决策表的属性约减算法综述

张任伟[1] 白晓颖[2] 郁 莲[1] 陆 皓[2]
(北京大学软件与微电子学院 北京 102600)[1] (清华大学计算机科学与技术系 北京 100084)[2]

摘 要 介绍决策表的基本概念,分析决策表冲突条件以及判定条件组合爆炸的问题,明确决策表冲突检测属性约减的必要性。从本质描述、算法分类、算法效率等几个方面,对决策表冲突检测算法和属性约减算法进行了系统的综述和比较。最后,探讨了决策表的构建、效率、应用规模等6个属性约减研究的热点问题。

关键词 决策表,冲突检测,属性约减
中图法分类号 TP311 **文献标识码** A

Survey of Decision Table Research of Attribute Reduction

ZHANG Ren-wei[1] BAI Xiao-ying[2] YU Lian[1] LU Hao[2]
(School of Software and Microelectronics, Peking University, Beijing 102600, China)[1]
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)[2]

Abstract The paper reviewed the basic concept of decision tables. It analyzed the problems of conditions confliction and necessity of attribute reduction when a decision table, with a large number of conditions, has the issue of combinatorial explosion. The paper surveyed the state-of-the-art research on the above two problems, and compared their algorithms from different aspects including motivations, approaches and efficiencies. It then discussed research challenges and future research focuses including decision table construction, algorithm scalability and efficiency.

Keywords Decision table, Conflict detection, Attribute reduction

1 引言

决策表又称判定表,用精确而简洁的方式描述了复杂的决策情形。决策表可以表达对象属性间的逻辑关系,推理不同条件下的决策动作。决策表最初常用于计算机编程。随着人们对决策表研究的不断深入,它的应用领域也开始向其它具有复杂逻辑的领域扩展。主要应用领域包括医疗专家系统中的知识获取、一般的复杂过程化决策情形的模拟与验证、银行和保险公司中利率和奖金的决策计算以及基于决策表模型的测试用例生成等。

伴随着决策表在实际项目中的应用越来越多,这种基于数据的推理工具也暴露出一些问题,集中体现在以下2个方面。

1)决策表的规则之间可能存在着冗余或冲突。在一个决策表中,如果存在两条规则的条件属性相同而决策属性不同,称该决策表存在冲突,或称该决策表为不一致的。同样地,如果一个决策表中存在两条规则的条件属性相同而决策属性也相同,称该决策表是冗余的。

2)决策表的属性之间可能存在着冗余。如果决策表的一个条件属性在被删去之后,并不影响决策表的决策属性的取值,称该属性为冗余的。

针对上述问题,决策表的冲突检测和属性约减的研究就显得尤为重要。对于决策表规则的冲突检测,常转化为对决策表内规则的一致性检测,依次判定表内任意两条规则是否存在冲突。如果决策表过于庞大,或属性数量过于繁杂,这样的算法在实现时需花费很大的代价。对决策表的属性进行约减,不仅可以理顺条件属性间的关系,去除不必要的冗余属性,还有利于检测决策表规则的冲突与冗余。然而,利用算法直接进行约减显然是不现实的。一方面,这样的算法必然存在着效率低下的问题。另一方面,如何对冗余属性做出精确的定义,也是我们需要解决的难点。

本文从决策表的基本概念出发,分析了决策表中的冲突检测与属性约减问题,并介绍了常用的冲突检测与属性约减算法。最后,从决策表的构建、效率、应用规模等6个方面提出了决策表属性约减的研究热点。

2 决策表基本概念

决策表通常用于描述或处理判断条件较多、各条件又相互组合、有多种决策方案的情况。通过精确而简洁地描述复杂逻辑的方式,将多个条件与这些条件满足后要执行的动作相对应。在决策表中,这些条件称为条件属性,对应的执行动作称为决策属性。一组条件属性连同其对应的决策属性称为

到稿日期:2010-12-31 返修日期:2011-03-31 本文受航空科学基金自主项目(20091958005)资助。
张任伟 男,硕士生,主要研究方向为软件工程和软件测试,E-mail:Edward.gaist@gmail.com;白晓颖 女,副教授,主要研究方向为软件工程、软件测试和软件项目管理等;郁 莲 女,副教授,主要研究方向为面向服务的软件工程、软件质量保证、情境感知服务、分布式软件开发等;陆 皓 男,硕士生,主要研究方向为软件工程和软件测试。

决策表中的一个实例，又称为一条规则。

对于决策表的形式化定义，通常使用粗糙集理论进行描述。波兰学者 Pawlak 提出的粗糙集理论的主要思想是利用已知的知识库，将不精确或不确定的知识用已知的知识库中的知识来(近似)刻画，以有效地分析不精确、不一致(inconsistent)、不完整(incomplete)等各种不完备的信息，并对数据进行分析和推理，从中发现隐含的知识，揭示潜在的规律。

