基于数据相容填补的极大相容块构造算法

周石泉 蒙祖强
(广西大学计算机与电子信息学院 南宁 530004)

摘 要 极大相容块技术在不完备信息处理方面具有一定优势,但极大相容块获取本身是个耗时的过程。通过分析不完备信息系统中极大相容块的性质特点,给出极大相容块中不完备数据的相容填补方法;在不完备信息系统中使用相容填补方法,得到一种新的极大相容块构造算法;基于构造算法的特点,引入决策树存储结构对算法进行优化。使用标准的数据集验证了构造算法的有效性。实验结果表明,构造算法在较大规模的数据集上有较好的性能。

关键词 粗糙集,不完备信息系统,极大相容块,数据填补,决策树
中图法分类号 TP18 **文献标识码** A

Algorithm of Constructing Maximal Consistent Block Based on Consistent Data Reinforcement

ZHOU Shi-quan MENG Zu-qiang
(College of Computer and Electronic Information, Guangxi University, Nanning 530004, China)

Abstract Maximal consistent block technique has some advantages in dealing with the incomplete information. But the construction of maximal consistent blocks itself can be a time-consuming process. After the properties of maximal consistent blocks in incomplete information system were analyzed, the consistent data reinforcement method was defined to handle the missing data in maximal consistent blocks. Using the method in incomplete information system, a new algorithm was obtained to construct maximal consistent block. And the decision tree structure was introduced to optimize the algorithm based on the algorithm's characteristics. Algorithm was evaluated on the standard benchmark dataset. The result of the experiments indicates that the algorithm has better performance in large scale dataset.

Keywords Rough set, Incomplete information system, Maximal consistent block, Data reinforcement, Decision tree

1 引言

粗糙集理论[1]是由波兰科学家 Z. Pawlak 于 1982 年提出的一种用于处理模糊和不确定知识的数学工具。经过近 30 年的发展,它已经被广泛应用于数据挖掘和知识发现、模式识别、机器学习等多个领域[2-5]。Pawlak 粗糙集理论只适用于完备的信息系统,而对于不完备信息系统,通常需要先进行数据补齐等数据预处理操作,然后再运用 Pawlak 粗糙集理论来处理[6]。由于数据补齐过程会导致原信息系统在结构上发生变化,容易造成有用信息的丢失,所得到的结果不一定反映原始系统的真实情况,因此有必要针对不完备信息系统进行一些理论和方法的扩充,以直接从不完备信息系统中获取知识[7]。针对不完备信息系统的粗糙集理论方法,已经有了一些重要成果[7-11]。文献[11]提出极大相容块(即极大相容类,Maximal consistent block)技术,并通过对相容关系的分析,认为极大相容块是不完备信息系统中比较理想的基本粒,用极大相容块技术能得到更好的精度,能简化决策系统的区分矩阵规模,并提高约简的求解效率。

极大相容块技术提出以后,在不完备信息系统的知识获取方面不断得到应用和发展,取得了一些成果。如文献[12]针对集值决策信息系统,在极大相容块的基础上提出相对约简概念,并采用了布尔推理技术来获得集值决策信息系统的最优决策规则。文献[13]在极大相容块的基础上提出的不一致不完备决策表近似约简概念,为不一致不完备决策系统中的知识获取提供了理论依据和方法。文献[14]通过定义和计算极大相容块的 3 种近似约简,来获得包含模糊决策的不完备信息系统的最优决策规则。

极大相容块的计算是此类方法中最为耗时和关键的步骤,它是不完备信息中知识获取的重要前提。传统的极大相容块计算方法,不论是根据属性决定的相容关系求极大相容块集合,还是从对象所在的相容块集合中提取极大相容块,都比较复杂,实现起来也较困难。文献[15]提出分层递阶的构造算法,其通过逐个增加用于分解的属性,递阶地求系统在分解属性集下的"不可区分单元集",再经比较判定得到极大相容块,有效地简化了极大相容块的获取过程,且较容易实现。但分层递阶算法在对上一层得到的极大相容块进行相容分解时,需要对块内元素进行较多的比较;元素比较时需要直接访问元素的某一特定属性值,因此元素的所有属性值需要完全存储,占用空间较多;在每一层完成相容分解后,还需要进行相容块"极大性"检验,在数据规模较大时会消耗较多的时间。

到稿日期:2011-10-24 返修日期:2012-02-18 本文受国家自然科学基金项目(61063032),广西教育厅科研基金项目(201012MS010)资助。
周石泉(1975—),男,硕士生,主要研究方向为粗糙集理论、粒度计算,E-mail:skyrabbit@163.com;蒙祖强(1974—),男,博士,教授,主要研究方向为粒度计算、数据挖掘等。

