频繁和高效用项集挖掘

李 慧 刘贵全 瞿春燕
(中国科技大学计算机与技术学院 合肥 230000)

摘 要 对从事务数据库中挖掘有意义的项集的研究已超过10年。然而,大多数的研究要么使用频繁度或支持度(如频繁项集挖掘),要么使用效用值或利润(如高效用项集挖掘)作为主要的衡量标准。单独使用这两种衡量方式都有各自的局限性,比如频繁度很高的项集其效用值有可能很低,而效用值很高的项集其频繁度往往很低,将这些项集推荐给用户没有意义。将这两种衡量标准综合考虑,希望找出那些频繁度和效用值都很高的项集。该项工作最大的挑战是效用值既不满足单调性也不满足反单调性。因此,提出了高效算法FHIMA。FHIMA采用PrefixSpan的思想,挖掘时能避免产生非频繁的候选项集。此外,还根据效用和质量上界的一些性质,有效地缩小了搜索空间,极大地提高了FHIMA算法的效率。

关键词 Top-k,频繁,高效用,高质量项集
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.5.017

Mining Frequent and High Utility Itemsets

LI Hui LIU Gui-quan QU Chun-yan
(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230000, China)

Abstract Mining interesting itemsets from transaction database has attracted a lot of research work for more than a decade. However, most of these studies either use frequency/support (e.g., frequent itemset mining) or utility/profit (e.g., high utility itemset mining) as the key interestingness measure. In other words, these two measures are considered individually, which leads to some shortages that frequent itemsets may have low profit, or high profit itemsets may have very low support, so it is meaningless to recommend these itemsets to users. To this end, we considered these two measures from a unified perspective. Specifically, we proposed to identify the qualified itemsets which are both frequent and high utility. The key challenge to these problems is that the value of utility does not change monotonically when we add more items to a given itemset. Thus, we proposed an efficient algorithm named FHIMA (Frequent and High utility Itemset Mining Algorithms), where an effective upper bound based on frequency and utility is designed to further prune the search space. Moreover, FHIMA incorporates the idea of Prefixspan to avoid generating candidates, thus leading to high efficiency. Finally, the experiment results demonstrate the efficiency of FHIMA on real and synthetic datasets.

Keywords Top-k, Frequent, High utility, Qualified itemsets

1 简介

频繁项集挖掘[1,5-7,9,10]是希望挖掘出在数据库中大量出现的项集。如果一个项集的频繁度/支持度大于或等于用户指定的阈值,则称这个项集是频繁项集。但是频繁项集挖掘无法满足用户对高利润项集的需求,因此高效用项集挖掘[2,4,11,18,19,21,22]近几年被提出,用来挖掘那些有高效用(利润)的项集。在高效用项集挖掘中,每个项目都有唯一的权值(利润),并且在不同的事务中会出现不同的次数。如果一个项集的效用值大于或等于用户指定的阈值,则称这个项集是高效用项集。在目前的研究中,大多数要么使用频繁度/支持度,要么使用效用值/利润作为最主要的衡量手段,但这两种方法都有各自的局限性。具体来说,频繁项集挖掘可能找到大量频繁的但利润很低的项集,而高效用项集挖掘往往找到大量的利润值很高但是支持度很低的项集。例如表1(利润表)和表2(事务表),在利润表中有6个项,在事务表中有5个事务。项集$\{a\}$的频繁度和效用值分别为4和13;项集$\{f\}$的频繁度和效用值则分别为1和63。尽管项集$\{a\}$在不同的事务中出现了很多次,但是它的利润很低;向用户推荐那些像$\{f\}$那样效用值很高的项集也没有什么价值,因为$\{f\}$仅仅在t_2中出现了一次,而这很可能是偶然发生的。因此,向用户推荐这些项集往往没有意义。

本文将这两种衡量标准综合起来考虑,希望找到那些高质量的项集,即那些既频繁又有高效用的项集。另外,在所有

到稿日期:2014-03-20 返修日期:2014-05-07 本文受中央高校基本科研基金(WK2100100021),国家科技支撑计划(2012BAH17B03),安徽省自主创新专项-智能语音技术研发和产业化专项(13Z02008-5)资助。

李 慧(1990-),男,硕士,主要研究方向为数据挖掘、自然语言理解,E-mail:lopohui@mail.ustc.edu.cn;刘贵全(1970-),男,副教授,主要研究方向为机器学习、智能信息处理与挖掘、互联网信息抽取及深度搜索与挖掘;瞿春燕(1986-),女,硕士,主要研究方向为数据挖掘。