在粗糙集理论中，将决策表看作为一个决策信息系统[1]。决策信息系统是一种基于信息表的知识表达形式，用逻辑、规则以及形式化方法将现实存在的知识抽象为计算机理解的语义、规则。其基本成分包括研究对象的集合及对象所具有的知识，这些知识通过定义对象属性具体的值来描述。在决策信息系统中，又将属性划分为条件属性以及决策属性。决策信息系统的定义如下：

定义 1[2] 决策信息系统 S 可以用一个 4 元组(U,A,V,f)来表示。其中 U 是对象的非空有限集合，即论域；A 是属性的非空有限集合。通常分为条件属性 C 和决策属性 D，$A=C\cup D$，$V=\bigcup_{a\in A}V_a$，V_a 是属性的值域。$f:U\times A\to V$ 是一个赋值函数，它为每个对象的每个属性赋予一个具体的值，即 $\forall a\in A,x\in U,f(x,a)\in V_a$。

在决策信息系统中，论域内的实例可能会具有某些必要的属性，并且由这些属性能够将其划分到不同的类别。如果两个实例的必要属性具有相同的值，则称这两个实例具有不可分辨关系，见表 1。

表 1 不同图形的决策表

图形实例	边数	颜色	图形
1	3	红	三角形
2	3	蓝	三角形
3	4	红	四边形

表 1 中，1，2，3 为该决策信息系统的 3 个实例。“边数”、“颜色”为其条件属性，“图形”为其决策属性。我们知道，一个图形是三角形还是四边形是由其边数决定的。因此，在这个例子中，边数为该决策信息系统的必要属性，通过边数这个属性可以将图形进行分类。实例 1，2 因为有相同的“边数”属性值，所以这两个实例具有不可分辨关系。关于等价类、不可分辨关系的定义如下：

定义 2[2] 设 R 是 U 上的一个等价关系。$U/R=\{X_1,X_2,\cdots,X_n\}$表示 R 产生的分类，$[X]_R=\{y\in U\mid xRy\}$表示关系 R 下包含元素 x 的等价类。(U,R)称为近似空间。

设 $P\in A$，则 P 中的全部等价关系的交集称为 P 上的不可分辨关系，记为 $IND(P)$：$IND(P)=\{(x,y)\in U\times U,\forall a\in P,f(x,a)=f(y,a)\}$。

表 1 的例子里，“颜色”这一属性同样存在着不可分辨关系，存在着等价类的划分。但颜色的取值“红”与“蓝”并不会影响“图形”这一属性的判断。如何准确判定某些实例是否属于某一个“概念”(如三角形、四边形等描述一类实例典型特点的实例子集)，是决策信息系统中很重要的一个问题。在粗糙集理论中，定义了上近似集、下近似集的概念，以解决这一问题。

定义 3[1] 对于每一个概念 X(实例子集)和不可分辨关系 R，包含于 X 中的最大可定义子集和包含 X 的最小可定义集都能根据 R 确定，前者称为 X 的下近似集(记为 R_X)，后者称为 X 的上近似集(记为 $R^- X$)。

定义 4[3] 设集合 $X\subseteq U$，R 是一个等价关系，称 $R_X=\{x\mid x\in U,且[x]_R\subseteq X\}$为集合 X 的 R 下近似集；称 $R^- X=\{x\mid x\in U,且[x]_R\cap X\neq\emptyset\}$为集合 X 的 R 上近似集。称 $POS_R(X)=R\_X$ 为 X 的 R 正域；称 $NEG_R X=U-R^- X$ 为 X 的 R 负域。

引入正域的概念之后，对于决策表的冗余属性以及一致性判断，也就有了形式化的定义。

定义 5 在决策表 $S=(U,A,V,f)$ 中，若 $a\in A$，且 $POS_{A-a}(V)=POS_A(V)$，则称属性 a 为 S 中的冗余属性。

定义 6[3] 在决策表 $S=(U,A,V,f)$ 中，若 $POS_A(V)=U$，则称决策表 S 为一致决策表，否则称 S 为不一致决策表。

3 决策表的冲突检测和属性约减问题

决策表中，如果任意两个实例的条件属性相同而决策属性不同，则说明该决策表存在着冲突。对于这种不一致的决策表，通常会给实际的工程应用带来很大的问题。例如，基于决策表的测试(DecisionTable-based Testing)中，一个必要的测试准则就是确认决策表的一致性。否则，通过冲突的决策表模型将无法生成测试用例或生成错误的测试用例，严重影响测试结果的正确性。因此，研究决策表的冲突检测算法，将对实际应用提供非常大的帮助。

然而，仅仅进行决策表的冲突检测是不够的。工程中的决策表通常是庞大且杂乱无序的，这就会带来两方面的问题。第一，决策表的属性存在冗余。这些无用的属性会使得决策表内规则的表现形式过于拖沓，不利于把握规则的核心属性。而且，过多的冗余属性也会给决策表的冲突检测带来困扰，加剧冲突检测所耗费的时间。第二，决策表的样本(实例)过于繁冗。决策表中的样本往往只是机械地记录每一个实例的实际情况，其规则并不具有很强的适用性。这样的样本越多，只会加重机器阅读的负担，不利于核心决策规则的提取。