因此，分层递阶算法在时间和空间上的性能无法满足现实中大规模系统应用的需求。

受文献[15]的启发，本文通过分析极大相容块的性质，针对文献[15]在时间和空间性能上存在的问题，提出极大相容块的相容填补构造算法。本文算法的特点是，每一轮运算针对的是单个元素而不是某个属性，不需要直接访问元素的某一特定属性，因此可以使用决策树(Decision tree)存储方式进行优化，合并属性值相同的元素，在减小比较次数的同时极大地减少了空间占用；还避免了检验相容块"极大性"这一耗时较多的过程，从而实现了时间和空间性能较好的极大相容块构造算法。实验结果表明，本文的极大相容块获取算法在数据规模较大的系统中，不但时间性能优于文献[15]的分层递阶算法，而且极大地减少了空间需求，有利于极大相容块技术在现实中海量数据条件下的应用。

2 预备知识

一个信息系统通常表示为四元组：$S=(U,AT,V,f)$，其中，$U=\{x_1,x_2,\cdots,x_n\}$为对象的非空有限集合，称为论域；AT为属性的非空有限集合，称为属性集；$V=\bigcup_{a\in AT}V_a$，V_a为属性a的值域；$f:U\times AT\to V$为一个信息函数，它为每个对象的每个属性赋予一个信息值，即$\forall x\in U,a\in AT$，则有$f(x,a)\in V_a$。通常$S=(U,AT,V,f)$也简记为$S=(U,AT)$，$f(x,a)$简记为$a(x)$。

对一个信息系统$S=(U,AT,V,f)$，如果至少有一个属性$a\in AT$使得V_a包含空值(missing value)，则称S是一个不完备信息系统，否则它是完备的。不完备信息系统中的空值一般用"$*$"表示。

对给定属性子集$P\subseteq AT$，定义相容(相似)关系如下[16]：

$SIM(P)=\{(x,y)\in U\times U\mid \forall a\in P,a(x)=a(y)\vee a(x)=*\vee a(y)=*\}$

不难证明，$SIM(P)$是自反和对称的，但不一定是传递的，因而是一个相容关系，而且$SIM(P)=\bigcap_{a\in P}SIM(\{a\})$。

令$S_P(x)=\{y\mid (x,y)\in SIM(P),y\in U\}$，对$P$而言，$S_P(x)$是与$x$可能不可区分的对象的最大集合。令$U/SIM(P)=\{S_P(x)\mid x\in U\}$，显然$U/SIM(P)$是论域$U$的一个覆盖，$U/SIM(P)$中的任意元素$S_P(x)$称为相容块。

对象子集$X\subseteq U$，如果对任意$x,y\in X$，有$(x,y)\in SIM(P)$，则称X关于P是相容的。如果不存在对象子集$Y\subseteq U$使得$X\subset Y$且Y关于P是相容的，则称X为一个极大相容块[11]。

从极大相容块的定义可以知道，系统的对象子集要成为极大相容块，必须具备两个条件：一是相容性，即对属性集P，子集内部任意两元素不可区分；二是极大性，即不存在关于P相容的超集。

下文中，$C(P)$表示由属性$P\subseteq AT$决定的所有极大相容块所构成的集合，$C_P(x)$表示$C(P)$中包含x的所有极大相容块所构成的集合。

性质1 任意元素$x\in U$，包含x的所有极大相容块的集合构成x所在相容块的覆盖，即$S_P(x)=\bigcup_{X\in C_P(x)}X$。

证明：(1)先证明$\bigcup_{X\in C_P(x)}X\subseteq S_P(x)$。对任意$X\in C_P(x)$，有$x\in X$。任意$y\in X$，根据极大相容块的定义，可知$X$关于$P$是相容的，有$(x,y)\in SIM(P)$。又因为$X\subseteq U$，即$y\in U$，根据相容块$S_P(x)$的定义可知，$y\in S_P(x)$。因此对任意$X\in C_P(x)$，有$X\subseteq S_P(x)$，即$\bigcup_{X\in C_P(x)}X\subseteq S_P(x)$。

(2)再证明$S_P(x)\subseteq\bigcup_{X\in C_P(x)}X$。因为$x\in S_P(x)$，对任意$y\in S_P(x)$，有$(x,y)\in SIM(P)$。令$Y=\{x,y\}$，则$X$关于$P$是相容的。如果存在极大相容块$Z\in C_P(x)$使$Y\subset Z$成立，则$y\in Z$即$y\in\bigcup_{X\in C_P(x)}X$；否则，$Y$为一个极大相容块，可知$y\in\bigcup_{X\in C_P(x)}X$成立。故$S_P(x)\subseteq\bigcup_{X\in C_P(x)}X$成立。

