关于Top-*N*最频繁项集挖掘的研究

朱颢东，李红婵

(郑州轻工业学院计算机与通信工程学院 郑州 450002)

【摘要】最频繁项集挖掘决定了文本关联规则挖掘算法的性能，是文本关联规则挖掘中研究的重点和难点。该文分析了当前最频繁项集挖掘方面的不足，改进了传统的倒排表，结合最小支持度阈值动态调整策略，提出了一个新的基于改进的倒排表和集合理论的Top-*N*最频繁项集挖掘算法。同样，给出了几个命题和推论，并把它们用于该文算法以提高性能，实验结果表明，所提算法的规则有效率和时间性能优于NApriori算法和IntvMatrix算法。

关 键 词 关联规则； 倒排表； 频繁项集； 集合理论； 支持度

中图分类号 TP301.6 **文献标识码** A doi:10.3969/j.issn.1001-0548.2010.05.023

Research on Top-*N* Most Frequent Itemsets Mining

ZHU Hao-dong and LI Hong-chan

(School of Computer Communication Engineering ,Zhengzhou University of Light Industry Zhengzhou 450002)

Abstract Most frequent item sets mining is the focus and the difficulty of text association rules mining, and it directly determines the performance of text association rules mining algorithms. Firstly, several most frequent item sets mining algorithms are analyzd and summarized. And then, traditional inverted list is improved. Based on the improved list and set theory, a new TOP-N most frequent itemset mining algorithm combined minimum support threshold dynamic adjustment strategy is presented. In addition, several propositions and deductions for improving the performance of the performance of the provided algorithm are offered. Experimental results show that the provided algorithm is better than Napriori and IntvMatrix.

Key words association rules; inverted list; requent itemsets; set theory; supports;

随着最频繁项集数目的增加，文本关联规则的个数及其相应的挖掘算法的时空复杂度都急剧上升，使最频繁项集的挖掘成为文本关联规则挖掘中研究的重点和难点[1-15]。在最频繁项集挖掘过程中，最小支持度阈值是一个十分关键的参数，该参数一般都是通过人工指定，但主观指定方式很难符合客观实际。参数值指定过高，导致一些有价值的规则丢失；参数值指定过低，导致规则数量剧增，生成许多无用规则，降低挖掘算法的性能。文献[1]表明：当语料库的文档为1 000篇、文本事务集的特征数目达到100个、最小支持度阈值为0.1时，其频繁项集的数目高达2 316个，产生的规则数量十分巨大，但有用的规则却很少。

为了解决上述问题，已有众多学者提出了一些相关的算法[1-15]，大都摈弃最小支持度阈值，通过指定最频繁项集的数量*N*控制所生成的频繁项集规模。这些算法的不足之处在于：由于没有设定最小支持度阈值，在挖掘过程中很可能对支持度较小的频繁项集也进行处理，使算法的时空复杂度较高，尤其在文档数量巨大时，时空复杂度会更高。

文献[1-2,6-7]提出了挖掘Top-*N*最频繁项集的算法；文献[3]基于最小支持度阈值动态调整策略，提出了挖掘Top-*N*最频繁*k*-项集的Itemset-loop算法。经研究发现，Itemset-loop算法虽然在一定程度上弥补了文献[1]算法的缺点，但该算法在*k*步骤中需要找出全部*N*个最频繁*k*-项集，如果*m*为频繁项集的最大长度，意味着Top-*N*的候选项集中有*N*×*m*个频繁项集；此外，该算法所使用的最小支持度阈值存在减小的可能，当频繁*k*+1-项集的最小支持度阈值小于频繁*k*-项集的最小支持度阈值时，也即 $NS_{k+1} < NS_k$ 时，必须回溯重新挖掘最小支持度为 NS_{k+1} 的频繁*k*-项集，降低了挖掘算法的性能。文献[3]给出的NApriori算法，首先设定一个最小支持度作为全局统一的最小支持度阈值，然后用该支持度阈值与每步频繁项

收稿日期：2008－08－08； 修回日期：2009－03－18
基金项目：四川省科技计划项目(2008GZ0003)
作者简介：朱颢东(1980－)，男，博士，主要从事软件过程技术与方法、文本挖掘等方面的研究.

