP	UP	AP	UP	AP	UP	AP	UP
6	690	690	15	144 900	144 900	1	1	1	1	1	1	1	1	0.965 3	0.458 3	0.090 3
9	20 000	18 668	16	4 800 000	4 480 320	3	1	0.933 4	1	0.933 4	1	1	1	0.922 4	0.921 9	0.359 1
11	25 010	25 008	10	2 250 900	2 250 720	5	1	0.999 9	1	0.999 9	1	1	1	0.719 7	0.950 0	0.615 8
18	8 124	8 124	22	3 753 288	3 753 288	0	1	1	1	1	1	1	1	1	0.721 6	0.135 5
20	798	790	38	1 121 988	1 110 740	1	1	0.989 97	1	0.989 97	1	1	1	0.978 1	0.631 6	0.149 7

*注 $a = |C||C-1|$

为说明实验结果, 定义两个指标: $\mathrm{AP} = \mathrm{AC}/|C|$, $\mathrm{UP} = \mathrm{AE}/|U|$. 其中: AC 表示平均遍历属性数, 对每次运行遍历的属性数加总, 除以运行次数, 得到 AC 值; AE 表示平均遍历实体数, 计算方法与 AC 类似; U 是信息系统的论域. AC 在最差情况下等于 $|C|$, 即每次需要遍历全部属性; 同理, AE 在最差情况下等于 $|U|$. AP 和 UP 取值范围均为 $(0, 1]$, 均具有相同性质: 值越小, 算法遍历的属性和实体数越少, 效率越高.

Liu、Ge1、Ge2 和 Zhao 用经典思路求核算法, 通过判断 $\mathrm{POS}_C(D) \neq \mathrm{POS}_{C-\{a\}}(D)$ 来检验每个属性,

因此 AP 均为 1. EHash 和 Ge1 使用了压缩决策表, 不需要遍历全部实体, UP 有一定程度的下降, 但下降程度低于 Zhao 和本文算法. Zhao 算法使用不一致性检测的停止搜索策略, 在大多数情况下生效, 有效减少了遍历实体数, 但当信息系统无核情况下 (如在 18-mushroom 决策表上), 则无法缩减计算域; 同时, Zhao 算法也无法减少遍历属性数. 本文算法在所有决策表上的 AP $<$ 1, 说明纵向剪枝策略生效, 包括在无核情况下, 所有决策表上 UP $<$ 1, 说明横向剪枝策略生效, 双向剪枝策略使遍历属性和实体数均大幅度下降, 确保了粗等价类下求核与约简的高效性.

5.3 约简算法实验

5.3.1 UCI 决策集实验

各约简算法在 UCI 决策集上运行时间如表 7 所示. Ge1 和 Ge2 算法的最大差别在于 Ge1 使用基于全局等价类计算粒度的数据压缩策略, Ge2 使用原始决策表. 由于大部分决策表未能压缩或者压缩程度较小, Ge1 和 Ge2 性能差别非常小, 但决策表 10-nursery 压缩程度较高, 在该表上两算法显示出较大差异, 说明当信息系统压缩程度较高时, 压缩策略可显著提高效率.

表 7 各约简算法在 UCI 数据集上运行时间对比 ms

D-ID	Liu	Ge1	Ge2	Zhao	RAR1	RAR2	RAR2/RAR1	最快
1	471.80	8 144.95	8 112.95	665.05	27.30	24.20	0.89	RAR2, RAR1
2	85.00	36.05	35.85	15.70	4.65	3.00	0.65	RAR2, RAR1
3	2.30	134.30	135.05	9.05	3.15	0.80	0.25	RAR2, RAR1 , Liu
4	7.05	389.40	387.35	21.60	3.85	2.45	0.64	RAR2, RAR1 , Liu
5	14.80	16.35	17.85	1.80	7.90	7.15	0.91	RAR2, RAR1 , 本文
6	12.50	41.35	39.90	5.90	3.15	3.80	1.21	RAR2, RAR1 , Zhao, Liu
7	46.25	58.10	59.05	19.10	11.45	2.30	0.20	RAR2, RAR1 , Zhao
8	13.35	78.80	78.95	11.70	8.50	0.80	0.09	RAR2, RAR1 , Zhao, Liu
9	8 256.05	2 653.15	2 881.85	2 074.15	967.30	610.90	0.63	RAR2
10	134.50	139.80	397.90	49.55	47.55	43.55	0.92	RAR2, RAR1 , Zhao
11	7 567.20	1 187.40	1 159.75	3 008.40	713.85	482.45	0.68	RAR2
12	1.55	1.55	2.40	0.60	0.80	0.75	0.94	/
13	907.20	2 112.95	2 118.20	1 195.55	460.40	311.95	0.68	RAR2
14	20.25	32.10	33.50	11.85	3.15	0.75	0.24	RAR2, RAR1, Zhao
15	39.70	26.45	26.70	32.05	13.20	5.45	0.41	RAR2, RAR1
16	169.45	1 551.85	1 571.70	100.95	63.30	54.55	0.86	RAR2, RAR1
17	0.75	15.85	15.45	1.05	0.00	0.75	/	RAR2, RAR1, Zhao, Liu
18	477.25	814.50	798.75	572.85	166.40	56.20	0.34	RAR2
19	3.90	2.30	3.15	3.85	1.60	0.75	0.47	/
20	109.20	147.10	143.10	26.00	15.40	8.50	0.55	RAR2, RAR1