由(1)、(2)可得，$S_P(x)=\bigcup_{X\in C_P(x)}X$。

根据性质1可以推出下列结论。

推论1 对任意元素$y\in S_P(x)$，存在$X\in C_P(x)$，使$y\in X$，即x与任意相容的元素y必定共存于某一极大相容块中。

性质2 对于任意$a\in P$，在任意$X\in C(P)$中，如果存在$x\in X$使得$a(x)\neq *$，则a在极大相容块X中的非空取值唯一。

证明：假设a在X上的取值不唯一，即存在$x_1,x_2\in X$使得$a(x_1)\neq *\wedge a(x_2)\neq *\wedge a(x_1)\neq a(x_2)$，则$(x_1,x_2)\notin SIM(P)$，这与$X\in C(P)$矛盾。因此在任意$X\in C(P)$中，$a$的非空取值唯一。

3 缺失数据的相容填补

对于不完备信息系统，论域中的元素在某属性上的非空取值可能不唯一，即属性取值具有不确定性，因此缺失数据的补齐，只能通过近似算法来实现，这样极易使系统发生变化，造成最终结果的偏差。

对于极大相容块，根据性质2知，论域中属性的非空值唯一，因此元素的缺失属性值可采用块中相容元素的相应非空属性值来进行填补，称这种缺失数据的填补方法为相容填补方法(consistent data reinforcement method)。

定义1 极大相容块X的数据相容填补操作：

对$a\in P$，如果存在$x\in X$使得$a(x)\neq *$，则对所有$y\in X$，令$a(y)=a(x)$。

填补操作后，对任意$x,y\in X$，有$a(x)=a(y)$，所以$(x,y)\in SIM(P)$，即X的相容性保持不变；对任意$y\notin X,x\in X$，必定存在$a\in P$使得$a(x)\neq *\wedge a(y)\neq *\wedge a(x)\neq a(y)$，即$(x,y)\in SIM(P)$，因此$y\cup X$关于$P$不相容，$X$的极大性不变。

可见，极大相容块经过数据相容填补操作，可以在保持相容性和极大性的同时统一块内各元素的属性值，使极大相容块中任意属性具有单一值。

4 基于数据相容填补的极大相容块获取算法

极大相容块的数据填补操作是在极大相容块已经获得的条件下进行的，而本文的目的是获取极大相容块，因此我们尝试将数据填补操作提前，先使用相容填补操作对系统中的元素进行数据填补，再合并成极大相容块。

在不完备系统中应用极大相容块的相容填补操作必须符合相容填补的要求：(1)填补候选元素y与指定元素x共存于一个关于P的极大相容块X，则y必须与x关于P不可区分，即$y\in S_P(x)$；(2)根据推论1，填补时必须将$S_P(x)$中的所有元素作为候选元素，才能保证填补操作的取值完整性，即保证极大相容块集合的完整性。根据以上特点，在不完备系统中可作如下定义。

定义2 不完备系统中，元素x以$y(y\in S_P(x))$为候选

元素的数据相容填补操作：

对所有属性 $a\in P$，当 $a(y)\neq *$ 时，令 $a(x)=a(y)$。

在设计算法时，由于元素数据填补操作的候选元素可能有多个，为避免不同候选元素的填补操作相互影响，要先生成元素的副本，再对副本作补齐操作，最后将副本插入系统元素集合；将系统中属性值完全相同的元素合并形成一个“复合元素”（代表多个元素）作为单个元素处理，以达到减少元素个数、减少比较运算次数的目的。

算法 1 用于求信息系统 $S=(U,AT,V,f)$ 中由属性子集 $P\subseteq AT$ 决定的极大相容块集合 $C(P)$，描述如下。

算法 1 基于相容填补法的极大相容块获取算法

输入：不完备信息系统 S=(U,AT,V,f)；

输出：极大相容块集合 C(P)。

初始化：C(P)=∅；

步骤 1 若 U=∅ 则转步骤 9；

步骤 2 合并 U 中具有相同属性值的元素，生成的复合元素排列位置取最后一个原元素的位置；

步骤 3 在 U 中查找与首元素 x 相容的元素；

步骤 4 如果未查找到与 x 相容的元素，则 x 为极大相容块，将 x 加入 C(P)，从 U 中删除 x，转步骤 1；

步骤 5 针对每一个与 x 相容的元素 y 进行步骤 7 至步骤 8；

步骤 6 从 U 中删除首元素 x，转步骤 1；

步骤 7 生成 x 的副本，以 y 为候选元素，对副本作数据相容填补操作；

步骤 8 将补齐后的副本插入 U 中 y 之后的位置；

步骤 9 输出 C(P)，算法结束。

例 1 表 1 给出了一个关于计算机设备情况的不完备信息系统 $S=(U,AT,V,f)$ 的描述，其中 $U=\{1,2,\cdots,7\}$，$AT=\{CPU, RAM, HD, Monitor, Price\}$，设 $P=\{CPU, RAM, HD\}$。

