第41卷 第6A期
2014年6月

计 算 机 科 学
Computer Science

Vol. 41 No. 6A
June 2014

基于不完备决策表的正区域属性约简的压缩差别矩阵方法

王 婷 徐章艳 陈宇文 岳 明

(广西师范大学计算机科学与信息工程学院 桂林 541004)

摘 要 差别矩阵、二进制差别矩阵方法易懂,易设计,一直以来为广大学者所喜欢。但两方法在运算时会产生大量的重复元素与无用元素(若A是B的子集,则称B是A的无用元素),这些重复、无用元素会占用大量的空间,影响算法的效率。针对以往文献中基于差别矩阵的属性约简算法存储代价高的问题,结合二进制差别矩阵引入二叉树(B_Tree)的设计思想,提出基于压缩存储的属性约简算法。该算法将二进制差别矩阵的属性集存储在二叉树(B_Tree)的相应路径上,通过边存边剪枝(剪枝的思想就是从二叉树上删除那些在同一条路径上的重复、无用属性集)的思想,有效地降低了算法的时空效率。最后通过实例分析验证了新算法的有效性和可行性。

关键词 粗糙集,差别矩阵,二叉树,属性约简

中图法分类号 TP18 **文献标识码** A

Method of Compressed Discernibility Matrix of the Attribute Reduction Algorithm Based on Incompletion Decision Table

WANG Ting XU Zhang-yan CHEN Yu-wen YUE Ming

(School of Computer Science and Information Engineering, Guangxi Normal University, Guilin 541004 China)

Abstract Discernibility matrix and binary discernibility matrix method is easy to understand and design, hich has aroused great concern by many scholar. But the two methods produce a large number of repeated and useless elements (if A is the subset of B, B is the useless element of A) on the fly. These repeated and useless elements occupy a lot of space and will affect the efficiency of the algorithm. Attribute reduction based on discernibility matrix methods exist the high cost of storage problem in previous literatures. This paper propose a attribute reduction algorithm based on compressed storage in the thought of combining the binary discernibility matrix with the binary tree (B-Tree). The algorithm store the binary discernibility matrix attribute sets in the binary tree (B-Tree). The algorithm effectively reduce the time and space efficiency by storage while pruning (pruning is a thought, which delete those repeated and useless attribute sets from binary tree on the same path). Finally the analysis of example proves the feasibility and effectiveness of the new algorithm.

Keywords Rough set, Discernibility-matrix, Binary tree, Attribute Reduction

粗糙集理论作为一种新的分析不完整、不精确、不一致信息系统的有力工具,目前在数据挖掘、机器学习、人工神经网络等领域得到了广泛的应用[1,2]。在粗糙集理论中,属性约简是重要的研究内容之一。属性约简的主要思想是在保持一定分类能力不变的前提下,通过知识约简,导出问题的决策或分类规则。近年来,广大学者提出了很多属性约简的算法,绝大多数是以完备决策表作为研究对象。在实际应用中由于数据的测量误差、对知识获取的限制等各种原因,人们往往面临的是不完备决策表。在海量数据集的信息系统中,由于属性和实例数量巨大,属性约简算法的效率就显得尤为重要。粗糙集理论的约简算法迄今为止尚没有一个公认高效的算法。经研究,每种属性约简模型的属性约简都有差别矩阵方法[3-6],而差别矩阵方法因其简洁、高效备受关注。然而,对一个具有 N 个对象 m 个属性的决策表,其差别矩阵的存储复杂度为 $O(N^2 * m)$ 因此当 N 或 m 较大时,差别矩阵的存储代价较高。实际应用中,往往只要求出某个相对属性约简即可。

为了有效降低差别矩阵的存储代价,文献[7]在简化差别矩阵基础上给出了基于改进的 FP 树的快速属性约简算法;文献[8]在引入浓缩树(C-Tree)后,提出了基于 C-Tree 的属性约简算法,该算法将差别矩阵的各非空元素存储在浓缩树的相应路径上,使得差别矩阵的大量非空元素压缩在浓缩树的相应路径上,从而可有效地降低存储开销,改进属性约简的性能。但与众多属性约简算法一样,文献[9]提出的算法主要是针对静态的信息系统或决策表,不适合不完备信息系统或决策表动态变化的情况。而现实世界是发展变化的,决策表中的对象不断地在动态变化,已得到的属性约简将可能不再

本文受国家自然科学基金(61262004, 61363034, 60963008),广西自然科学基金(2011GXNSFA018163)资助。

王 婷 女,硕士生,主要研究方向为数据挖掘、粗糙集理论及其应用,E-mail:betty0907@sina.cn;徐章艳 男,博士,教授,主要研究方向为数据挖掘、粗糙集理论及应用、模糊集理论及应用;陈宇文 女,讲师,主要研究方向为数据挖掘、粗糙集理论及应用;岳 明 男,硕士生,主要研究方向为数据挖掘、粗糙集理论及其应用。