在大多数决策表上, Ge2 算法效率最低; Liu 算法和 Ge2 均使用数据压缩策略, 在决策表 5、10、12 上效率接近, 因为这几个决策表的核是约简; 此时 Ge1 算法仅求核和进行约简检验即可完成, 此时效率与不求核的 Liu 算法效率相近.

在其他决策表上, Liu 算法基于 Hash 的设计, 约简中不求核, 在大多数情况下性能优于 Ge1 和 Ge2 算法; 但 Liu 算法未使用删减计算域策略, 性能低于 Zhao 算法. Zhao 算法在求核阶段使用了停止搜索策略, 并对 Hash 中的 key 使用数字计算替代, 把大量的求 key 计算的时间复杂度从 O(key) 降至 O(1), 并且使用单属性正区域替代经典的属性重要度, 计算量较小, 因此效率较高. 本文算法使用了多种策略, 包括数据压缩、双向剪枝, 而且通过数据增量划分算法减少重复计算, 故大多数情况下 RAR1 和 PAR2 算法性能均更佳, RAR2 略优于 RAR1, 尤其在属性数 (>20) 和实体数较多情况下, 效率明显优于其他算法. 下面对 RAR1 和 RAR2 进行比较. 本文两种算法在 UCI 数据集上运行时间对比如图 1 所示.

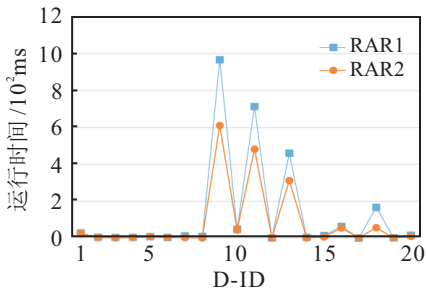

图 1 本文两种约简算法在 UCI 数据集上运行时间对比

由表 7 和图 1 可见, 当实体数低于 8 000 时, 本文的两种约简算法效率相近, RAR2 效率略优于 RAR1; 但当实体数多于 8 000 时, 两者的差距增大. 由图 1 可见, 在决策表 9、11、13、18 上, 两种算法的运行时间差值较大; 而在决策表 3、7、8、14、15、18、19 上的运行时间比值差异较大, 在此类决策表上 RAR2 不到 RAR1 时间的一半. 上述两种情况说明, 实体数或属性数量越多, 不求核约简算法的效率越高.

5.3.2 海量和超高维数据集实验

各约简算法在海量和超高维数据集上的运行时间见表 8. 海量和超高维数据集的可压缩程度较高, 使用压缩策略后计算域大大下降, 因此, Ge1 算法效率较大幅度地优于 Ge2 算法. 在 Kddcup1 和 Kddcup2 上, Liu 算法较大幅度地优于 Ge1 算法, 但不如 Zhao 算法. 当数据量增加时, 在 Kddcup3 和 Kddcup4 上, 由于 Liu、Zhao 算法在约简过程中无剪枝策略, 两算法效率大幅度下降, Liu 算法效率优于 Zhao 算法; 在 Kddcup4 上, Ge1 算法优于 Liu 和 Zhao 算法. 实验数据表明, 剪枝策略对提高效率有显著效果, 数据量越大, 剪枝策略带来的效率提升效果越显著. 本文两种算法均使用双向剪枝策略, 显示出良好性能, 优于在 Ge1 单边横向剪枝策略, 效率显著高于其他算法. 在超高维数据上, Liu 算法效率最低, 运算时间大幅度地高于其他算法; Zhao 算法效率优于本文以外的其他算法; 在大多数情况下, 本文两种算法效率均显著地高于其他算法. 下面对本文两种算法进行比较.