表 2 中提供的例子是我们实际工程中的一个应用。

表 2 CREATPROCESS 函数输入参数及返回值决策表

	1	2	3	4	5	6
NAME	VALID	INVALID	VALID	VALID	VALID	VALID
ENTRY_POINT	ANY	ANY	ANY	ANY	ANY	ANY
STACK_SIZE	VALID	ANY	INVALID	VALID	VALID	VALID
BASE_PRIORITY	VALID	ANY	ANY	INVALID	VALID	VALID
PERIOD	VALID	ANY	ANY	ANY	INVALID	VALID
TIME_CAPACITY	VALID	ANY	ANY	ANY	ANY	INVALID
DEADLINE	ANY	ANY	ANY	ANY	ANY	ANY
FLOAT_REQUIREMENT	ANY	ANY	ANY	ANY	ANY	ANY
RETURN_CODE	NO_ERROR	NO_ACTION	INVALID_PARAM	INVALID_PARAM	INVALID_PARAM	INVALID_PARAM

这是一个测试操作系统函数的决策表模型。其条件属性为函数的输入参数，即 NAME，ENTRY_PIOINT 等，决策属性为函数的返回值，即 RETURN_CODE。在这个决策表中，ENTRY_POINT，DEADLINE，FLOAT_REQUIREMENT 3 个属性均为冗余属性，故可以约去。另一方面，根据该决策表，条件属性有 8 个，每个条件属性的值域都是{VALID，IN-

VALID}，因此，总共将生成 $2^8=256$ 条规则（由于篇幅关系，将某些规则的属性用 ANY 替代，表示可以取 VALID 或者 INVALID）。而通过属性约减，提取出的核心规则数仅为 $2^5=32$ 条。相应地，基于决策表生成的测试用例数目也将大大减少。

因此，决策表的属性约减是一个决策信息系统不可缺少的环节，通过约减，删去无用的属性，使得决策信息系统的逻辑关系更为清晰，便于核心规则的提取。

4 研究现状分析

4.1 决策表冲突检测的主要方法

决策表的冲突检测，通常可以转化为对决策表内规则的一致性检测[4]。接下来介绍的这种算法，就是采用的这种思路。算法通过遍历决策表内的规则，检查任意两条规则是否一致。如果不一致，则返回；否则，继续遍历，直至结束。

设决策表内的规则集为 R，$R=\{r_1,r_2,\cdots,r_n\}$，$n=|R|$。对于 R 内的每一条规则 r，LHS(r)表示 r 的条件属性值的集合，$RHS(r)$表示 r 的决策属性值的集合，incompatible 关键字表示两个集合的元素不能同时相同。函数 Direct-Confliction (r_x,r_y)表示 r_x,r_y 的冲突关系，若其值为 true，则说明两规则存在冲突。

算法 1[5] Check directly conflicting rules

输入：决策表内的规则集 R，$R=\{r_1,r_2,\cdots,r_n\}$，$n=|R|$

输出：决策表的冲突判断

Step 1 Let $S=(r_1,r_2,\cdots,r_n)$，$n=|S|$

Step 2 for $i=1$ to $n-1$ do

Step 2.1 for each rule r_x from r_i

Step 2.2 for $j=i+1$ to n do

Step 2.3 for each rule r_y from r_j do

Step 2.4 if LHS(r_x) ⊆LHS(r_y)
then if RHS(r_x)∩RHS(r_y)=∅ or
RHS(r_x) and RHS(r_y) are incompatible
then Direct-Confliction(r_x,r_y) ←true
return

Step 2.5 Direct-Confliction(r_x,r_y) ← false

Step 3 end

决策表的一条规则中，某些条件属性可能会取空值，表示该属性取任意值时，均不改变这条规则的决策属性取值。因此，算法 1 在判断两条规则一致关系时，采用的是 LHS(r_x)⊆LHS(r_y)而不是 LHS(r_x)=LHS(r_y)，使得算法 1 更具有通用性。

从该算法中不难看出，算法的时间复杂度为 $O(n^4)$。因此该算法更适合用于规则集较少的决策表，当决策表增加到一定规模时，算法的成本是难以容忍的。然而，在实际项目中，不可避免会有大规模决策表应用的例子。因此，对决策表属性约减的研究就十分必要了。

4.2 决策表属性约减的主要方法

由于粗糙集理论研究的是不完整、不确定知识和数据的表达、归纳方法，可以查找决策表中的冗余属性，因此基于粗糙集的决策表属性约减方法得到了越来越广泛的关注。基于粗糙集的约减方法其主要思想是归纳已有的决策规则信息，用数学公式定量地对每一个决策表内的每一个条件属性进行冗余分析，删除存在的冗余属性。近年来，有大量学者在此方面开展了一定的相关研究。对这些约减方法归类之后，通常可将其分为 3 类：基于代数方法的约减、基于信息熵方法的约减，以及基于区分矩阵的约减。