有效,为此,文献[8]给出了C-Tree的改进算法,有效降低了存储代价。此外,文献[10]在差别矩阵基础上提出了基于C-Tree的属性约简增量式更新算法,该C-Tree算法设计思想还是以FP树为基础;文献[11]利用差别矩阵构造决策树,给人们求属性约简提供了一条新途径。这些设计思想也都是以完备决策表为研究对象的。

差别矩阵方法中,由其定义知:若差别元素A是差别元素B的子集,则称B相对于A是可去的,因此称差别元素B是无用元素,不必要存储差别元素B。基于这些算法的启发,为更大程度减少重复无用元素,本文以不完备决策表为研究对象,提出了一种基于B-Tree的压缩存储差别矩阵的属性约简算法,主要讨论差别矩阵中那些重复、无用元素,通过二叉树对重复、无用元素进行压缩,节省存储空间,降低算法执行效率。理论分析和实验结果表明,本文提出的算法是可行有效的。

1 相关概念

定义1[12] 五元组$S=(U,C,D,V,f)$是一个不完备决策表,其中$U=\{x_1,x_2,\cdots,x_n\}$表示对象的非空有限集合,称为论域;$C=\{c_1,c_2,\cdots,c_n\}$表示条件属性的非空有限集;D表示决策属性的非空有限集且$C\cap D=\emptyset$;$V=\bigcup_{a\in C\cup D}V_a$,$V_a$是属性$a$的值域,$f:U\times C\cup D\to V$是一个信息函数,它为一个对象的每一个属性赋予一个信息值,即$\forall a\in C\cup D,x\in U$,有$f(x,a)\in V_a$。

在五元组中,若至少有一个属性$a\in C$,使得V_a包含空值(用$*$表示),即至少有一个属性$a\in C$,存在一个$x\in C$,使得$f(x,a)=*$,称之为不完备决策表。

定义2[12] 设在一个不完备决策表$S=(U,C,D,V,f)$中,令$B\subseteq C$,定义U上的容差关系$T(B)$为$T(B)=\{(x,y)\in U\times U\mid\forall b\in B,f(x,b)=f(y,b)\vee f(x,b)=*\vee f(y,b)=*\}$。用$T_B(x)$表示在$B$下与$x$具有容差关系的全体对象集$\{y\in U\mid(x,y)\in T(B)\}$。

定义3[13] 设在一个不完备决策表$S=(U,C,D,V,f)$中,$\forall B\subseteq C\cup D$,定义论域$U$的一个划分$U/T_B$:$U/T_B=\{T_B(x)\mid x\in U\}$。

定义4[13] 设在一个不完备决策表$S=(U,C,D,V,f)$中,$X\subseteq U$,$\forall B\subseteq C\cup D$,记$B_-(X)=\{x\in U\mid T_B(x)\subseteq X\}$为$X$关于$B$的下近似集,$B^-(X)=\{x\in U\mid T_B(x)\cap x\neq\emptyset\}$为$X$关于$B$的上近似集。

定义5[5] 设在一个不完备决策表$S=(U,C,D,V,f)$中,$\forall B\subseteq C$,$U/D=\{D_1,D_2,\cdots,D_k\}$表示决策属性集$D$对论域$U$的划分,称$POS_C(D)=\bigcup_{D_i\in U/D}C_-(D_i)$为$C$关于$D$的正区域,简记为$U_{pos}$,$U_{neg}=U-U_{pos}$。

定义6[5] 设在一个不完备决策表$S=(U,C,D,V,f)$中,若$\forall b\in B\subseteq C$,若$POS_B(D)=POS_{B-\{b\}}(D)$,则称$b$为$B$中相对于$D$是可省的(不必要的);否则称$b$为$B$中相对于$D$是不可省的(必要的)。对$\forall B\subseteq C$,若$B$中任一元素相对于$D$都是必要的,则称$B$相对于$D$是独立的。

定义7[5] 设在一个不完备决策表$S=(U,C,D,V,f)$中,若$\forall B\subseteq C$,$POS_B(D)=POS_C(D)$且B相对于D是独立的,则称B是C的相对于D的属性约简。

定义8[5] 设在一个不完备决策表$S=(U,C,D,V,f)$中,$U=U_{pos}\cup U_{neg}$,定义简化的二进制差别矩阵$M=(m(i,j),k)$,其元素定义如下:

$$m((i,j),k)=\begin{cases}1,\ c_k\in C,f(x_i,c_k)\neq f(x_j,c_k)\wedge f(x_i,c_k)\neq *\wedge f(x_j,c_k)\neq *,f(x_i,D)\neq f(x_j,D)\text{且}x_i,x_j\text{在}U_{pos}\text{中};\\ 1,\ c_k\in C,f(x_i,c_k)\neq f(x_j,c_k)\wedge f(x_i,c_k)\neq *\wedge f(x_j,c_k)\neq *,\text{且}x_i,x_j\text{一个在}U_{pos}\text{中,一个在}U_{neg}\text{中};\\ 0,\ \text{否则}\end{cases}$$