表 1 计算机设备情况表

PC	CPU	RAM	HD	Monitor	Price
1	*	DDR2	*	CRT	2800
2	INTEL	DDR3	*	*	3500
3	AMD	*	SATA	LCD	3100
4	*	*	IDE	CRT	2500
5	AMD	DDR3	IDE	*	3100
6	*	DDR2	*	LCD	3000
7	INTEL	DDR3	SATA	LCD	4200

下面利用算法 1 求论域中各对象在属性集 P 下的相容块集合，过程如下：

初始系统对象及属性值

PC	CPU	RAM	HD
1	*	DDR2	*
2	INTEL	DDR3	*
3	AMD	*	SATA
4	*	*	IDE
5	AMD	DDR3	IDE
6	*	DDR2	*
7	INTEL	DDR3	SATA

合并相同属性的元素

（第 1 轮步骤 2）

PC	CPU	RAM	HD
2	INTEL	DDR3	*
3	AMD	*	SATA
4	*	*	IDE
5	AMD	DDR3	IDE
1,6	*	DDR2	*
7	INTEL	DDR3	SATA

C(P): { }

对首元素“2”作相容填补操作

（第 1 轮步骤3—步骤8）

PC	CPU	RAM	HD
3	AMD	*	SATA
4	*	*	IDE
2	INTEL	DDR3	IDE
5	AMD	DDR3	IDE
1,6	*	DDR2	*
7	INTEL	DDR3	SATA
2	INTEL	DDR3	SATA

C(P): { }

处理首元素“3”、“4”合并同属性元素

（第 2、3 轮步骤，第 4 轮步骤 2）

PC	CPU	RAM	HD
2,4	INTEL	DDR3	IDE
4,5	AMD	DDR3	IDE
1,6	*	DDR2	*
4	*	DDR2	IDE
3	AMD	DDR2	SATA
2,7	INTEL	DDR3	SATA

C(P): { }

首元素“2,4”无相容元素归入极大相容块集合

（第 4 轮步骤 3—4 步骤）

PC	CPU	RAM	HD
4,5	AMD	DDR3	IDE
1,6	*	DDR2	*
4	*	DDR2	IDE
3	AMD	DDR2	SATA
2,7	INTEL	DDR3	SATA

C(P): {{2,4}}

继续算法过程直至 $U=\varnothing$

（算法结束）

PC	CPU	RAM	HD
$U=\varnothing$			

C(P): {{1,3,6} {1,4,6} {2,4} {4,5} {2,7}}

最后得到表 1 中系统 S 关于属性集 P 的极大相容块集合：$C(P)=\{\{1,3,6\},\{1,4,6\},\{2,4\},\{4,5\},\{2,7\}\}$。

由例 1 可知，算法 1 无需“极大性”验证即可得到极大相容块集合，减少了验证所需的时间。但是算法在处理过程中有生成元素副本的操作，这可能使系统中某一时刻的元素个数比初始时更多，从而增加算法的空间复杂度；新增元素的插入操作需要进行较多的元素移动，元素个数的增加使查找相容元素等操作时的比较次数增多，可能使算法时间复杂度增加。

通过分析可以发现：(1)算法 1 中查找相容元素时需要与论域中所有元素按属性逐个进行比较，但不需要直接访问元素的某个特定属性值；(2)合并同属性值元素时，也需要与所有元素进行比较；(3)新增元素的插入操作是有序的，位置要在候选元素之后；(4)首元素在处理完成后要删除或归入极大相容块集合。如果使用二维表（或二维数组）方式存储元素，不但不能突出可直接访问特定属性值的优势，反而因增删元素时大量的元素移动操作造成算法效率降低。为此，引入决策树存储结构，以改变算法效率不佳的状况。

5 基于决策树存储结构的改进算法

决策树[17]的定义为：(1)决策树由一个根节点、若干叶节点和若干非叶节点构成；(2)根节点对应于学习任务；(3)每个叶节点都包含一个分类名，即包含一个概念；(4)每个非叶节点都包含一个属性测试，对该属性可能取的每一个值用一个分支引出到另一个节点。

本文使用的决策树存储结构有如下规则：(1)根节点对应的是系统元素集合；(2)每个叶节点代表属性值相同的元素的集合；(3)每一个非叶节点代表属性取值；(4)树是有序的：非叶节点按节点值（属性值）排序（升序或降序），* 值节点排在首位；叶节点按集合中元素的名称排序（升序或降序）。

使用决策树存储结构可有效改进算法 1 的效率：(1)查找当前元素的相容元素时，只需要按照当前元素的属性顺序在树中查找，不需要与所有元素进行比较，大大减少了比较次数；(2)不需要进行合并同属性值元素的步骤，新增元素插入时即可完成合并；(3)不需要进行元素排序，因为 * 值排在首位，新增元素插入位置必在相容元素之后（或合并）；(4)插入、