4.2.1 基于代数方法的约减

对决策表 S 进行约简，就是删除决策表中无效的或可有可无的等冗余属性。在一个知识表达系统中，属性 $a(a\in B,B\subseteq A)$是冗余的，当且仅当 $IND(B-\{a\})=IND(B)$，否则，就说属性 a 不可或缺。类似地，在决策信息系统中，条件属性 a $(a\in B,B\subseteq C)$是冗余的，当且仅当 $POS_{B-\{a\}}(D)=POS_B(D)$；否则，就说条件属性 a 不可或缺。因此，从代数角度考虑，基于代数方法的约减，就是对每个条件属性，验证条件集在没有该属性时的 D 正域是否与之前的相等。

定义 7[6] 给定决策表 $S=\langle U,A,V,f\rangle$，$A=C\cup D$，$B\subseteq C$，若 $POS_B(D)=POS_C(D)$且对于任意的 B'，有 $POS_{B'}(D)\neq POS_C(D)$，则称 B 是 C 相对于 D 的属性约简，称为代数约简。

根据定义 7，判断 $B\subseteq C$ 是否为 C 的代数约简的关键在于判断是否有 $POS_B(D)=POS_C(D)$成立。而判断两个集合是否相等是比较费时的。对于这个问题，王国胤博士、黄国顺教授各自提出了一些简化算法。

王国胤博士在文献[7]中提到，对于属性的代数约减，可以简化为判断属性的相关重要度 $Sgf(a,B,D)$，其中 $B\subseteq A$。并给出了如下计算公式：

$$Sgf(a,B,D)=POS_{B+\{a\}}(D)-POS_B(D)$$

如果求出 $Sgf(a,B,D)$为 0，则说明属性 a 对于属性集合 B 是冗余的。而通过求各条件属性的相关重要度，就能求出原属性集的约减。

黄国顺教授也给出一种简便方法，它将判断两正域是否相等简化为判断它们的基数是否相等，得到定理 1。

定理 1[8] 给定决策表 $S=\langle U,A,V,f\rangle$，$A=C\cup D$，$B\subseteq C$，$POS_B(D)=POS_C(D)$的充分必要条件是：

$$|POS_B(D)|=|POS_C(D)|$$

定理证明请参见文献[8]。

4.2.2 基于信息熵方法的约减

信息的基本作用就是消除人们对事物了解的不确定性。美国信息论创始人香农发现任何信息都存在冗余，冗余的大小与信息的每一个符号出现的概率和理想的形态有关，香农借鉴了热力学的概念，把信息中排除了冗余后的平均信息量称为“信息熵”，并给出了计算信息熵的数学表达式。从这个角度讲，可以用“信息熵”来确定一个决策信息系统的核心属性，并对非核心属性进行约减。这就是基于信息熵方法约减的基本思想。

信息熵的概念与概率论密不可分。设 U 为一个论域，可以认为 U 上任一属性集合是定义在 U 上的子集组成的 σ 代数上的一个随机变量，其概率分布可以通过如下方式确定。

定义 8[9] 设 P,Q 在 U 上导出的划分分别为 $X,Y(X=\{X_1,X_2,\cdots,X_n\},Y=\{Y_1,Y_2,\cdots,Y_m\})$，则 P,Q 在 U 上的子集组成的 σ 代数上的概率分布为：

$$[X;p]=\begin{bmatrix}X_1 & X_2 & \cdots & X_n\\ p(X_1) & p(X_2) & \cdots & p(X_n)\end{bmatrix}$$

$$[Y;p]=\begin{bmatrix}Y_1 & Y_2 & \cdots & Y_m\\ p(Y_1) & p(Y_2) & \cdots & p(Y_m)\end{bmatrix}$$

$$p(x_i)=\frac{|x_i|}{|U|},i=1,2,\cdots,n;p(Y_j)=\frac{|Y_j|}{|U|},j=1,2,\cdots,m;$$

定义 9[9] 知识(属性集合)P 的熵 $H(P)$定义为：

$$H(P)=-\sum_{i=1}^{n}p(x_i)\log p(x_i)$$

定义 10[9] 知识(属性集合)$Q(U\mid \mathrm{IND}(Q)=\{Y_1,Y_2,\cdots,Y_m\})$相对于知识(属性集合)$P(U\mid \mathrm{IND}(P)=\{X_1,X_2,\cdots,X_n\}$的条件熵 $H(Q\mid P)$定义为：

$$H(Q|P)=-\sum_{i=1}^{n}p(x_i)\sum_{j=1}^{m}p(Y_j|X_i)\log p(Y_j|X_i)$$

$$p(Y_j|X_I)=\frac{|Y_j\cap X_i|}{|X_i|},i=1,2,\cdots,n,j=1,2,\cdots,m$$

从信息熵的角度对决策表进行约减，就是求约减后的条件属性集 B 相对于决策属性集 D 的条件熵 $H(D|B)$，并将其与约减之前的条件熵 $H(D|C)$进行比较，因此，有如下定义：

定义 11[6] 若对于 $B\subseteq C$ 相对于决策属性 D 有 $H(D|B)=H(D|C)$且对于 B 中任意属性 b 都有 $H(D|B)\neq H(D|B-\{b\})$，则称 B 是 C 信息熵观点下的属性约简，简称为信息熵约简。