其中,$k=1,2,\cdots,r$。

定义9[5] 设$M=(m(i,j),k)$为不完备决策表$S=(U,C,D,V,f)$的简化二进制差别矩阵,$\forall P\subseteq C$,若P满足:(1)P中所有属性对应的各列所构成的M的子阵中,不全为0的行数等于M中不全为0的行数;(2)$\forall B'\subset B$均不满足(1),则称P是C关于D的一个属性约简。

定义10[5] 设$M=(m(i,j),k)$为不完备决策表$S=(U,C,D,V,f)$的简化二进制差别矩阵,对$\forall A\in M$,若$\exists B\in M$使得$B\subseteq A$,则称A为无用差别元素。

定理1[5] 设$M=(m(i,j),k)$为不完备决策表$S=(U,C,D,V,f)$的简化二进制差别矩阵,$\forall P\subseteq C$,若P满足:P中所有属性对应的各列所构成的M的子阵中,不全为0的行数等于M中不全为0的行数,则有$POS_P(D)=POS_C(D)$。

定理2[5] 设$M=(m(i,j),k)$为不完备决策表$S=(U,C,D,V,f)$的简化二进制差别矩阵,$\forall P\subseteq C$,若$POS_P(D)=POS_C(D)$,则P中所有属性对应的各列所构成的M的子阵中,不全为0的行数等于M中不全为0的行数。

定理3[5] 基于正区域的属性约简与基于简化二进制差别矩阵的属性约简定义是等价的。

2 二叉树(B-Tree)构造算法

定义8 二叉树的结点定义为:

lchild	data	parent	count	flag	rchild	rbrother

其中,data存储属性值,count记录能到达该节点的路径所表示的非空元素的数目;flag为0时表示该结点为非叶结点,flag为1时表示该结点为叶结点;lchild存放左孩子,rchild存放右孩子;rbrother存放兄弟结点。

数组$A[i]$为0时表示属性C_i不存在,$A[i]$为1时表示属性C_i存在;$flag[i]$为0时表示属性C_i为非叶结点,$flag[i]$为1时表示属性C_i为叶结点。

定义8 属性表定义为:

attribute-name	freq	next

其中,attribute-name表示决策表中对应的属性值,freq表示该属性值出现的频率,next表示该结点的后继,且都指向树结点。

算法1 B-Tree构造算法

输入:不完备决策表S=(U,C,D,V,f)中,U={$x_1,x_2,\cdots,x_n$},C={$c_1,c_2,\cdots,c_r$};

输出:二叉树B-Tree

1. 由文献[5]算法1求出容差类Tci(a)(a∈U)
2. 由文献[5]算法2求出$U_{pos}=\{y_1,y_2,\cdots,y_s\}$,$U_{neg}=\{z_1,z_2,\cdots,z_t\}$
3. for(i=1;i≤r;i++)

{ AB[i].attribute-name=c_i;

```
    AB[i].freq=0;
    AB[i].next=null;}
4. 创建 B-Tree 的根结点 root
   root.data=null;root.count=0;root.flag=0;
   root→next=root→parent=null;
5. for(i=1;i<s;i++)
     for(j=1;j<s+1;j++)
       if(f(x_i,D)≠f(x_j,D))
       { for(k=1;k<r+1;k++)
         if(f(x_i,c_k)≠f(x_j,c_k) ∧ f(x_i,c_k)≠* ∧ f(x_j,c_k)≠*)
           m((i,j),k)=1;
       else m((i,j),k)=0;}
     for(h=r;h>0;h--)
     { if(m((i,j),h)==1)A[h]=1;flag[h]=1;break;}
     if (h!=0)
       { for(l=h;l>0;l--)
         {A[h]=m((i,j),h);flag[h]=0;}
         Insert-BTree(A[h],flag[h],**root);}
   }
6. for(i=1;i<s;i++)
   for(j=1;j<t+1;j++)
   { for(k=1;k<r+1;k++)
     if(f(x_i,c_k)≠f(x_j,c_k) ∧ f(x_i,c_k)≠* ∧ f(x_j,c_k)≠*)
         m((i,j),k)=1;
       else m((i,j),k)=0;}
     for(h=r;h>0;h--)
     { if(m((i,j),h)==1),A[h]=1;flag[h]=1;break;}
   if (h!=0)
   { for(l=h;l>0;l--)
     {A[h]=m((i,j),h);flag[h]=0;}
     Insert-BTree(A[h],flag[h],**root);}
   }
7. for(i=1;i<=r;i++)
   { s=AB[i]→next;
     while(s!=null)
     { p=s;
       AB[i].freq=AB[i].freq+s.count;
       s=p→next;}
8. 输出 B-Tree。
```