集的支持度阈值作对比，如果当前全局最小支持度阈值较小，就令它的值为当前的最小支持度阈值。虽然NApriori算法克服了Itemset-loop算法的不足，但该算法需要多次扫描事务数据库，时间复杂度较高。文献[5]在文献[6-7] COFI树和倒排矩阵的基础上，提出了一个IntvMatrix算法，该算法采用倒排矩阵索引结构加快频繁项集的生成速度。但是倒排矩阵中存在众多空元素，并且是在内存中对倒排矩阵进行检索的，会浪费很多内存空间。另外，该算法在生成候选频繁项集过程中需多次扫描倒排矩阵，也会在一定程度上降低该算法的效率。

为了克服IntvMatrix算法的不足，本文借用文献[3]动态调整支持度阈值的思想，提出一种基于集合和倒排表的Top-*N*最频繁项集挖掘算法。通过实验对比，该算法优于NApriori算法和IntvMatrix算法。

1 相关知识

定义 1 把全部项集依照支持度从大到小排序，假设*NS*等于第*N*位项集的支持度，则最频繁项集：

$$\text{Top-}N=\{X|\text{support}(X)>=NS\}. \quad (1)$$

因为支持度等于*NS*的项集个数可能有多个，Top-*N*中的最频繁项集的数目有可能多于*N*个。例如，如果第*N*+1、*N*+2、…、*N*+*m*个项集的支持度也是*NS*，则Top-*N*由*N*+*m*个频繁项集组成。

定义 2 把全部*k*-项集依照支持度从大到小排序，假设 NS_k 等于第*N*位的*k*-项集的支持度，则最频繁*k*-项集：

$$\text{Top-}N_k=\{X|\text{support}(X)>=NS_k \text{ 且 } |X|=k\} \quad (2)$$

命题 1 假设Top-*N*的最小支持度是*NS*，Top-N_k的最小支持度是 NS_k，则$NS>=NS_k$。

证明 假设$NS<NS_k$，从定义1可知Top-*N*中一定存在一个频繁项集Item的支持度为*NS*；从定义2可知，Top-N_k中也一定存在频繁*k*-项集Item_k，其支持度为 NS_k；根据假设可知，Item_k比Item的支持度高，但Item∈Top-*N*中$\text{Item}_k\notin$Top-*N*，与Top-*N*是*N*个最频繁项集矛盾。因此，命题1成立。

推论 1 Top-*N*与Top-N_k的关系为 $\text{Top-}N \subseteq \{\bigcup_k \text{Top-}N_k\}$。

证明 假设存在一个频繁项集Item∈Top-*N*，但Item$\notin$Top-N_k，即 $\text{Top-}N \not\subseteq \{\bigcup_k \text{Top-}N_k\}$。如果Item是*m*-项集，则Item∈Top-*N*，但Item$\notin$ Top-N_m。因为Item∈Top*N*，由命题1可知Item的支持度大于Item_m的最小支持度 NS_m，就存在一个不属于Top-N_m且支持度大于 NS_m的频繁*m*-项集Item_m，与Top-N_m是*N*个最频繁*m*-项集矛盾。推论得证。

由推论1可知，$\{\bigcup_k \text{Top-}N_k\}$是Top-*N*的候选项集。所以，可以先用Itemset-loop算法生成全部Top-N_k，然后再从中选取最频繁的*N*个项集组成Top-*N*，就可以降低算法的时间复杂度。

命题 2 假设Top-N_k的支持度阈值为 NS_k，Top-N_{k-1}的支持度阈值为 NS_{k-1}，则 $NS_k >= NS_{k-1}$。

证明 假设 $NS_k < NS_{k-1}$，由文献[3]的NApriori算法和文献[5]的IntvMatrix算法可知，Top-N_k是由Top-N_{k-1}与第*k*轮新生成的当前频繁项集中支持度>= NS_k 的项集组成，因此Top-N_k包括Top-N_{k-1}中的所有项集。由假设 $NS_k < NS_{k-1}$知，至少有一个项集*x*满足 NS_k =<support(*x*)< NS_{k-1}，显然*x*ï Top-N_{k-1}，*x*î Top-N_k，所以Top-N_k至少包括Top-N_{k-1}中所有支持度>= NS_{k-1} 的*N*个项集以及项集*x*，而且support(*x*)< NS_{k-1}。由此可知，Top-N_{k-1}是当前*N*+1个最频繁项集，与Top-N_k是当前*N*个最频繁项集矛盾。因此，命题2成立。