删除元素不需要作元素移动操作。

通常信息系统的元素数据以二维表的形式给出，要使用决策树存储结构，就需要进行转换。算法 2 将信息系统表述成决策树存储结构。

算法 2 不完备信息系统的决策树生成算法

输入：存放信息系统属性值的二维表 a(i,j)；

输出：决策树 tree。

说明：二维表 a(i,j)中，i=1,2,…,max，代表元素序号；j=1,2,…,pmax，代表属性子集 P 中的属性序号；分枝节点按节点值升序排列，* 值为最小值。

初始化：新建子树为空的树顶 tree，i=1；

步骤 1　如果 i>max，转步骤 11；

步骤 2　t=tree，j=1；

步骤 3　如果 j>pmax，转步骤 9；

步骤 4　如果 t 没有子节点，则新建节点值为 a(i,j)的子节点；

步骤 5　如果 a(i,j)=* 且 t 的第一个子节点值不为 *，则新建节点值为 * 的子节点，并插到第一子节点前；

步骤 6　将 a(i,j)与 t 的子节点值逐个进行比较，直到 a(i,j)的值小于或等于某一子节点 s 的值；

步骤 7　如果 a(i,j)等于子节点 s 的值，则 t=s，j=j+1，转步骤 3；

步骤 8　新建节点值等于 a(i,j)的子节点 n 插入到 s 前，t=n，j=j+1，转步骤 3；

步骤 9　新建值为 i 的叶节点，插入 t 的子节点集合；

步骤 10　i=i+1，转步骤 1；

步骤 11　输出决策树 tree，算法结束。

在得到信息系统对应的决策树后，结合相容填补方法，就可以得到系统的极大相容块集合。这样，算法 1 的改进版本就可以表述如下(称为算法 3)。

算法 3 基于决策树结构和相容填补法的极大相容块构造算法

输入：决策树 tree；

输出：叶节点为极大相容块的决策树 tree。

步骤 1　对 tree 进行深度优先遍历，访问每个叶节点 leaf 时执行步骤 2 至步骤 3；

步骤 2　在 leaf 右侧的元素中查找与 leaf 相容的叶节点；

步骤 3　如果查找到与 leaf 相容的叶节点，则根据查找到的每个相容的叶节点 c-leaf 执行步骤 4 至步骤 6，最后删除叶节点 leaf；

步骤 4　令属性值序列 path 等于 leaf 的属性值序列；

步骤 5　将 path 按 c-leaf 的属性值序列进行相容填补；

步骤 6　复制叶节点 leaf，按属性值序列 path 插入 tree；

步骤 7　算法完成，输出决策树 tree。

下面利用算法 2 和算法 3，求例 1 论域中各对象在属性集 P 下的相容块集合，过程如下：

(1)表 1 中的不完备系统经算法 2 转换后得到的决策树结构如图 1 所示。

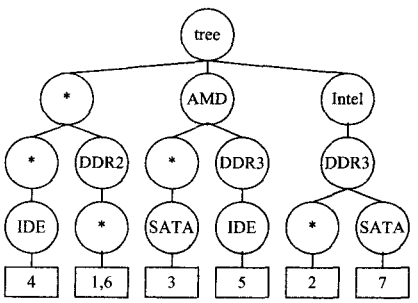

图 1　表 1 经算法 2 转换得到的决策树结构

(2)使用算法 3 对图 1 所示的决策树求极大相容块集合。

(a)对决策树作深度优先遍历，首先访问第 1 叶节点“4”，根据查找到的与“4”相容的叶节点“1,6”、“5”、“2”，执行步骤 4—步骤 6，最后删除原节点“4”，得到的决策树结构如图 2 所示。

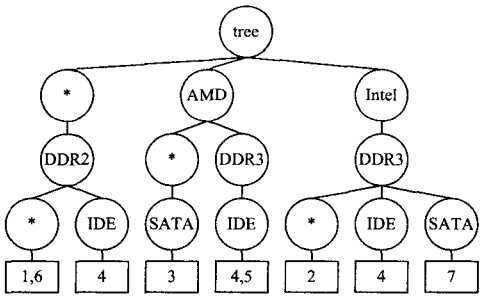

图 2　图 1 中的决策树处理叶节点“4”所得结果

(b)访问下一叶节点“1,6”，根据查找到的与“1,6”相容的叶节点“4”、“3”，执行步骤 4—步骤 6，最后删除原节点“1,6”，得到的决策树结构如图 3 所示。