其中，算法 1 引用的子函数如下：

```
Insert-BTree(B[],flag[],B-tree **root)
{ B-tree *p,*q,*r,*s;
  p=r=root;i=1;
  while(flag[i]!=1)
  {
  if(B[i]==1)
  { r=p→rchild;
    if(r!=null)
    { if(r→flag!=1)
      {i++;
      p=r;
      p.count++;
      if(p.data!=Ø)
        {
        AB[i]→next=p;
        p→next=AB[i]→next;
        }
      }
    else
    {q=(B-Tree*)malloc(sizeof(B-Tree));
      q→data=c_i;q→flag=0;
      q→lChild=null;
      q→rChild=null;p=q;
      p.count=1;p→next=null;
      AB[i]→next=p;
      i++;
    }
  if(B[i]==0)
  { r=p→lchild;}
    if(r!=null)
    {
      i++;
      p=r;
    }
    else
    { q=(B-Tree*)malloc(sizeof(B-Tree));
      q→data=Ø;q→flag=0;
      q→lChild=null;
      q→rChild=null;p=q;i++;
    }
  if(p→lChild!=null ∨ p→rChild!=null)
  { 删除 p 结点的左右子树，并且删除与属性表的链接，修改对应属性表的 freq 与 next;}
  else
  { if(p→flag==1)}
    { p=r;p.count--;s=p→parent;
      while(s!=root)
      { p=s;
        if(p.data!=Ø)p.count--;}
  }
  return;
  }
```

Insert-BTree$(A[h], flag[h], *N)$功能为：1. $A[h]$为差别元素，当树上有一个分支是 $A[h]$的子集时，则不需插入 $A[h]$；2. $A[h]$为差别元素，当 $A[h]$为树上一个分支（即一个差别元素）的子集时，则保留 $A[h]$，去掉上述分支；3. 当一个结点的属性不为空集时，则将该结点链接到右兄弟结点。故有如下结论：若差别元素 $|A|>1$，则该树上不存在差别元素 B 是 A 的子集（$|B|>1$）。

算法 1 的时间与空间复杂度分析如下：算法 1 第 1 步的时间复杂度为 $O(|U|)$，第 2 步的时间复杂度为 $O(K|C||U|)$

(其中 $K=\max\{|TC(x_i)|,x_i\in U\}$),第 3 步的时间复杂度为 $O(|C|)$,第 4 步忽略不计,第 5 步、第 6 步时间复杂度为 $O(|C||U_{pos}||U_{pos}|)+O(|C||U_{pos}||U_{neg}|)=O(|C||U_{pos}|\|U\|)$,第 7 步的时间复杂度为 $O(|C|)$,第 8 步忽略不计。所以算法 1 的时间复杂度为 $\max\{O(|C||U_{pos}|\|U\|),O(K|C||U|)\}$(其中 $K=\max\{|TC(x_i)|,x_i\in U\}$)。

3 属性约简算法

算法 2 属性约简算法

输入:B-Tree

输出:不完备决策表的属性约简 reduce(C)

```
1. attribute-eount=0;
   for(i=1;i≤r;i++)
      attribute-eount=attribute-eount+AB[i]. freq;
2. while(attribute-eount != 0)
{
   2.1 for(i=1;i≤r;i++)
       { s=AB[i]→next;q=s;
                 while(q→next != null)
                 {
          count=1;
          p=q→parent;
          while(p != root)
          { if(p. data != Ø)count++;
            p=p→parent;
                    }
          if(count != 1)
               q=q→next;
               if(count==1)
{ reduce(C)=reduce(C)∪{ci};
  while(s→next != null)
  { if(s. flag==1)
     {
     r=s→parent;
     attribute-eount--;
     while(r != root)
        { if(r. data != Ø)
        attribute-eount--;
     r=r→parent;
           }
     }
else
{preorder(s);
           }
                  }
     AB[i]. freq=0;
     AB[i]→next=null;
        }
           }
   }
   2.2 令 max=AB[1]. freq;k=1;
     for(i=2;i≤r;i++)
     { if(AB[i]. freq>max)
       { max=AB[i]. freq;
         k=i;}
     }
     reduce(C)=reduce(C)∪AB[k]. attribute. name;
s=AB[k]→next;
          while(s→next != null)
     { if(s. flag==1)
    {
     r=s→parent;
     attribute-eount--;
     while(r != root)
     {if(r. data != Ø)
          attribute-eount--;
       r=r→parent;
           }
           }
           else
           {preorder(s);
                 }
     AB[k]. freq=0;
     AB[k]→next=null;
   }
3. if(attribute-eount==0)
   输出属性约简 reduce(C);
```

算法 2 中调用的子函数如下所示:

```
voidpreorder(B-tree * s)
{
if(s != null ∧ s. data != Ø)
attribute-eount--;
preorder(s→lchild);
preorder(s→rchild);
}
```