这些高质量的项集中，用户往往只对 top-k 的项集感兴趣，所以我们将问题形式化表述为挖掘 top-k 个频繁和高效用项集，即在对支持度和相对效用值做加权和得到质量值(quality)后，取 k 个最高质量值的项集推荐给用户。其中最大的挑战是，效用值既不满足单调性也不满足反单调性。因此，我们提出了一种被称为 FHIMA(frequent and high utility itemsets mining)的算法。FHIMA 利用了效用和质量的一些上界的性质，能够有效地缩小搜索空间。另外，FHIMA 采用了 PrefixSpan 的思想，在挖掘的过程中避免了产生非频繁的候选项集，进一步提高了算法的效率。

表 1 利润表

Item	a	b	c	d	e	f
Profit	1	2	1	1	1	3

表 2 事务表

t_{id}	Transaction	Count
t_1	{a,b,c,d}	{2,1,3,6}
t_2	{a,b,c,f}	{7,4,2,21}
t_3	{a,b,d,e}	{3,2,2,1}
t_4	{a,d,e}	{1,3,1}
t_5	{c,d}	{9,9}

该研究的意义主要包括：

1)在一个统一的框架中考虑支持度和效用值。具体来说，提出了一种新的方法来挖掘那些高质量的项集(既频繁又有高效用的项集)。另外，在所有高质量的项集中，将问题转化为挖掘 top-k 的频繁和高效的项集。

2)提出了一个新的算法来解决挖掘高质量项集的问题。具体来说，FHIMA 采用了 PrefixSpan 算法的思想，避免了非频繁候选项集的产生，另外利用了效用和质量上界的性质进一步缩小了搜索空间，提高了算法的效率。

3)在各种真实和模拟的数据上验证了 FHIMA 算法的效率，并且通过设置不同的参数，对不同的计算上界的方法进行比较来进一步对算法进行评估。

2 背景

2.1 定义和符号

设 $I=\{i_1,i_2,\cdots,i_m\}$ 为所有项的完备集，I 中每个项 i_p 都有对应的利润，记为 $p(i_p)$，如表 1 所列。一个事务表 $D=\{t_1,t_2,\cdots,t_m\}$ 是事务的集合，其中每个事物都有一个唯一的标示符 t_{id} 和它所包含的项，以及每个项对应出现的次数，事务 t_d 中每个项目 i_p 的数量记为 $c(i_p,t_d)$，如表 2 所列。项的集合称为项集，记为 $\{i_{p_1},i_{p_2},\cdots,i_{p_m}\}$。若事务 t_i 包含项集 X 中所有的项，则称 t_i 为项集 X 的支持数据库，X 的支持数据库的集合记为 T_X。一个项集中包含的项的个数，称为项集的长度。一个长度为 k 的项集，也叫做 k-项集。

项集 X 的频繁度是 D 中包含 X 的事务的个数，记为 $fre(X)$。一个项集的支持度是 D 中包含 X 的事务所占的比例，记为 $\sigma(X)$。如果 $\sigma(X)$ 大于或等于用户指定的最小支持度阈值 α，则称 X 是频繁的。令 $\alpha=0.4$，则项集 $\{f\}$ 不是频繁的。

定义 1 项目 i_p 在事务 t_d 的效用 $u(i_p,t_d)$，定义为 $p(i_p)*c(i_p,t_d)$。

例如：在表 1 和表 2 中，$p(b)=2$，$c(b,t_1)=1$，因此 $u(b,t_1)=p(b)*c(b,t_1)=2*1$。

定义 2 项集 X 在其支持事务 t_d 的效用 $u(X,t_d)$，定义为 $\sum_{i_p\in X\wedge X\in t_d} u(i_p,t_d)$。

定义 3 项集 X 在 D 中的效用值 $u(X)$，是 X 在所有支持事务的效用值之和，定义为 $\sum_{t_d\in T_X} u(X,t_d)$。

例如：在表 1 和表 2 中，$u(\{bc\},t_1)=u(b,t_1)+u(c,t_1)=2\times1+1\times3=5$，$u(\{bc\},t_2)=u(b,t_2)+u(c,t_2)=2\times4+1\times2=10$，因此 $u(\{bc\})=u(\{bc\},t_1)+u(\{bc\},t_2)=5+10=15$。

定义 4 D 的总效用值 $TU(D)$，是所有 1-项集效用值之和，定义为 $\sum_{i_p\in I\wedge t_d\in D} u(i_p,t_d)$。

定义 5 项集 X 的相对效用值 $\phi(X)$，用来衡量 $u(X)$ 占 $TU(D)$ 的比例，定义为 $u(X)/TU(D)$。

定义 6 给定项集 X 以及用户指定的最小相对效用的阈值 β，若 $\phi(X)\geqslant\beta$，则称 X 为高效用项集。所有高效用项集的集合称为 $HUIS$(high utility itemsets)。