与代数方法遇到的困难相同，直接求 $H(D|B)=H(D|C)$的集合仍非易事。对此，王国胤博士也给出了类似于代数方法的属性的相关重要度，并提出了两种实现算法。

设 $S(U,A,V,f)$是一个决策表系统，其中 $A=C\cup D,C$ 是条件属性集合，D 是决策属性集合，且 $B\subseteq C$，则对于任意属性 $a\in C-B$ 的条件熵重要度 $\mathrm{SGF}(a,B,D)$[9]定义为：

$$\mathrm{SGF}(n,A,D)=H(D|A)-H(D\mid A\cup\{a\})$$

下面给出两个基于条件熵的知识约简算法——CEBARKCC 算法和 CEBARKNC 算法。CEBARKCC 算法以决策表核属性集为起点，逐次选择使 $H(D|B\cup\{a\})$最小的非核条件属性 a 添加到核属性集中，直到满足终止条件 $H(D\mid B)=H(D\mid C)$，得到约简结果；而 CEBARKNC 算法以决策属性 D 相对条件属性 a 的条件熵 $H(D|\{a\})$的大小作为条件属性 a 对于决策的相关重要度，$H(D\mid\{a\})$的值越大，属性 a 对于决策的相关重要度越小，算法的起点是初始条件属性集，采用逐步删除属性来达到约简的目的。

算法 2[9] CEBARKCC

输入：一个决策表 $S(U,C\cup D,V,f)$，其中，U 为论域，C,D 分别为条件属性集和决策属性集。

输出：该决策表的一个相对约简 B。

Step1 计算决策表 S 中决策属性集 D 相对条件属性集 C 的条件熵 $H(D\mid C)$

Step2 计算条件属性集 C 中相对决策属性集 D 的核属性集 C_0，并令 $Att=C-C_0$。

Step3 令 $B=C_0$

Step3.1 如果 $|B|\ !=0$，则计算条件熵 $H(D|B)$，转 Step3.4；

Step3.2 对每个属性 $a_i\in Att$，计算决策属性集 D 相对条件属性集 $B\cup\{a_i\}$的条件熵 $H(D\mid B\cup\{a_i\})$；

Step3.3 选择使 $H(D\mid\{B\cup\{a_i\})$最小的属性 a_i(若同时有多个属性达到最小值，则从中选取一个与 B 的属性值组合数最少的属性)，$Att=Att-\{a_i\},B:B\cup\{a_i\}$；

Step3.4 若 $H(D\mid B)=H(D\mid C)$则终止，否则转 Step3.2。

算法 3[9] CEBARKNC

输入：一个决策表 $S(U,C\cup D,V,f)$，其中，U 为论域，C,D 分别为条件属性集和决策属性集。

输出：该决策表的一个相对约简 B。

Step1 计算决策表 S 中决策属性 D 相对条件属性 C 的条件熵 $H(D|C)$.

Step2 计算决策属性相对每个条件属性的条件熵 $H(D|\{a_i\})(a_i\in C)$，将 a_i 按 $H(D|\{a_i\}))$降序排列。

Step3 令 $B=C$，按 $H(D|\{a_i\})$递减的顺序对每个 a_i 重复下述操作：

Step3.1 计算决策属性集相对条件属性集 B 在删掉 a_i 后的条件熵 $H(D|B-\{a_i\})$；

Step3.2 如果 $H(D|C)=H(D|B\{a_i\})$，则属性 a_i 应约简，$B=B-\{a_i\}$；否则，属性 a_i 不能被约简，B 不变。

这两种算法都是从信息论的角度出发，引用了知识熵的概念，但它们的理论出发点不同。CEBARKCC 算法和 CEBARKNC 算法虽然都是建立在决策属性集相对条件属性集的条件熵的基础之上的，但 CEBARKCC 算法的启发式信息为 $H(D|B\cup\{a\})$，终止条件 $H(D|B)=H(D|C)$。

CEBARKCC 算法是先计算决策表的核，然后再对非核属性进行约简；而 CEBARKNC 算法不计算核，直接在原有的条件属性集上进行约简。从时间复杂度上分析，$T_{\mathrm{CEBARKCC}}=O(mn^2+O(n^2)$，$T_{\mathrm{CEBARKNC}}=O(m^2n)+O(mn^2)$($m$ 为决策表实例的个数，n 为决策表属性的个数)。

4.2.3 基于区分矩阵方法的约减

前面的两种方法是从粗糙集的基本概念出发，从本质上来说，前两种方法可以看作是互为补充的。除了前两种方法，近些年还有很多人开始对基于区分矩阵的约减方法进行研究，发表了大量的论文，并提出了很多的算法。典型的算法有 HU 的区分矩阵算法[10]、叶东毅的改进区分矩阵算法[11]、陈贞的 AM-RASR 启发式属性约简算法[12]以及程京的对 AM-RASR 的改进算法[3]。