算法 2 的时间与空间复杂度分析如下:算法 2 主要对数进行遍历搜索,二叉树包括根节点在内共有 $|C|+1$ 层,对二叉树每一条路径进行遍历,因重复元素,无用元素在建立二叉树时已覆盖删除,所以遍历二叉树的时间远远小于 $2|C|$,故该时间复杂度为 $O(L|C|)$(其中 L 为二叉树的路径数目)。该算法空间复杂度为 $O(M|C|)$(其中 M 为二叉树中存储的差别元素个数,差别元素不包括重复元素与无用元素)。

4 实例分析

为验证算法的有效性、可行性,以文献[11]中的不完备决策(见表 1)为例进行分析说明。

表 1 不完备决策表

U	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	d
a_1	3	2	1	1	1	0	*	*	0
a_2	2	3	2	0	*	1	2	1	0
a_3	2	3	2	0	1	*	3	1	1
a_4	*	2	*	1	*	2	0	1	1
a_5	*	2	*	1	1	2	0	1	1
a_6	2	3	2	1	3	1	*	1	1
a_7	3	*	*	3	1	0	2	*	0
a_8	*	0	0	*	*	0	2	0	1
a_9	3	2	1	3	1	1	2	1	1
a_{10}	1	*	*	*	1	0	*	0	0
a_{11}	*	2	*	*	1	*	0	1	0
a_{12}	3	2	1	*	*	0	2	3	0

通过算法 1 的第 1 步与第 2 步计算容差类与正域得:

$T_C(a_1)=\{a_1,a_{11},a_{12}\}$;$T_C(a_2)=\{a_2,a_3\}$;

$T_C(a_3)=\{a_3,a_2\}$;$T_C(a_4)=\{a_4,a_5,a_{11}\}$;

$T_C(a_5)=\{a_5,a_{11}\}$;$T_C(a_6)=\{a_6\}$;

$T_C(a_7)=\{a_7,a_{12}\}$;$T_C(a_8)=\{a_8,a_{10}\}$;

$T_C(a_9)=\{a_9\}$;$T_C(a_{10})=\{a_{10},a_8,a_{11}\}$;

$T_C(a_{11})=\{a_{11},a_1,a_4,a_5\}$;

$T_C(a_{12})=\{a_{12},a_1,a_7\}$;

$U/D=\{\{a_1,a_2,a_7,a_{10},a_{11},a_{12}\},\{a_3,a_4,a_5,a_6,a_8,a_9\}\}$。

$U_{pos}=\{a_1,a_6,a_7,a_9,a_{12}\}$

$U_{neg}=\{a_2,a_3,a_4,a_5,a_8,a_{10},a_{11}\}$

创建二叉树前得到的二进制差别矩阵如表 2 所列。

表 2　二进制差别矩阵

m(i,j)	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
m(1,6)	1	1	1	0	1	1	0	0
m(1,9)	0	0	0	1	0	1	0	0
m(6,7)	1	0	0	1	1	1	0	0
m(6,9)	1	1	1	1	1	0	0	0
m(9,12)	0	0	0	0	0	1	0	1
m(1,2)	1	1	1	1	0	1	0	0
m(1,3)	1	1	1	1	0	0	0	0
m(1,4)	0	0	0	0	0	1	0	0
m(1,5)	0	0	0	0	0	1	0	0
m(1,8)	0	1	1	0	0	0	0	0
m(1,10)	1	0	0	0	0	0	0	0
m(1,11)	0	0	0	0	0	0	0	0
m(6,2)	0	0	0	1	0	0	0	0
m(6,3)	0	0	0	1	1	0	0	0
m(6,4)	0	1	0	0	0	1	0	0
m(6,5)	0	1	0	0	1	1	0	0
m(6,8)	0	1	1	0	0	1	0	1
m(6,10)	1	0	0	0	1	1	0	1
m(6,11)	0	1	0	0	1	0	0	0
m(7,2)	1	0	0	1	0	1	1	0
m(7,3)	1	0	0	1	0	0	1	0
m(7,4)	0	0	0	1	0	1	1	0
m(7,5)	0	0	0	1	0	1	1	0
m(7,8)	0	0	0	0	0	0	0	0
m(7,10)	1	0	0	0	0	0	0	0
m(7,11)	0	0	0	0	0	0	1	0
m(9,2)	1	1	1	1	0	0	1	0
m(9,3)	1	1	1	1	0	0	1	0
m(9,4)	0	0	0	1	0	1	1	0
m(9,5)	0	0	0	1	0	1	1	0
m(9,8)	0	1	1	0	0	1	0	1
m(9,10)	1	0	0	0	0	1	0	1
m(9,11)	0	0	0	0	0	0	1	0
m(12,2)	1	1	1	0	0	1	1	1
m(12,3)	1	1	1	0	0	0	1	1
m(12,4)	0	0	0	0	0	1	1	1
m(12,5)	0	0	0	0	0	1	1	1
m(12,8)	0	1	1	0	0	0	0	1
m(12,10)	1	0	0	0	0	0	0	1
m(12,11)	0	0	0	0	0	0	1	1