表 8　各约简算法在海量和超高维数据上运行时间对比　　ms

D-ID	Liu	Ge1	Ge2	Zhao	RAR1	RAR2	最快
21	112 458	317 226	420 787	64 528	24 659	5 924.25	RAR2
22	578 000	527 490	888 493	209 707	49 898	12 023	RAR2
23	479 748	710 382	1 390 255	2 021 341	66 045	17 099.4	RAR2
24	1 310 017	684 569	1 955 170	2 103 442	99 645	41 231	RAR2
25	25 102 052	1 935 648	3 861 587	495 424	56 854	3 417	RAR2

本文两种算法在海量和超高维数据集上的运行时间对比如图 2 所示. 由表 8 和图 2 可见, 本文两种算法在海量和超高维数据集上的差距显著增大. 当 KDDCup 数据集的数据量线性增加时, 本文两种算法的运行时间也基本呈线性增长趋势, RAR1 的增幅大于 RAR2, 说明处理海量数据和超高维数据时, 不求核比求核具有更大优势. 在超高维数据集上, RAR2 也较为显著地优于 RAR1, 说明本文不求核的约简算法尤其适用于大型和超高维数据集.

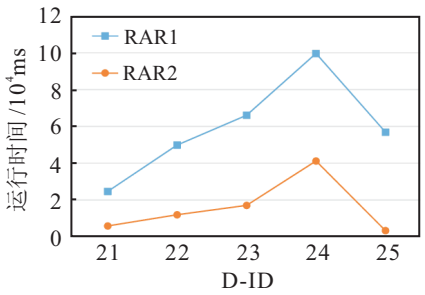

图 2　本文两种约简算法在海量和超高维数据集上运行时间对比

5.4 约简质量对比

5.4.1 UCI 数据集实验

UCI 上各算法约简长度见表 9, 可见各算法约简长度几乎无差别, 除了决策表 18、19、20 上略有差别, 原因是各算法使用的属性重要度不同而形成差异. Liu、Ge1、Ge2 和本文两种算法均使用正区域增量作为属性重要度 sig, 但仍有差别. 本文两种算法除使用 sig 外, 还使用属性对 0-粗等价类的划分能力作为第 2 选择指标. Liu 和 RAR2 不求核, 直接求约简, 在有核决策表中形成的约简属性序列与 Ge1、Ge2 和 RAR1 均有差别. Ge1 算法把不一致决策表处理成一致决策表, 可能使属性重要度计算结果与原始决策表形成一定差异. Zhao 算法使用单个属性的正区域大小作为属性重要度.

综上所述, 在决策集属性和实体数均较小的情况下, 是否压缩决策表, 使用何种指标选择属性, 对于约简长度无明显影响, 尤其在属性数较少的情况下.

表 9　各约简算法在 UCI 数据集上约简长度对比

D-ID	Liu	Ge1	Ge2	Zhao	RAR1	RAR2
1	2	2	2	2	2	2
2	3	3	3	3	3	3
3	1	1	1	1	1	1
4	1	1	1	1	1	1
5	6	6	6	6	6	6
6	3	3	3	3	3	3
7	6	6	6	8	6	6
8	2	2	2	2	2	2
9	11	11	11	11	11	11
10	8	8	8	8	8	8
11	7	7	7	7	7	7
12	8	8	8	8	8	8
13	4	4	4	4	4	4
14	4	4	4	4	4	4
15	8	8	8	8	8	8
16	3	3	3	3	3	3
17	2	2	2	2	2	2
18	5	5	4	5	5	5
19	11	11	10	10	10	11
20	7	7	7	8	7	7

5.4.2 海量与超高维数据集实验

各算法在海量数据集和超高维数据集上的约简长度见表 10. 虽然差别较小, 但在所有数据集上均存在差别, 除 21-数据集外, Liu 和 RAR2 在其他数据集上均得到最小约简, 略优于其他算法. 这在一定程度上说明, 不以核为起点的约简, 有可能使具有高划分能力的非核属性先并入约简属性集, 从而形成更优解. 在超高维数据上, 各算法间差异增大, Zhao 算法结果

的偏差被放大, 说明使用单属性正区域大小作为选择属性的指标不佳. 其他算法使用 sig 作为选择标准, 约简结果均较理想.