由命题2可知，在生成Top-*N*的过程中，当前最小支持度阈值被逐步增大，减少了候选项集的规模，提高了算法效率。

2 本文所提推论

倒排表是一种高级索引结构，由词表和文档表两部分组成。词表由文档集中的特征词组成，对词表的任一特征词，在文档表中都有一行与包含该特征词的文档ID所对应。在IR领域，倒排表常用于文本索引，可以提高查找速度。

表1为一个文档事务数据库，表2是其相应的倒排表。

表1 文档事务数据库

ID	Items(Features)						
1	*a*	*b*	*c*	*d*	*e*		
2	*a*	*c*	*e*	*h*	*g*	*r*	
3	*b*	*c*	*d*	*a*	*e*		
4	*f*	*a*	*h*	*g*	*j*	*p*	*q*
5	*a*	*b*	*c*	*e*	*i*		
6	*k*	*a*	*e*	*i*	*c*		
7	*h*	*c*	*g*	*i*			
8	*k*	*l*	*m*	*n*	*o*		
9	*l*	*q*	*a*				
10	*n*	*b*	*a*	*m*			

表2　相应的倒排表

Items	ID
a	1 2 3 4 5 6 9 10
b	1 3 5 10
c	1 2 3 5 6
d	1 3
e	1 2 3 5 6 7
f	4
g	2 4 7
h	2 4 7
i	5 6 7
j	4
k	6 8
l	8 9
m	8 10
n	8 10
o	8
p	4
q	4 9
r	2

倒排表虽然在一定程度上提高了单一特征词的查找效率，也即1-项集的查找效率，但是对其他*k*-项集而言，很难获得相应的支持度，这是因为倒排表使得出现在同一事务中的各个项之间变得相互独立; 并且从表2还可以看出，倒排表存在大量空元素，极大地浪费了内存空间。因此，本文对倒排表进行了改造。

(1) 在词表中，各项按其文档频从大到小的顺序排序，并用序号表明其位置。此时的词表由序号、项名、指向文本事务集合的指针3部分组成。

(2) 文档表中，每一行表示一个集合，集合元素由特征出现的文本事务号组成。此时表中各项以其所包含的元素个数降序排列。表3为改造后对应的倒排表。

推论 2　设T_1、T_2为项集，T_1所在的全部文档集为D_1，T_2所在的全部文档集为D_2，则对项集$T=T_1\cup T_2$，T所在的全部文档集为$D=D_1\cap D_2$。

证明　(1) $\forall d\in D$，由$D=D_1\cap D_2$可得$d\in D_1$，因为$T_1\subseteq D_1$，所以$T_1\subseteq d$，同理也有$T_2\subseteq d$。因此，$D=D_1\cap D_2$包含$T=T_1\cup T_2$。(2) $\forall d\supseteq T$，由$T=T_1\cup T_2$可得$T_1\subseteq T$，所以$T_1\subseteq d$。又由D_1是包含T_1的全部文档集，所以$d\subseteq D_1$。同理也有$d\subseteq D_2$，所以$d\subseteq D=D_1\cap D_2$。推论得证。

根据推论2，在进行连接操作时就不需要再次扫描数据库，如连接时*a*与*e*只需对它们的指针所指的集合取交集，就能够提高算法效率。*a*与*e*连接结果如表4所示。

表3　改造表1后的倒排表

项号	项名	指针所指向的集合
1	*a*	{1,2,3,4,5,6,9,10}
2	*e*	{1,2,3,5,6,7}
3	*c*	{1,2,3,5,6}
4	*b*	{1,3,5,10}
5	*g*	{2,4,7}
6	*h*	{2,4,7}
7	*i*	{5,6,7}
8	*d*	{1,3}
9	*k*	{6,8}
10	*l*	{8,9}
11	*m*	{8,10}
12	*n*	{8,10}
13	*q*	{4,9}
14	*f*	{4}
15	*j*	{4}
16	*o*	{8}
17	*p*	{4}
18	*r*	{2}