根据算法 1 的第 3 步到第 7 步创建的二叉树如图 1 所示。

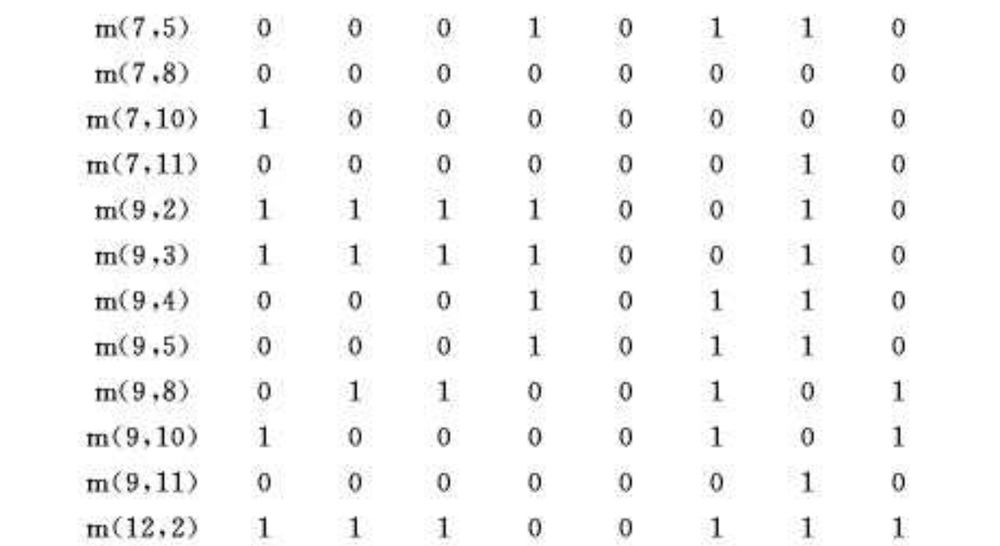

图 1　二叉树

单单分析表 2，重复元素有：$m(1,4)=m(1,5)=\{c_6\}$；

$m(1,10)=m(7,10)=\{c_1\}$；

$m(6,8)=m(9,8)=\{c_2,c_3,c_6,c_8\}$

$m(7,4)=m(7,5)=m(9,4)=m(9,5)=\{c_4,c_6,c_7\}$；

$m(7,11)=m(9,11)=\{c_7\}$；

$m(9,2)=m(9,3)=\{c_1,c_2,c_3,c_4,c_7\}$；

$m(12,4)=m(12,5)=\{c_6,c_7,c_8\}$。

无用元素：

$m(1,6)$、$m(6,7)$、$m(6,9)$、$m(1,2)$、$m(1,3)$、$m(6,10)$、$m(7,2)$、$m(7,3)$、$m(9,2)$、$m(9,3)$、$m(9,10)$、$m(12,2)$、$m(12,3)$、$m(12,10)$都是 $m(1,10)=m(7,10)=\{c_1\}$的无用元素；

$m(1,9)$是 $m(6,2)$的无用元素；

$m(9,12)$、$m(12,4)$是 $m(1,4)$的无用元素；

$m(6,8)$、$m(9,8)$、$m(12,8)$是 $m(1,8)$的无用元素；

$m(6,5)$是 $m(6,11)$的无用元素；

空集：$m(1,11)=m(7,8)=\emptyset$。

综上，生成的二叉树的数据如表 3 所列。

表 3　生成二叉树信息

m(i,j)	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
m(1,4)	0	0	0	0	0	1	0	0
m(1,8)	0	1	1	0	0	0	0	0
m(1,10)	1	0	0	0	0	0	0	0
m(6,2)	0	0	0	1	0	0	0	0
m(6,4)	0	1	0	0	0	1	0	0
m(6,11)	0	1	0	0	1	0	0	0
m(7,11)	0	0	0	0	0	0	1	0

根据算法 2 的第 1 步算出 attribute-eount=11，由算法 2 的 2.1 得到的核属性有 c_7、c_6、c_4、c_1，把 $\{c_7\}$、$\{c_6\}$、$\{c_4\}$、$\{c_1\}$并到 $reduce(C)$中，得到 $reduce(C)=\{c_7,c_6,c_4,c_1\}$，并将属性表相应属性的 freq 改为 0，相应属性的 next 的值改为 null，如图 2 所示。