基于代数方法与基于信息熵方法的共同点为，都需要遍历决策表内的每一个属性，之后通过数学运算求出核属性集，从而实现对决策表的约减。这么做的不便之处在于，求解核属性不够直接，在算法实现时需要录入复杂的计算公式。为此，HU 等人引入代数中的区分矩阵概念，将求核属性的过程转化为求一个改进了的区分矩阵的过程。这么做的好处在于，利用二维矩阵可以直接表示决策表内属性间的不可分辨关系，从而清晰、直观地求出决策信息系统的核属性。

HU 提出的改进的区分矩阵 $M=\{m_{ij}\}$[10]为：

$$m_{ij}=\begin{cases}\{a\in C:f(x_i,a)\neq f(x_j,a)\}, & \text{当 } f(x_i,d)\neq f(x_j,d)\\ \varnothing, & \text{其他情况}\end{cases}$$

并给出如下结论：当且仅当某个 m 为单个属性时，该属性属于核 $Core(C)$。

根据 HU 的算法，求出决策表的区分矩阵，然后求出矩阵中仅为单个属性的项，此项即为属性集的核。而叶东毅等人指出该算法存在着缺陷，并在文献[13]中通过反例加以证明。

叶东毅等人认为，HU 的算法对不一致决策表的约减并不适用，并提出改进的差别矩阵 $M=\{m_{ij}\}$[11]。

$$m_{ij}=\begin{cases}\{a\in C:f(x_i,a)\neq f(x_j,a)\},\text{当 } f(x_i,D)\neq f(x_j,D),\\ \quad \text{且 } \min\{d(x_i),d(x_j)\}=1 \text{ 时}\\ \varnothing, \quad \text{其他情况}\end{cases}$$

式中，$d(x_i)$表示 U 中所有与 x_i 在关系 $\mathrm{IND}(C)$下是等价的元素相应的决策属性值构成的集合的基数。并证明了由此差别矩阵得到的单属性元素即为核。

与 HU 提出的区分矩阵相比，叶东毅给出的区分矩阵多了一个标记函数 $d(x_i)$，由其函数定义可得出一个重要的性质：当 $d(x_i)=1$ 时，表示[x_i] C 中的元素同属于划分$\{Y_1, Y_2, \cdots, Y_k\}$中的某一个子集，而当 $d(x_i)>1$ 时，表示[x_i]C 中的元素不属于划分$\{Y_1, Y_2, \cdots, Y_k\}$中的同一个子集，也即相对决策属性而言，数据存在着不一致。因此，加上对 $\min\{d(x_i), d(x_j)\}$的判断后，新的区分矩阵可以在不一致决策表中得以应用。当然，叶东毅等人的研究虽然成功解决了不一致决策表的适用问题，但不足之处在于对存储的要求过高，而且 $\min\{d(x_i), d(x_j)\}$的计算量也较大。

陈贞等人在文献[12]中提出了 AM-RASR 算法，该算法利用属性的重要度作为启发式去求约简，避免了之前的算法要求区分函数的最小析取范式（即最小属性集）的复杂运算。算法在求出决策表对应的区分矩阵（不是 HU 的区分矩阵）之后，遍历条件属性，求出条件属性中权值最大的一个，权值是决策表对象总数除以每一个差别矩阵中包含该属性的元素含有的条件属性个数的累加，权值最大的属性即为核属性。然后将区分矩阵中包含该属性的元素清空，重新遍历，进行新一轮的挑选，直至区分矩阵为空。算法描述如下：

算法 4[12] AM-RASR

输入：一个决策表 $S=(U, C\cup D, V, f)$

输出：该决策表的一个相对约简

Step1 求决策表相对应的区分矩阵；

Step2 求每个属性 a_k 的权值 $\omega(a_k)$，基数 Card(a_k)；其中：$\omega(a_k)$是决策表对象总数除以每一个差别矩阵中包含 a_k 的元素含有的条件属性个数的累加，Card(a_k)是差别矩阵中包含 a_k 元素的个数。$R\leftarrow\emptyset, W\leftarrow 0$

Step3 For $k=1$ to n — /* n 为属性个数 */
If $\omega(a_k) > W$
$W\leftarrow\omega(a_k)$
If $\omega(a_k)=W$ and Card(a_k)较大
$W\leftarrow\omega(a_k)$；

Step4 $R\leftarrow R\cup a_k$
将区分矩阵中包含属性 a_k 的元素清空；

Step5 如果区分矩阵不为空，重新计算余下的每个 $\omega(a_k)$，转向 Step3。

AM-RASR 算法只需对决策表扫描一次，再通过差别属性的加权频率来求得约简，简化了计算。但是该算法仍需存储整个差别矩阵（不考虑矩阵的压缩存储），没有考虑到决策表中重要的核的概念，也没有考虑到不一致决策表中的冲突对象。