例如：$TU(D)=(u\{a\})+u(\{b\})+u(\{c\})+u(\{d\})+u(\{e\})+u(\{f\})=13+14+14+20+2+63=126$。假设 $\beta=0.5$，$HUIS$ 中则有 $\{f\}$：63，$\{af\}$：70，$\{bf\}$：71，$\{cf\}$：65，$\{abf\}$：78，$\{acf\}$：72，$\{bcf\}$：73，$\{abcf\}$：80，在每个项集后面的数字代表其效用值。

频繁度具有反单调性。而效用值既不具有单调性也不具有反单调性，即当有更多的项加入到一个给定的项集中时，新的项集的效用值既不会单调递增，也不会单调递减。例如，给定项集 $\{e\}$，$u(\{e\})=2$，将项 d 加入，$u(\{de\})=7$。将项 b,d 加入之后 $u(\{bde\})=5$。项集 $\{de\}$ 的效用值比项集 $\{e\}$ 的效用值大，但是项集 $\{bde\}$ 的效用值反而比项集 $\{e\}$ 的效用值小。为了解决这个问题，Y. Liu 等[13]基于下面的定义，提出了事务加权向下闭合的策略，来减小搜索空间。

定义 7 事务 t_d 的效用 $tu(t_d)$，定义为 $u(t_d,t_d)$。

例如：$tu(t_1)=u(\{abcd\},t_1)=13$。

定义 8 项集 X 的事务加权效用值 $TWU(X)$，是 T_X 中所有事务的事务效用之和，定义为 $TWU(X)=\sum_{t\in T_X} tu(t_d)$。

性质 1 事务加权向下闭合。给定项集 X，如果 $TWU(X)\leqslant\beta*TU(D)$，则项集 X 以及 X 的所有超集都不是高效用项集。

证明：如果 $X\subseteq X'$，那么 $u(X')\leqslant TWU(X')\leqslant TWU(X)<\beta*TU(D)$。

2.2 问题公式化

根据前面对支持度和效用值的定义，我们可以将两者结合起来计算项集的质量值。

定义 9 项集 X 的质量 $q(X)$，是支持度和相对效用值的加权和，定义为 $\sigma(X)+\lambda\phi(X)$。其中 $\lambda\geqslant0$ 是用户指定的相对效用值的权重参数，用来调整支持度和相对效用值两者的相对重要程度。

因为质量值是二者的加权和，所以很容易证明在 $\lambda>0$ 时，质量值既不具有单调性也不具有反单调性。

给定事务数据库 D、最小支持度阈值 α 以及最小相对效用值阈值 β，目标是在 D 中找出所有的高质量项集，即那些既频繁又是高效用的项集。

实际中找出所有的高质量的项集很费时；另外，用户往往只对那些比较重要的项集感兴趣。因此，将问题转化为挖掘 top-k 个频繁和高效用的项集。

2.3 相关工作

R. Agrawal 和 R. Srikant[1] 最早提出了频繁项集挖掘问题，并且提出了 Apriori 算法。2000 年 Jiawei Han 等人[9] 提出了基于 FP 树生成频繁项集的 FP-growth 算法。FP-growth 不产生任何候选项集，并且只需要对原来的数据库进行两次扫描，所以它比 Apriori 算法快了近一个量级。

高效用数据挖掘最早是 H. Yao 等[22] 提出来的。因为效用不具有单调性和反单调性，Y. Liu 等[12] 提出了事务加权向下闭合的策略来缩小搜索空间。2004 年，K. Gade[8] 提出了 3 种修剪策略，进一步减小了搜索空间。之后很多两阶段的算法被提出，比如 IHUP[3]、UP-Growth[19]、TKU[21] 等。在第一阶段，这些算法挖掘出潜在的高效用的项集。在第二阶段，通过扫描原来的数据库，计算出第一阶段挖掘出来的项集的真正效用值。因为第二阶段非常耗时，如果第一阶段挖掘出来的候选项集太多，会使得算法的表现很差。

文献[5-7,21]中都进行了 top-k 的项集挖掘，它们虽然采用的数据结构不同，但是都采用了一个列表 L 存取 top-k 的 itemsets，并用 L 中最小的值来更新当前的阈值。

3 挖掘频繁和高效用的项集

FHIMA 算法主要有两步，在第一步中，通过扫描原有的数据库，将那些没有可能的项(即这些项的所有超集都不可能是频繁或者是高效用的)从数据库中删掉；在第二步中，使用 PrefixSpan 算法的思想来避免在挖掘高质量项集的过程中，非频繁的候选项集的产生。同时，提出了计算相对效用值和质量值上界的方法，利用上界的性质，能够进一步减小搜索的空间，提高算法的效率。

3.1 将事务数据库进行转换

为了能够得到转换之后的数据库，需要对原有的数据库进行 3 次扫描。在第一次对数据库进行扫描时，计算所有的 1-项集的支持度。那些非频繁的项在接下来的挖掘过程中将不再被考虑。例如，给定 $\alpha=0.4$，在扫描完表 2 后，非频繁的项 f 在接下来的处理过程中不再被考虑。