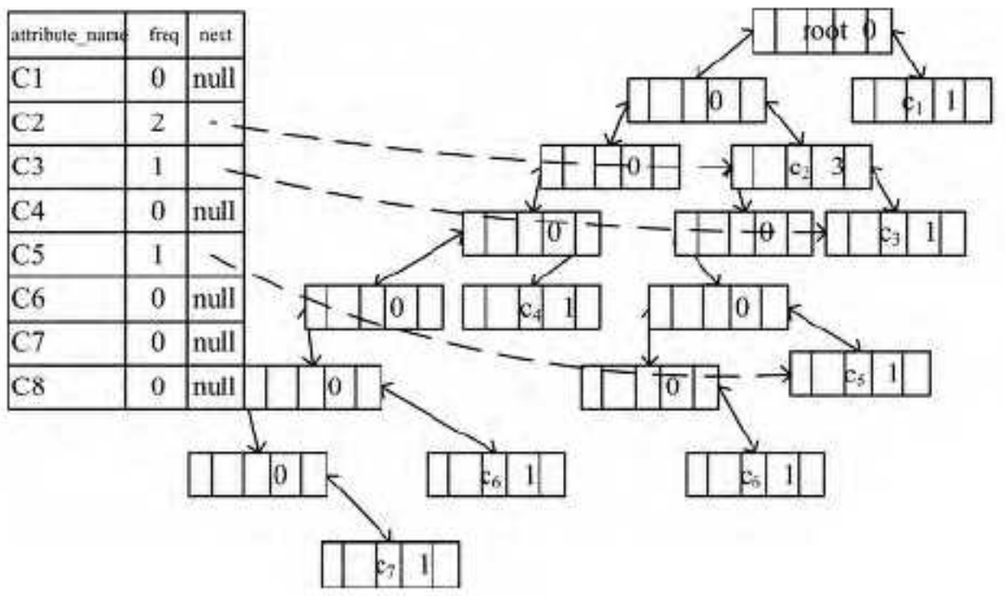

图 2　修改单个属性后二叉树

根据算法 2 的步骤 2 选择属性 freq 值最大的属性值，即为 c_2，由步骤 2 知 attribute-eount=0，由步骤 3 知算法结束，属性约简 $reduce(C)=\{c_7,c_6,c_4,c_1,c_2\}$。

结束语　本文提出了一种基于 B-Tree 的属性约简算法，通过实例分析，主要讨论差别矩阵中那些重复元素、无用元素。通过实例分析，对实例中二进制差别矩阵中有 40 条记录，通过二叉树，将数据按一定规则存放在二叉树中，最后存储在二叉树中仅有 7 条路径，也就是说二叉树为消除重复、无用元素提供了一条新的途径。

参 考 文 献

[1]　Pawlak Z. Rough sets[J]. Int J of Information and Computer

Science,1982,11(5):341-346

[2] Pawlak Z. Rough set approach to multi-attribute decision analysis[J]. European J of Operational Research, 1994,72(3):443-459

[3] 徐章艳,杨炳儒,宋威. 基于简化的二进制差别矩阵的快速属性约简算法[J]. 计算机科学,2006,33(4):155-158

[4] 支天去,苗夺谦. 二进制可辨识矩阵的变换及高效属性约简算法的构造[J]. 计算机科学,2002,29(2):140-142

[5] 舒文豪,徐章艳,等. 一种快速不完备决策表的差别矩阵属性约简算法[J]. 小型微型计算机系统,2011,32(9):103-110

[6] 徐章艳,杨炳儒,宋威,等. 几种不同属性约简的比较[J]. 小型微型计算机系统,2008,29(5):848-853

[7] 黄丽宇,徐章艳,等. 基于改进的 FP 树的快速属性约简算法[J]. 计算机工程与应用,2010,46(35):152-191

[8] Yang Ming, Yang Ping. A novel condensing tree structurefor rough set feature selection[J]. Neurocomputing,2008,71(4-6):1092-1100

[9] Yang Ming, Yang Ping. A novel approach to improving。C-Tree for feature selection[J]. Applied Soft Computing,2010,11(2):1924-1931

[10] 杨明,吕静. 一种基于 C-Tree 的属性约简增量式更新算法[J]. 控制与决策,2012,27(12):1769-1775

[11] 高静,韩智东. 利用差别矩阵构造决策树[J]. 计算机工程与应用,2011,47(33):18-21

[12] Kryszkiewicz M. Rough set approach to incomplete information system[J]. Information Science,1998,112(1):39-49

[13] Kryszkiewicz M. Rules in incomplete information systems[J]. Information Sciences,1999,113(2):271-292

(上接第 365 页)

其中,T_{di} 为完成服务 d_i 的最终期限,$ETC(d_i,t)$ 为服务请求 d_i 在资源 r_j 上的预期执行时间,e_{ij} 为完成服务 d_i 用户愿意支付的服务费用,P_{ij} 为服务 d_i 使用资源 r_j 的价格。