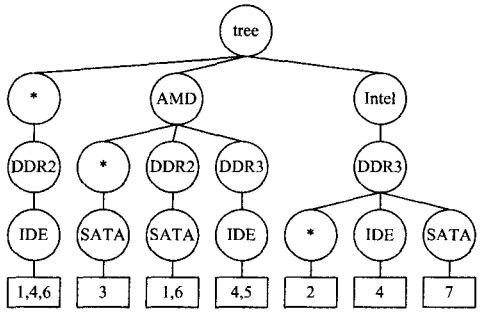

图 3　图 2 中的决策树处理叶节点“1,6”所得结果

(c)图 3 中叶节点“1,4,6”在右侧元素中未查找到相容叶节点，因此无操作。继续访问下一叶节点“3”，根据查找到的与“3”相容的叶节点“1,6”，执行步骤 4—步骤 6，最后删除原节点“3”，得到的决策树结构如图 4 所示。

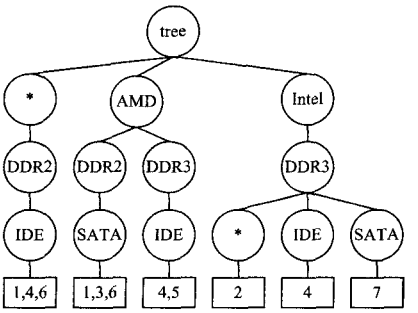

图 4　图 3 中的决策树处理叶节点“3”所得结果

(d)继续访问决策树剩余叶节点并进行相应操作，直到决策树遍历完成，得到的决策树结构如图 5 所示。

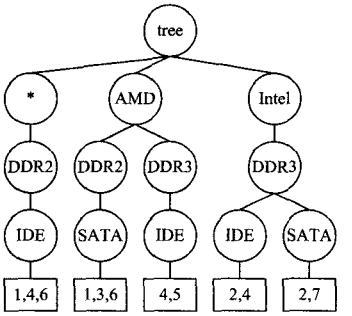

图 5　算法 3 处理完成得到的决策树结构

如图5所示，决策树的叶节点均为极大相容块。最后得到表1中系统 S 关于属性集 P 的极大相容块集合 $C(P)=\{\{1,3,6\},\{1,4,6\},\{2,4\},\{4,5\},\{2,7\}\}$。

6 实验分析

为测试本文算法(算法2和算法3)的实际效果，以UCI数据库中mushroom数据集和thyroid0387数据集为例，对文献[15]中的分层递阶算法和本文的算法进行比较。由于两数据集的规模较小，不能达到大规模数据测试的要求，因此应根据数据集中各属性取值和取值概率，通过随机生成数据的方法，对原数据集进行扩充，以生成数据规模较大的数据集。实验中使用的计算机硬件配置为i5-650CPU、2G内存，编程语言为JAVA，运行环境为eclipse 3.5。

实验中，分层递阶算法和本文算法在两数据集上的运算消耗时间如图6、图7所示，其中X轴表示数据规模(即元素数)，Y轴表示运算时间。两算法处理相应数据规模元素的最小内存需求如图8、图9所示，其中X轴表示数据规模，Y轴表示算法运行所需的最小内存量。

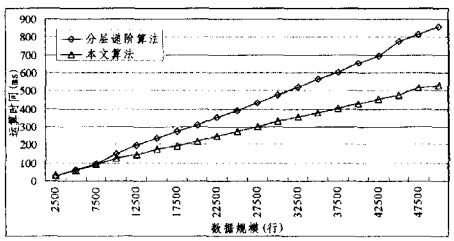

图6 算法在mushroom数据集上的运算时间比较

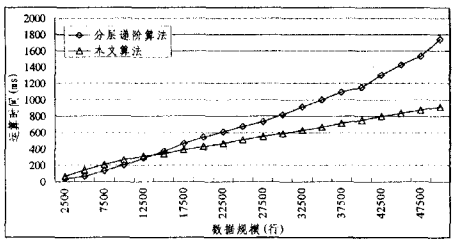

图7 算法在thyroid0387数据集上的运算时间比较

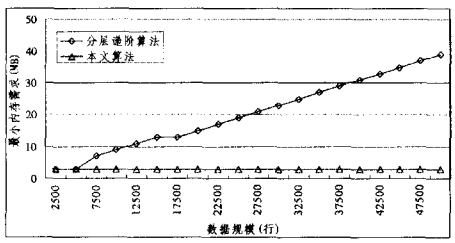

图8 算法在mushroom数据集上的最小内存需求比较

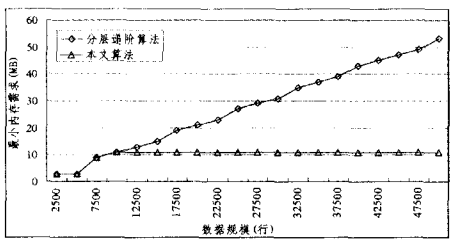

图9 算法在thyroid0387数据集上的最小内存需求比较

从实验结果可以看出，当数据规模较小时，两算法的空间占用相近，分层递阶算法在运算时间上有一定优势；当数据规模增大时，分层递阶算法的空间占用直线上升，运算时间增加也较多，而本文算法的空间占用增长较少，且运算时间也渐优于分层递阶算法。因此，本文算法在数据规模较大的情况下，具有比分层递阶算法更好的空间和时间性能。