基于此，程京等人把叶东毅的解决不一致决策表的思想用到 AM-RSAR 算法中，但是与叶东毅算法不同，并没有去求 $\min\{d(x_i), d(x_j)\}$。采用边计算边修正的方法，用 3 个容器分别存储区分矩阵中的单属性元素、多属性元素和冲突实例集合，前两个容器的并集就是不可辨识属性的集合。在比较决策表中的实例时，需要跳过决策条件相同的实例。如果比较的两个实例存在冲突，则将其放到冲突实例集合中；如果比较结果为单属性，则存入单属性集合中；如果比较结果是多属性，则存入多属性集。在比较过程中，需要注意的是，首先应判断比较的两个实例是否在冲突集中。算法描述如下：

算法 5[3] AM-RASR-A

输入：一个决策表 $S=(U, C\cup D, V, f)$

输出：一个较优约简

Step1 用算法 1 求得单属性集 S 和多属性集 D（过程参见文献[3]）；

Step2 通过将 S 中每个元素与 D 中每个元素做与运算并根据结果来判断 D 中的元素是否包含了 S 中的核属性，将包含了核属性的元素从 D 中移除；

Step3 while $D\neq\emptyset$；

Step4 求出 D 中每个条件属性 a_k 的权值 $\omega(a_k)$、基数 Card(a_k)；

Step5 计算 $\omega(a_k)$最大的属性，若 $\omega(a_k)$最大的有多个，选 Card(a_k)最大的；

Step6 将 Step5 获得的权重最大的属性加入到 S 中，再次通过运算判断 D 中是否包含了该属性的元素，若包含，则将其从 D 中移除；

Step7 endwhile；

Step8 输出 S。

4.2.4 约减算法比较

基于代数方法、基于信息熵方法以及基于区分矩阵方法这 3 种约减方法，从原理上来说都是基于粗糙集的基本理论。代数方法从决策表约减的本质出发，借用粗糙集中正域的概念，对决策表中各个条件属性的冗余性进行判定，从而实现决策表的约减。信息熵方法则是利用信息系统中“条件熵”这一度量衡，将判断决策表属性的冗余性改为直接求属性的条件熵，并运用相应算法进行约减。基于区分矩阵的方法，则是将决策表转换为一个数学矩阵，通过求区分集来求出决策表的核属性。

信息熵论观点和代数观点在一些条件下存在等价关系，而在另外的条件下又呈现出不等价的关系（如包含关系等），具体有[9]：

- 对于一般信息表，约减计算的代数定义和信息定义是等价的；
- 对于一致决策表，约减计算的代数定义和信息定义也是等价的；
- 对于普通（可能含有矛盾、冲突）的决策表，属性相关重要度概念和知识约简在这两种观点下是不等价的。一个决策表在代数观点下的约简，不一定能够保证约简之后的信息熵不发生变化；反之，一个决策表在信息论观点下的约简，同样是代数观点下的约简。也就是说，知识约简的信息论观点包含了其代数观点。

从算法效率上考虑，基于代数的属性约减的复杂之处在于求出一个最小的集合 B，使得 $POS_B(D)=POS_C(D)$，虽然可以转化为求单个属性的重要度 SGF，来判断每一个属性对于当前属性集 B 是否是冗余的，但仍需要耗费 $O(m^2n^2)$的时间复杂度；基于信息熵的属性约减有两种思路，根据是否先计算核属性，有 CREBARKCC 和 CEBARKNC 两种实现算法，复杂度 $T_{CEBARKCC}=O(mn^2+O(n^2)$，$T_{CEBARKNC}=O(m^2n)+O(mn^2)$；而基于区分矩阵的属性约减算法（如本文的算法 4、算法 5）时间的耗费在于构建区分矩阵上，时间复杂度为 $O(m^2n^2)$，如果还要加上对决策表的一致性判断，则还需耗费一些存储空间。

5 研究关注点

通过以上的比较，对决策表的属性约减的目的及原理有了清晰的了解。无论是基于代数方法、信息熵方法，还是基于区分矩阵的方法，其根本目标就是为了找出一个决策信息系统的核心属性。在进行属性约减时，还应关注以下几个方面的问题。

1. 决策表的构建问题

决策表的构建问题，就是如何将领域问题抽象成决策表表示。在抽象的过程中，存在着两个难点。第一，如何将领域的知识精确地用一个决策表展现出来。因为领域的知识往往是复杂的，且关联性强，有时无法用一个独立的决策表表达，比如前面提到的表 2 的例子中函数 CREATEPROCESS 的输入参数与返回值构成的决策表。如果这时，要求返回值不仅跟输入参数有关，还需跟中断开关、当前进程数以及别的未知的因素相关。多了这些外部的因素之后，我们又如何改造之前的决策表？第二，在抽象的过程中，如何对条件属性进行预处理，提前删去显而易见的冗余属性。如果能在决策表的构建过程中，提前判断出决策表属性的冗余，将会大大提高约减的效率。

2. 约减算法的正确性

这里提到的“正确性”有两层含义：算法对于不一致决策表，仍能约减出正确的结果；另一方面，算法的约减结果中不存在冗余属性。当前的研究，通常会对不一致决策表的约减算法加以重视，如叶东毅对 HU 的算法改进。但是，对约减结果的完备性证明，往往却做得不够多。如文献[14]提到的算法，其约减的时间复杂度仅为 $O(n^2)$，但无法保证约减结果的完备性。