Step 4 TTP 向 B 发送接收消息 m 的请求。

Step 5 B 向 TTP 发送 EOR。

Step 6 TTP 通过两次解密并校验有效性,来向 B 发送 (m,EOO)。

Step 7 同时向 A 发送 EOD。

Step 8 交换信息 m 完毕。

4 仿真实验与性能分析

本文采用 CloudSim 模拟工具进行了仿真实验来验证基于可信第三方的云资源管理模型的有效性和性能。

CloudSim 软件是澳大利亚墨尔本大学的网格实验室和 Gridbus 项目联合开发设计的云计算仿真软件。在 SimJava 离散事件模拟的包基础上开发了函数库,同时采用虚拟化技术将数据中心的资源进行整合,将其虚拟化为资源池,使其支持云计算的安全认证、资源管理和调度模拟。本文以 CloudSim 为基础进行二次开发,搭建基于可信第三方的云资源管理建模的实验环境。云主机上安装 Red Hat 9.0 Linux 版本操作系统,并配置 jdk1.6.0_13 软件。通过将 CloudSim 1.0 beta 版解压到 jdk1.6.0_13 软件安装目录下后,在 ClassPath 中加入路径“C:\CLOUDSIM\jars\cloudsim.jar;C:\CLOUDSIM\jars\gridsim.jar;.\CLOUDSIM\jars\simjava2.jar;”,完成了 CloudSim 的配置。仿真实验的基本步骤如下:初始化 GridSim 函数库、创建数据中心、创建 TTP、创建虚拟机、创建云任务、建模仿真、结果统计。

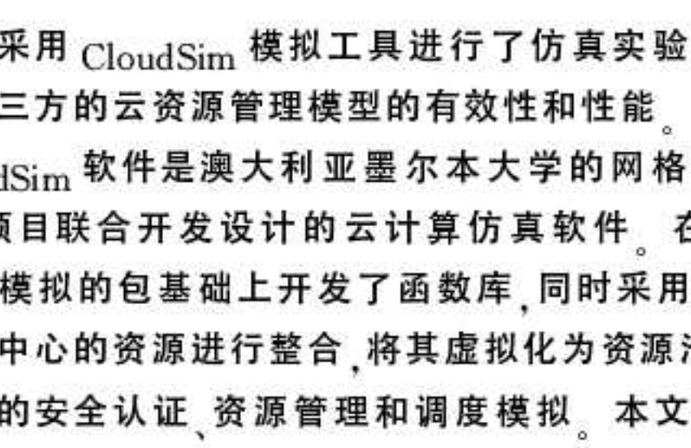

图 5 SJF、FCFS 与 TTP 策略任务成功执行率比较

本文将短任务先执行(Shortest Job First,SJF)策略、先来先服务(First Come First Serve,FCFS)策略,与本文基于 TTP 的策略做任务成功率的比较。由图 5 看出,随着云用户的增多,任务执行成功率有所下降,但是本文策略的成功率要高于 SJF、FCFS 策略。由此得出结论,基于 TTP 策略构建的云环境中,将可信度最高的云节点资源提供给云用户,能够有效提高资源分配的效率、提高可信度。

结束语 信任与安全成为云计算面临的重大威胁,本文将可信第三方引入云安全的解决方案,通过第三方的信誉监督解决了客户的安全负担,第三方的任务是在分布式信息系统中保证特定的安全特性,从而构建实体之间的信任网。改进的基于 TTP 的资源分配算法可有效提高资源分配率、可信度。本文工作只是个起点,下一步研究工作的重点是:模型的细节实现;基于证书的授权问题;进一步完善系统的完整性、保密性、真实性等。

参 考 文 献

[1] 刘鹏. 网格计算与云计算(PPT)[EB/OL]. http://www.chinacloud.cn/download/PPT/GridCloudComputing.ppt

[2] Buyya R, Abramson D, Giddy J, et al. Economic models for resource management and scheduling in grid computing [J]. Concurrency & Computation,2002,14(13-15):1507-1542

[3] 高宏卿,邢颖. 基于经济学的云资源管理模型研究[J]. 计算机工程与设计,2010,31(10):4139-4142

[4] Geimer M, Shende S, Malony A D, et al. A generic and configurable source-code instrumentation component[C] // Allen G, Nabrzyski J, Seidel E, et al., eds. ICCS (2), Vol. 5545 of Lecture Notes in Computer Science. Springer,2009:696-705

[5] International Telecommunication Union. X-509|ISO/IEC 9594-8, The directory: Public-key and attribute certificate frameworks [S]. ITU, X-Series,2001

[6] 卿斯汉. 电子商务协议中的可信第三方角色[J]. 软件学报,2003,14(11):1936-1943

[7] Lekkas D, Gritzalis S, Katsikas S. Quality assured trusted third parties for Deploying secure Internet-based health care applications[J]. International Journal of Medical Informatics,2002,65(2):79-96

[8] 陈冬娥,杨扬,刘丽. 基于效用最优的网格计算资源调度算法[J]. 计算机工程与应用,2006,42(2):191-193