在实际应用中，不完备信息系统的知识获取是较复杂的过程，计算极大相容块只是应用极大相容块技术的一个前提步骤，如果计算占用的内存空间过多，就可能影响其它程序的正常运行，因此内存分配会有所限制。大型系统(如销售、人口、网页等方面)的数据规模比较大，可能达到百万、千万甚至以亿计，如果运算需要的内存超过限制，无法将数据全部调入内存，则与外部存储进行的数据交换会增多，从而极大降低效率。由此可见，对于大型系统，空间占用是衡量算法效率的一个重要方面，本文算法有利于极大相容块技术的实际应用。

结束语 极大相容块技术为不完备信息系统知识获取提供了有效方法，而获得属性集决定的极大相容块是这一技术的基础。本文研究了不完备信息系统中极大相容块的性质，给出了不完备信息系统中基于相容填补法的极大相容块构造算法，然后根据算法特点，在存储方式上引入决策树结构进行改进，形成了基于决策树结构的改进算法，有效地减少了运算耗时和空间占用。进一步，选取UCI数据库中的两组非完备数据集进行的实验分析验证了算法在大规模数据条件下的空间和时间性能优势。本文算法不但有助于提高不完备系统中相关算法的性能，而且为有限内存资源条件下处理海量数据提供了切实有效的手段。

参考文献

[1] Pawlak Z. Rough sets[J]. International Journal of Computer and Information Sciences, 1982, 32(11): 341-356

[2] Hu Qing-hua, Liu Jin-fu, Yu Da-ren. Mixed feature selection based on granulation and approximation[J]. Knowledge-Based Systems, 2008, 21(4): 294-304

[3] Qian Yu-hua, Liang Ji-ye, Dang Chuang-yin. Converse approximation and rule extraction from decision tables in rough set theory[J]. Computers & Mathematics with Applications, 2008, 55(8): 1754-1765

[4] Qian Yu-hua, Liang Ji-ye, Li De-yu, et al. Measures for evaluating the decision performance of a decision table in rough set theory[J]. Information Sciences, 2008, 178(1/2): 181-202

[5] 白亮，梁吉业，曹付元. 基于粗糙集的改进K-Modes聚类算法[J]. 计算机科学，2009，36(1)：162-164，176

[6] 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

[7] 王国胤. Rough集理论在不完备信息系统中的扩充[J]. 计算机研究与发展，2002，39(10)：1238-1243

[8] Marzena K. Rough set approach to incomplete information systems[J]. Information Sciences, 1998, 112(1-4): 39-49

[9] Marzena K. Rules in incomplete information systems[J]. Information Sciences, 1999, 113(3/4): 271-292

[10] Jerzy S, Alexis T. On the Extension of Rough Sets under Incomplete Information[C]// Zhong Ning, Andrzej S, Setsuo O, eds. Proc. of the 7th International Workshop on New Directions in

Rough Sets, Data Mining, and Granular-Soft Computing. Berlin: Springer-Verlag, 1999: 73-81

[11] Leung Y, Li De-yu. Maximal consistent block technique for rule acquisition in incomplete information systems[J]. Information Sciences, 2003, 153(1): 85-106

[12] Guan Yan-yong, Wang Hong-kai. Set-valued information systems[J]. Information Sciences, 2006, 176(17): 2507-2525

[13] Qian Yu-hua, Liang Ji-ye, Li De-yu, et al. Approximation reduction in inconsistent incomplete decision tables[J]. Knowledge-Based Systems, 2010, 23(5): 427-433

[14] Yang Fang, Guan Yan-yong, Li Shu-jin, et al. Attributes reduct and decision rules optimization based on maximal tolerance classification in incomplete information systems with fuzzy decisions [J]. Journal of Systems Engineering and Electronics, 2010, 21 (6): 995-999

[15] 梁吉业，王宝丽，钱宇华，等. 一种不完备信息系统中极大相容块的构造算法[J]. 计算机科学，2006，33(11A)：79-82

[16] 张文修，吴伟志，梁吉业，等. 粗糙集理论与方法[M]. 北京：科学出版社，2001

[17] 史忠植. 知识工程[M]. 北京：清华大学出版社，1988

(上接第 182 页)

控网络都包含最重要的 3 项"P","$T*P$"和"$O_{92}(t-1)$"。随后，对基因调控网络的数学表达式各项做敏感度分析，发现"P","$T*P$"两项前的参数对开花日期的影响远远比"T"项前的参数重要，如表 4 所列。图 6 对比了人工开花日期数据和算法重建后的基因调控网络的预测开花日期数据，同时对数据作了线形回归分析。从图中可以看出，人工开花日期数据与算法重建的基因调控网络模型预测的开花日期数据非常相近。