3. 约减算法的效率问题

在保证约减算法正确性的前提下，需兼顾算法的效率，体现在时间及空间上。如基于区分矩阵构造的时间复杂度为 $O(m^2n^2)$，叶东毅的算法在此基础上，还需记录 $\min\{d(x_i), d(x_j)\}$，耗费了额外的存储空间。那么能不能通过合适的算法，将时间复杂度做进一步提升？文献[15]中提到的算法，成功地将时间复杂度提升到了 $O(m^2n\log n)$，但只适用于决策条件为一个的情况。

4. 约减算法的应用规模

决策表的属性约减算法的时间复杂度，取决于决策表实例和属性的规模。随着决策表实例和属性的不断扩大，对时间的耗费会显著提升。决策表在多大规模以内，属性约减可以通过算法有效解决，这是我们需要关注的问题，也即算法的适用规模问题。另一方面，如何对大规模决策表进行属性约减？文献[16]提出用分解决策表的方法，将大规模决策表分解成一个个的子决策表。这种方法值得我们去做进一步研究。

5. 约减算法的适用对象

决策表的约减算法可以分为 3 类，那么每一类算法的适用对象有什么限制？这也是在研究决策表约减时应注意的问题。事实上，由于这 3 类算法的出发点不同，约减的结果也会有所出入，这一点，在对不一致决策表的约减时尤为明显。文献[17]中，用实际的例子分析了这 3 类方法约减的本质差异，但仍未对这些方法的适用对象进行深入的探讨。

6. 决策表约减算法在其他软件开发技术中的应用

当前对决策表约减的研究，大多停留在算法等学术讨论上，并未深入分析其实际应用价值。近年来，决策表技术在软件工程中的应用越来越多，尤其是基于决策表的软件测试已较为成熟。然而，在利用决策表生成测试用例时，如果能利用这些算法进行属性约减，势必将大大减少软件测试用例的数量，提高测试的效率。

结束语 本文在对粗糙集理论中关于决策信息系统的概念进行简介之后，分析了决策表面临的问题，指出了决策表冲突检测和属性约减的必要性。重点介绍了决策表属性约减的 3 种方法——基于代数的方法、基于信息熵的方法和基于区分矩阵的方法，分别列出了常用的实现算法，并从本质联系、效率上对这 3 种方法进行了分析、比较。

最后，从决策表的构建、正确性、效率、应用规模等 6 个方面总结了决策表属性约减算法还应关注的热点问题。关于决策表的属性约减算法，可以从提高基于区分矩阵的算法效率，以及如何约减大规模决策表两个方面进行深入的研究。

参 考 文 献

[1] 王国胤. Rough 集理论与知识获取[M]. 西安：西安交通大学出版社，2001：20

[2] Pawlak Z. Rough sets[J]. International Journal of Information and Computer Science，1982，11(5)：341-356

[3] 程京，朱婧，张帆. 一个基于差别矩阵的属性约简改进算法[J]. 湖南大学学报，2009，36(4)：85-88

[4] Suwa M，Scott C，Shortliffe E H. An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System [J]. AI Magazine，1982，3(4)：16

[5] Zhao Shen-sheng，Shen S N T. The consistency problem of knowledge bases[Z]. IEA/AIE，1990

[6] 曾凡智，卢炎生，黄国顺. 信息熵约简与代数约简差异性的数学本质分析[J]. 中山大学学报：自然科学版，2009，3：28-32

[7] Wang Guo-yin. Rough Reduction in Algebra View and Information View [J]. International Journal of Intelligent Systems，2003，18(6)：679-688

[8] 黄国顺. 基于数据库系统的决策表核与属性约简算法[J]. 计算机应用，2008，28(5)

[9] 王国胤，于洪，杨大春. 基于条件信息熵的决策表约简[J]. 计算机学报，2002，25(7)：759-766

[10] Hu X H，Cercone N. Learning in relational databases：a rough set approach[J]. Computational Intelligence，1995，11(2)：323-327

[11] 叶东毅，陈昭炯. 一个新的差别矩阵及其求核方法[J]. 电子学报，2002，30(7)：1086-1089

[12] 陈贞，张全伙. 基于模糊粗糙集的属性约减研究[D]. 华侨大学计算机科学与技术学院，2006

[13] 叶东毅. 不相容决策表分解的若干性质[J]. 小型微型计算机系统，2006，27(4)：695-697

[14] 苗夺谦，胡桂荣. 知识约减的一种启发式算法[J]. 计算机研究与发展，1999，36(6)：681-684

[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

[16] 王加阳，刘柳明，罗安. 大型决策表分解方法研究[J]. 计算机科学，2007，34(8)：211-214

[17] Yin Yi-qi，M Duo-qian，W Rui-zhi，et al. Comparative Analysis of Knowledge Reduction Approaches in Inconsistent Decision Table[J]. Journal of Guangxi Normal University：Natural Science Edition，2006