表4　*a*与*e*连接结果

项号	项名	指针所指向的集合
19	*a*-*e*	{1,2,3,5,6}

推论 3　设L_k为*K*-频繁项集的集合，如果L_k中的项集个数不大于*K*，则L_k为极大频繁项集。

证明　经典Apriori算法指出，任何强项集的子集必定是强项集。因此可知，如果存在L_{k+1}(即*K*+1频繁项集的集合)，则$\forall l_{k+1}\in L_{k+1}$，$l_{k+1}$一定有*K*+1个*k*-频繁子集在$L_k$中，因此，如果$L_k$的项集个数不大于*K*，则必定不能生成$L_{k+1}$。推论得证。

根据推论3，在产生(*k*+1)-项候选频繁项集之前，先统计*k*-项频繁集中项集的个数，如果数目<=*k*，则算法可以终止，也能提高算法效率。

推论 4　设$\forall l_k\in L_k$(L_k为*k*-频繁项集)，$\forall$Item$\in l_k$，如果Item在L_k中的支持数小于*K*，则l_k不能用于生成L_{k+1}。

证明　经典Apriori算法指出，任何强项集的子集必定是强项集。因此可知，$\forall l_{k+1}\in L_{k+1}$，$l_{k+1}$中必然存在*K*+1个*k*-频繁项集属于$L_k$。明显有$\forall$Item$\in l_k$，在$l_k$的*K*+1个*k*-频繁项集中，Item的支持数至少为*K*。所以，$\exists$Item$\in l_k$，且Item在L_k中的支持数小于*K*，则l_k不能用于生成L_{k+1}。推论得证。

根据推论4，如果某个*k*-项频繁项集中的一项在其中出现的次数<*k*，则该项集不能被用于连接生成(*k*+1)-项频繁项集，在连接时将其排除，可以减少候选频繁集的数目，提高算法的效率。

3　本文算法

3.1　本文算法思想

借用文献[3]中动态调整支持度阈值的思想，利用文中所给出的几个命题和结论，将倒排表和集合

相结合，可以获得一个Top-N最频繁项集挖掘算法。

3.2 算法描述

输入：文本事务数据库D，最小支持数δ_0(初始值为1)，频繁项集数N；

输出：Top-N最频繁项集。

步骤 1 扫描D以产生改造的倒排表W。

步骤 2 如果倒排表W中各项的频率<N，则令$\delta=\delta_0$（为当前最小支持数）；否则，令δ=max{δ_0,δ_N}(δ_N为倒数第N个项的支持数)，以支持数>δ的前N项组成L_1(L_1为N个最频繁1-项集，形式为改造后的倒排表)。

步骤 3 for (k=2；Count (L_{k-1})≥k；k++)

{ L'_{k-1}=Delete (L_{k-1})//为了减少k-项候选集的个数，利用推论3从中L_{k-1}删除不能产生频繁k-项集的k–1-项集；

C_K=Candidate_gen (L'_{k-1});

如果C_K中元素个数<N，则令$\delta=\delta_0$；否则，令δ=max{δ_0,δ_{kn}}作为支持数(其中δ_{kn}为倒数第N个项的支持数)，支持数>=δ的前N项组成L_k(L_k为当前N个最频繁k-项集，形式为改造后的倒排表)。

步骤 4 令$L=\bigcup_k L_k$，以降序形式排列L中的项集并从中取前N个频繁项集输出，算法结束。

3.3 本文算法过程说明

Procedure Delete(L_{k-1})//从L_{k-1}中删除包含项次数小于k–1的项集；

给定∀项i，初始值i.count=0

{ For all l in L_{k-1}中的项集部分

If $i\in l$ then i.count++//计算项i在L_{k-1}中出现的次数;}

If (i.count<k–1) then

{ For all l in L_{k-1}中的项集部分

If $i\in l$ then 则从L_{k-1}中删除项集部分为l的频繁项集;}

Return L_{k-1}

Procedure Candidate_gen(L'_{k-1})//产生k-项候选集

begin

for ∀项集 $l_1\in L'_{k-1}$

for ∀项集$l_2\in L'_{k-1}$

如果 l_1、l_2的前k–2个项相同而第k–1项不同，则c=jion(l_1,l_2)//连接以产生候选项集集合；

if has_infrequent_subset(c, L_{k-1}) then 删除c//剪枝以删除非频繁候选项集

else 把c 加入到C_K；

return C_K；

End

Procedure jion(l_1,l_2)