表 10 各约简算法在海量和超高维数据集上长度比较

D-ID	Liu	Ge1	Ge2	Zhao	RAR1	RAR2
21	5	4	4	4	4	5
22	7	8	8	8	8	7
23	9	10	10	10	10	9
24	9	10	10	10	10	9
25	77	82	80	1 478	80	77

进一步, 把 Liu、Ge1、Ge2 和本文两种算法在超高维数据集上的解在检验前后的约简结果进行对比, 见表 11. 由表 11 可知, 以核为起点的约简算法在检验前能求得较优的约简, 检验工作量较小. Liu 算法约简在检验前最长, 但在检验后可得到最小约简, 原因与前面分析类似. 重要度更高的非核属性优先进入约简, Liu 的贪婪算法以较高的检验计算成本可能得到更优解. RAR2 检验前的结果优于 Liu 算法, 说明本文算法选择属性的双指标优于 Liu 的单指标; 检验后的结果与 Liu 算法相同, 优于其他算法.

表11 各算法在超高维数据上约简检验前后解长度比较

D-ID	Liu		Ge1		Ge2		AR1		AR2	
	前	后	前	后	前	后	前	后	前	后
25	90	77	86	82	83	80	82	80	86	77

5.4.3 实验总结

由上述分析可见, 本文两种算法在大多数情况下均优于同类算法, 其中 AR2 优于 AR1 算法, 原因分析如下:

1) 本文提出粗等价类概念, 细分出 3 类粗等价类, 设计 0-粗等价类下正区域和约简的等价计算方法, 从而给出面向 1 和 −1-粗等价类的双边剪枝策略, 可在计算过程中不断有效减少计算量, 优于现有算法的单边剪枝策略[16-18].

2) 本文提出了删减遍历属性的纵向剪枝策略, 可用于求核或者约简检验, 有效缩减计算域.

3) Hash 在求等价类等基础算法上有着天然优势, 优于现有基于排序算法[16-18]的等价类算法. 虽然两类算法时间复杂度相同, 但基于 Hash 的算法遍历论域次数大大少于基于基数排序的算法, 因此实际运行效率较高.

4) 设计多次 Hash 的属性增量划分算法, 通过增量计算可有效避免重复计算, 效率优于二次 Hash 的设计[19].

5) 通过大量实验验证, 约简中不含求核时, 约简效率可能更高且结果质量更优, 以此作为约简算法设计的基础, 从而给出不含求核的高效完备约简算法.

6) 使用双指标选取最大增益属性, 可能使具有高分辨能力的属性优先入选, 使约简质量更高, 在一定程度上优于单一选取指标. 同时, 属性选取指标的计算转换为 1 和 0-粗等价类的增量计算, 计算简单、易于实现.

7) 本文算法无需增加任何额外操作, 即可支持不一致数据约简.

6 结 论

本文提出了一种新的约简算法. 首先以全局等价类作为基本计算粒度, 提出粗等价概念, 使计算域从 $|U|$ 下降到 $|U/C|$. 深入研究粗等价类性质, 细分出 0、1 和 −1 三种粗等价类, 证明在粗等价类下的求核和约简与原始信息系统等价. 分析粗等价类与正区域的内在联系, 证明可以通过对 1 和 −1-粗等价类的双边剪枝不断减少计算域, 设计正区域的等价计算方法, 从而提出横向删减 1 和 −1-粗等价类的双边剪枝策略, 同时提出可以减少遍历属性的纵向剪枝策略. 对求约简是否应包含求核进行了探讨, 大量实验表明, 不求核约简算法在一定程度上, 其效率和约简质量均优于含求核的约简算法. 最后, 给出了完整的不含求核的高效完备约简算法. 通过 UCI 中 20 个决策表, KDDCup 海量和超高维数据集, 从遍历属性和实体数、约简时间、约简质量多方面对本文算法和多个算法进行了对比. 综上所述, 本文算法在大多数情况下, 无论信息系统属性多或少, 实体数多或少, 均具有较好的性能, 双向剪枝策略可以有效提升约率, 基于多次 Hash 的增量划分算法可以减少重复计算, 尤其适用于海量和超高维数据集.

参考文献(References)

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

[2] 王珏, 王任, 苗夺谦. 基于 Rough Set 理论的“数据浓缩”[J]. 计算机学报, 1998, 21(5): 393-400.
(Wang J, Wang R, Miao D Q. Data enriching based on rough set theory[J]. Chinese J of Computers, 1998, 21(5): 393-400.)