表 3　基因调控网络的数学表示式(参数替代前和参数替代后)

网络	最终开花决定基因的输出	参数替代后最终开花决定基因的输出
人工基因调控网络	$O(t)=C_{18}*C_{32}*C_{80}*C_{92}*T+C_{24}*C_{80}*C_{92}*P+C_{18}*C_{24}*C_{54}*C_{92}*T*P+O_{92}(t-1)$	$O(t)=5.35\times10^{-04}*T+4\times10^{-04}*P+6.2\times10^{-05}*T*P+O_{92}(t-1)$
算法重建后的基因调控网络	$O(t)=C_{18}*C_{32}*C_{54}*C_{92}*P+(C_{18}*C_{24}*C_{54}*C_{92}+C_{18}*C_{24}*C_{92})*T*P+O_{92}(t-1)$	$O(t)=3.5\times10^{-05}*P+1.4\times10^{-04}*T*P+O_{92}(t-1)$

表 4　对表 3 中数学公式各项的敏感度分析

每项的相关敏感度	预测开花日期的数学表达式	T	P	T*P
人工基因调控网络模型	$O(t)=5.35\times10^{-04}*T+4\times10^{-04}*P+6.2\times10^{-05}*T*P+O_{92}(t-1)$	1.97	7.16	8.6
算法重建的基因调控网络模型	$O(t)=3.5\times10^{-05}*P+1.4\times10^{-04}*T*P+O_{92}(t-1)$	—	7.35	6.34

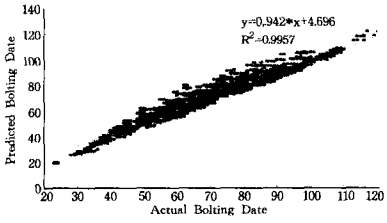

图 6　人工开花日期数据 vs 重建基因调控网络预测开花日期数据

结束语　从实验数据中自动地生成基因网络结构，是目前生物信息领域非常受关注的研究方向。在本文中，我们提出了一种能够从实验数据中直接推导出网络结构和相关参数的算法。算法运用了笛卡尔遗传规划和线性递减惯性权重粒子群优化算法。我们将其应用于拟南芥开花调控系统的模型重建问题。实验结果表明了算法的有效性。在今后的研究工作中，我们将进一步改进防止陷入局部最优的算法，并在更多真实的实验数据上对算法进行应用分析。

参 考 文 献

[1] Hanks R J, Ritchie J T. Modeling plant and soil systems[M]. Agronomy Monograph, 1991: 545

[2] Welch S M, Roe J L, Dong Z. A genetic neural network model of flowering time control in Arabidopsis thaliana[J]. Agronomy Journal, 2003, 95(1): 71-81

[3] Welch S M, Roe J L, Das S, et al. Merging genomic control networks and Soil-Plant-Atmosphere-Contium (SPAC) models[J]. Agricultural Systems, 2003, 86(3): 243-274

[4] Cooper M, Chapman S C, Podlich D W, et al. The GP problem: quantifying gene-to-phenotype relationships[J]. Silico Biology, 2002, 2(2): 151-164

[5] Bernardo D, Gardner T S, Collias J J, et al. Robust Identification of Large Genetic Networks[J]. Pacific Symposium on Biocomputing, 2004, 9: 486-497

[6] Lähdesmäki H, Shmulevich I, Yli-Harja O. On Learning Gene Regulatory Networks under the Boolean Network Model[J]. Machine Learning, 2003, 52(1/2): 147-167

[7] 刘昱昊，刘桂霞，苏兰莹，等. 边排序贝叶斯网络结构学习算法应用于基因调控网络构建[J]. 吉林大学学报：理学版，2010，48 (4)：624-630

[8] 葛玲玲，王浩和，姚宏亮. 基于改进 SEM 算法的基因调控网络构建方法[J]. 计算机应用研究，2010，27(2)：450-258

[9] Chen X W, Gopalakrishna A, Wang X K. An effective structure learning method for constructing gene networks[J]. Bioinformatics, 2006, 22(11): 1367-1374

[10] Mitra S, Das R, Hayashi Y. Genetic networks and soft computing[J]. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2011, 8(1): 94-107

[11] Baldi P, Hatfield G. DNA microarrays and gene expression[M]. Cambridge UK: Cambridge University Press, 2002

[12] Welch S M, Dong Z, Roe J L, et al. Flowering time control: gene network modeling and the link to quantitative genetics[J]. Australian Journal of Agricultural Research, 2005, 56(9): 919-936

[13] Dong Z. Incorporation of Geneic Information into the Simulation of Flowering Time in Arabidopsis Thaliana[D]. Agronomy Department, Kansas State University, 2003

[14] Shi Y, Eberhart R C. A modified particle swarm optimizer[C]// IEEE Congress on Evolutionary Computation (CEC). Anchorage, AK, 1998: 69-73