Begin

令l_1、l_2中的项集部分别为l'_1、l'_2，指针所指的事务集分别为l_1^1、l_2^2；则令$l_c=l'_1\infty l'_2$为连接后c的项集部分，$l_c^c=l_1^1\cap l_2^2$为c的事务集部分。

Return c

End

procedure has_infrequent_subset(c, L_{k-1})// 判断L_{k-1}的子集是否有非频繁集

begin

for each (k−1)-subset s of c的项集部分

if $s\notin L_{k-1}$的项集部分 then return TRUE；

return false

End

4 实验验证及其分析

4.1 实验验证

实验使用数据集由从新华网上下载的新闻材料组成。新闻材料的发表日期范围为2006～2008年，共1 500篇，经预处理后(忽略所有的报头)，挖掘该数据集的频繁项集并进行实验，比较本文算法与文献[3]提出的NAprioir算法和文献[5]提出的IntvMatrix算法的性能差异。

实验 1 比较3种算法在所选择数据集上挖掘出的规则数目和有效规则数，结果如表5所示。

表5 3种算法的有效规则率对比

算法名称	挖掘出的规则数	有效规则数	有效率/(%)
本文算法	462	455	98.48
IntvMatrix	538	454	84.39
NAprioir	697	437	62.70

实验 2 比较3种算法对不同规模的频繁项集进行挖掘时所消耗的时间，从实验结果选取差别较明显的实验数据作对比图，如图1所示。

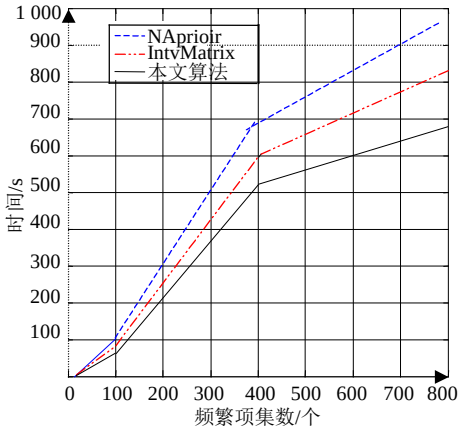

图1 3种算法执行时间对比

4.2 实验分析

从表5可以看出，由于本文算法采用支持数动态自适应调整策略，所挖掘的规则有效率较高；IntvMatrix算法的规则有效率次之；Napriori算法产生了大量无意义的规则，规则有效率最低。

从图1可以看出，在频繁项集个数相同的情况下，本文算法的时间性能明显优于IntvMatrix和Napriori算法，分析其原因在于：

(1) 本文算法采用倒排索引这种数据结构组织事务集，提高了检索速度。

(2) 本文算法扫描数据库过程的步骤1中，以改造后的倒排表组织文档并对其进行扫描，这是该算法唯一一次扫描数据库，对于海量数据库来说，其时间效率的提高很明显。

(3) 本文算法中Count(L_{k-1})函数用来计算L_{k-1}中频繁集的个数，根据推论3，如果L_{k-1}中频繁项集的个数小于k，则无需再生成L_k，此时算法可以结束，也提高了算法的时间效率。

(4) 根据推论4，Delete(L_{k-1})函数用来从L_{k-1}中删除包含项次数小于k–1的项集，从而减少k-项候选集的数目，可以解决“候选项集瓶颈”问题，也提高了算法的时间效率。

(5) 本文算法中，Candidate_gen(L'_{k-1})函数主要用于生成k-项候选集，它分别调用jion(l_1,l_2)和has_infrequent_subset(c, L_{k-1})两个函数。其中jion(l_1,l_2)函数用于实现频繁项集的连接及求两个事务集合的交集(推论2)；has_infrequent_subset(c, L_{k-1})函数用于判断一个k-项候选集的k–1子集之中是否有非频繁项集，如果有则该k-项候选集被删除，否则该k-项候选集被加入到k-项候选集集合之中(推论1)。