[3] Chen D, Li W, Zhang X, et al. Evidence-theory-based numerical algorithms of attribute reduction with neighborhood-covering rough sets[J]. Int J of Approximate Reasoning, 2014, 55(3): 908-923.

[4] 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.

[5] Wang C, Shao M, Sun B, et al. An improved attribute reduction scheme with covering based rough sets[J]. Applied Soft Computing, 2015, 26: 235-243.

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

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

[8] Zhang Z, Tian J. On attribute reduction with intuitionistic fuzzy rough sets[J]. Int J of Uncertainty, Fuzziness and Knowledge-Based Systems, 2012, 20(1): 59-76.

[9] Xu W, Zhang X, Zhong J, et al. Attribute reduction in ordered information systems based on evidence theory[J]. Knowledge and Information Systems, 2010, 25(1): 169-184.

[10] Feng T, Zhang S, Mi J. The reduction and fusion of fuzzy covering systems based on the evidence theory[J]. Int J of Approximate Reasoning, 2012, 53(1): 87-103.

[11] Wang F, Liang J, Dang C. Attribute reduction for dynamic data sets[J]. Applied Soft Computing, 2013, 13(1): 676-689.

[12] Yang X, Qi Y, Yu H, et al. Updating multigranulation rough approximations with increasing of granular structures[J]. Knowledge-Based Systems, 2014, 64: 59-69.

[13] Zheng K, Hu J, Zhan Z, et al. An enhancement for heuristic attribute reduction algorithm in rough set[J]. Expert Systems with Applications, 2014, 41(15): 6748-6754.

[14] 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 S H, Sheng Q J, Shi Z Z. A new method for fast computing positive region[J]. J of Computer Research and Development, 2003, 40(5): 637-642.)

[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 S H, Sheng Q J, Wu B, et al. Research on efficient algorithms for rough set methods[J]. Chinese J of Computers, 2003, 26(5): 524-529.)

[16] 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 Z Y, Liu Z P, Yang B R, et al. A quick attribute reduction algorithm with complexity of $\max(O(|C||U|), O(|C^2|U/C|))$[J]. Chinese J of Computers, 2006, 29(3): 391-399.)

[17] 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 H, Li L S, Yang C J. An efficient attribute reduction algorithm based on conflict region[J]. Chinese J of Computers, 2012, 35(2): 342-350.)

[18] 葛浩, 李龙澍, 杨传健. 基于冲突域渐减的属性约简算法[J]. 系统工程理论与实践, 2013, 33(9): 2371-2380.
(Ge H, Li L S, Yang C J. Attribute reduction algorithm based on conflict region decreasing[J]. Systems Engineering — Theory & Practice, 2013, 33(9): 2371-2380.)

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

[20] Guan J W, Bell D A. Rough computational methods for information systems[J]. Artificial Intelligence, 1998, 105(1): 77-103.

[21] 杜金莲, 迟忠先, 翟巍. 基于属性重要性的逐步约简算法[J]. 小型微型计算机系统, 2003, 24(6): 976-978.
(Du J L, Chi Z X, Zhai W. An improved algorithm for reduction of knowledge based on significance of attributes[J]. Mini-micro Systems, 2003, 24(6): 976-978.)

[22] 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 H, Li L S, Yang C J. Incremental updating algorithm of the computation of core based on the collision[J]. Control and Decision, 2011, 26(7): 984-990.)

[23] 赵洁, 梁俊杰, 董振宁, 等. 位运算和核属性快速识别下的粗糙集属性约简算法研究[J]. 小型微型计算机系统, 2015, 36(2): 316-321.
(Zhao J, Liang J J, Dong Z J, et al. Rough set attribute reduction algorithm using bit arithmetic and core attributes quick identification[J]. Mini-micro Systems, 2015, 36(2): 316-321.)

[24] 赵洁, 梁俊杰, 董振宁, 等. 基于全局正区域不一致性的快速求核算法[J]. 计算机科学, 2015, 42(8): 259-264.
(Zhao J, Liang J J, Dong Z N, et al. Global positive region inconsistency based attributes core computation[J]. Computer Science, 2015, 42(8): 259-264.)

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

[26] 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 H, Yang C J, Li L S. Efficient algorithm for computing core attributes[J]. Computer Engineering and Applications, 2010, 46(26): 138-141.)

[27] 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 H, Li L S, Yang C J. Quick algorithm for computing core attribute[J]. Control and Decision, 2009, 24(5): 738-742.)

(责任编辑：李君玲)