5 结 束 语

本文针对高维文本特征空间中仅以最小支持度阈值为约束条件产生的频繁项集规模难以确定的不足，在分析NApriori和IntvMatrix两个经典算法的基础上，提出了一种基于倒排表和集合的TOP-*N*最频繁项集算法。通过分析和实验表明，无论是在规则有效率方面，还是时间效率方面，本文算法都优于NApriori算法和IntvMatrix算法，使本文算法在文本关联规则挖掘中有一定的应用价值。

参 考 文 献

[1] 陈晓云. 文本挖掘若干关键技术研究[D]. 上海: 复旦大学, 2005.
CHEN Xiao-Yun. Research on several key technology of text mining[D]. Shanghai: Fudan University, 2005.

[2] HAN Jia-wei, PEI Jian, YIN Yi-wen. Mining frequent patterns without candidate generation:a frequent pattern tree approach[J]. Data Mining and Knowledge Discovery, 2004, 8(1): 53-87.

[3] FU A W C, KWONG R W W , TANG J. Mining *N*-most interesting itemsets[C]//Proceedings of 2000 ISMIS. Berlin: Springer, 2000: 59-67.

[4] BODON F. A survey on frequent itemset mining[C]// Proceedings of the ACM SIGKDD Workshop on OSDM ’04. Chicago, USA: [s.n.], 2004: 523-531.

[5] 陈晓云, 胡运发. *N*个最频繁项集挖掘算法[J]. 模式识别与人工智能, 2007, 20(4): 512-518.
CHEN Xiao-yun, HU Yun-fa. Mining algorithms of *N*-most frequent itemsets[J]. PR & AI, 2007, 20(4): 512-518.

[6] HAJJ M E, ZAIANE O R. Inverted matrix: Efficient discovery of frequent items in large datasets in the context of interactive mining[C]//2003 Int’1 Conf on Data Mining and Knowledge Discovery (ACM SIGKDD). California, USA: [s.n.], 2003: 109-118.

[7] HAJJ M E, ZAIANE O R. Non recursive generation of frequent k-itemsets from frequent pattern tree representations[C]//Proceedings of 5th International Conference on Data Warehousing and Knowledge Discovery. Melbourne: Australia, 2003: 371-380.

[8] RACZ B. NonordFP：an FP-growth variation without rebuilding the FP-tree[C]//Proceedings of the IEEE ICDM Workshop on FIMI ’04. Brighton, UK: [s.n.], 2004: 1089-1097.

[9] LIU Gui-mei, LU Hong-jun. AFOPT: an efficient implemefitation of pattern growth approach[C]//Proceedings of the IEEE ICDM Workshop on FIMI ’04. Brighton, UK: [s.n.], 2004: 2056-2067.

[10] 陈　耿, 朱玉全, 杨鹤标, 等. 关联规则挖掘中若干关键技术的研究[J]. 计算机研究与发展, 2005, 42(10): 1785-1789.
CHEN Geng, ZHU Yu-quan,Yang He-biao. Study of some key techniques in mining association rule[J]. Journa1 of Computer Research and Development, 2005, 42(10): 1785-1789.

[11] 陈晓云, 陈　祎, 王　雷, 等. 基于分类规则树的频繁模式文本分类[J]. 软件学报, 2006, 17(5): 1017-1025.
CHEN Xiao-yun, CHEN Wei, WANG Lei, et a1. Text categorization based on classification rules tree by frequent patterns[J]. Journal of Software, 2006, 17(5): 1017-l025.

[12] WU Fan, CHIANG S W, LINJ R. A new approach to mine frequent patterns using item-transformation methods[J]. Information Systems, 2007, 32(7): 1056-1072.

[13] 战力强, 刘大昕. 频繁项集快速挖掘算法研究[J]. 哈尔滨工程大学学报, 2008, 29(3): 266-271.
ZHAN Li-qiang, LIU Da-xin. Study on fast algorithm of frequent item-set mining [J]. Journal of Harbin Engineering University, 2008, 29(3): 266-271.

(下转第773页)

Transactions on Signal Processing, 2003, 51(7): 1995-2007.
[6] 张 涛, 平西建. 空域LSB信息伪装的隐写分析及其对策[J]. 通信学报, 2003, 24(12): 156-163.
ZHANG Tao, PING Xi-jian. Steganalysis of spatial LSB-based steganographic algorithms and countermeasures [J]. Journal of China Institute of Communications, 2003, 24(12): 156-163.
[7] 罗向阳, 陆佩忠, 刘粉林. 一类可抵御SPA分析的动态补偿LSB信息隐藏方法[J]. 计算机学报, 2007, 30(3): 463 - 473.
LUO Xiang-yang, LU Pei-zhong, LIU Fen-lin. A dynamic compensation LSB steganography method defeating SPA[J]. Chinese Journal of Computers, 2007, 30(3): 463-473.
[8] LUO Xiang-yang, HU Zong-yun, YANG Can, et al. A secure LSB steganography system defeating sample pair analysis based on chaos system and dynamic compensation[C]// Advanced Communication Technology on the 8th International Conference. Phoenix Park: [s.n.], 2006.
[9] 田 源, 程义民, 谢于明, 等. 一种抗SPA分析的图像信息隐藏方法[J]. 中国科学技术大学学报, 2008, 38(12): 1376-1380.
TIAN Yuan, CHENG Yi-min, XIE Yu-ming, et al. Steganographic scheme for images against SPA steganalysis [J]. Journal of University of Science and Technology of China, 2008, 38(12): 1376-1380.
[10] 汪小帆, 戴跃伟, 茅耀斌. 信息隐藏技术-方法与应用[M]. 北京: 机械工业出版社, 2001: 124-125.
WANG Xiao-fan, DAN Yue-wei, MAO Yao-bin[M]. Information Hiding Technology-Methods and application. Beijing: China Machine Press, 2001: 124-125.
[11] WANG Hong, PENG Jian-hua, ZHOU Zheng-ou. Design of a new chaos circuit and its encryption to digital information[J]. Journal of Electronic Science and Technology of China, 2004, 2(4): 25-28.

编 辑 蒋 晓

(上接第746页)

[7] POPOVIC M R, GOLDENBERG A A. Modelling of friction using spectral analysis[J]. IEEE Transaction on Robotics and Automation, 1998, 14(1): 114-122.
[8] TAGHIRAD H D, BELANGER P R. Model1ing and parameter identification of harmonic drive systems[J]. J of Dynamic Systems, Measurement, and Control, 1998, 120(6): 439-444.
[9] TUTTLE T D, SEERING W P. A Nonlinear Model of a Harmonic Drive Gear Transmission[J]. IEEE Transaction on Robotics and Automation, 1996, 12(3): 368-374.
[10] LEMMER L, KISS B. Modeling, identification, and control of harmonic drives for mobile vehicles[C]// Proceedings of the IEEE 3rd International Conference on Mechatronics. Budapest: IEEE Press, 2006: 369-374.
[11] CHOI J J, HAN S I, KIM J S. Development of a novel dynamic friction model and precise tracking control using adaptive back-stepping sliding mode controller[J]. Mechatronics, 2006, 16(2): 97-104.
[12] JATTA F, LEGNANI G, VISIOLI, A. Friction compensation in hybrid force/velocity control of industrial manipulators[J]. IEEE Transaction Industry Electronics, 2006, 53(2): 604-613.

编 辑 张 俊

(上接第761页)

[14] 田 宏, 董爱杰. 基于向量矩阵的频繁项集挖掘算法[J]. 大连交通大学学报, 2008, 29(3): 74-77.
TIAN Hong, DONG Ai-jie. A frequent itemsets mining algorithm based on vector matrix[J]. Journal of Dalian Jiaotong University, 2008, 29(3): 74-77.
[15] 张忠平, 李 岩, 杨 静. 基于矩阵的频繁项集挖掘算法[J]. 计算机工程, 2009, 35(1): 84-86.
ZHANG Zhong-ping, LI Yan, YANG Jing. Frequent itemsets mining algorithm based on matrix[J]. Computer Engineering, 2009, 35(1): 84-86.

编 辑 漆 蓉