在第二次扫描中，计算所有的 1-项集的事务加权效用值 TWU 占 D 的总效用值的比例。那些比值小于最小相对效用阈值 β 的项，在以后的挖掘中也不再被考虑。例如，在第二次扫描后得到 $TWU(\{e\})=15$，$TU(D)=128$。假定 $\beta=0.2$，因为 $TWU(\{e\})/TU(D)<\beta$，根据性质 1，$\{e\}$ 的所有超集都不可能是高效用项集，所以项 e 在接下来的过程中将不再被考虑。表 3 列出了在不考虑项 f 后所有 1-项集的 TWU。

表 3 1-项集的 TWU

Itemset	{a}	{b}	{c}	{d}	{e}
TWU	44	39	47	45	15

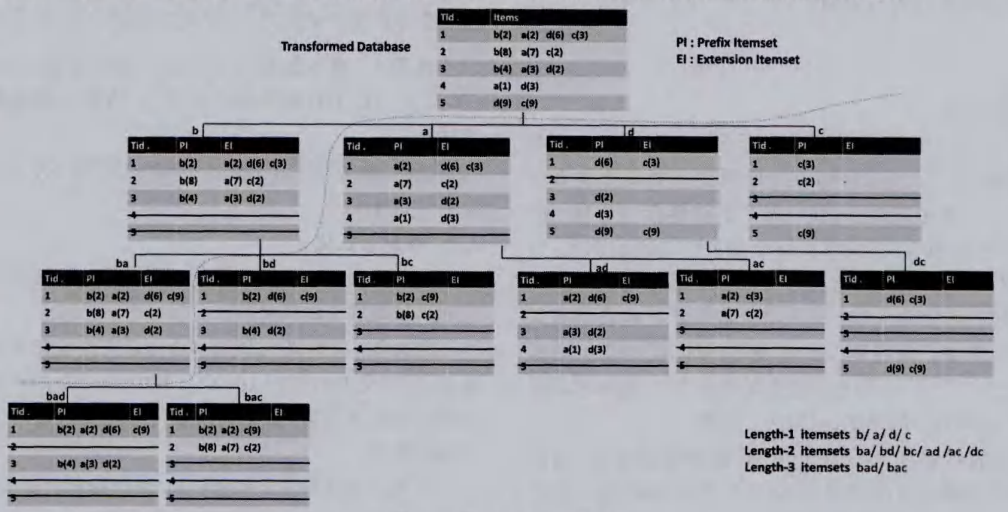
最后，将那些有可能的项按照 TWU 值递增的顺序进行排序。文献[12]中的结果表明，将项按照 TWU 递增的顺序排序比按照 TWU 递减的顺序排序在挖掘速度上几乎快上 3 倍。在不考虑项 f 和 e 后，按照 b,a,d,c 的顺序将项进行排序。表 4 是将表 2 进行转化后的事务数据库。

表 4 转化后的事务数据库

t_{id}	Item(util.)
t_1	b(2) a(2) d(6) c(3)
t_2	b(8) a(7) c(2)
t_3	b(4) a(3) d(2)
t_4	a(1) d(3)
t_5	d(9) c(9)

3.2 使用不生成非频繁候选项集的挖掘算法

解决这个问题最简单的方法是计算出所有项集的支持度和相对效用值，然后计算出相应的质量值，再从中选出 top-k 个高质量项集。但是，一个包含 k 个项的数据集可能产生出 2^k 个可能的项集，使得搜索空间呈指数级增长；另外，对这些项集计算其支持度和相对效用值也非常耗时。这里将利用频繁度和效用值的性质来减小搜索空间。



其中 $\alpha=0.4$，$\beta=0.2$，$\lambda=2$，$k=1$

图 1 FHIMA 算法的框架

本文提出的 FHIMA 算法和 PrefixSpan[15] 算法具有相似之处，如图 1 所示，每个结点由支持事务的 id(Tid)、前缀项集(PI)以及扩展项集(EI) 3 部分组成。给定结点 $N(N=\{i_{n_1}, i_{n_2}, \cdots, i_{n_m}\})$，设 $t_{id}\in T_N$，则 t_{id} 所对应的前缀项集 $PI(N,t_{id})$ 为 $\{i_{n_1}, i_{n_2}, \cdots, i_{n_m}\}$，$t_{id}$ 相对应的扩展项集由 t_{id} 中 i_{n_m} 项之后的所有项构成，记为 $EI(N,t_{id})$。首先扫描转换过后的数据库，提取不同的项作为不同的前缀项集，同时计算出每个前缀项集对应的支持事务和扩展项集。如图 1 所示，在第一次扫

描后，长度为1的项集$\{b\}$、$\{a\}$、$\{d\}$和$\{c\}$能够被找到。一旦找到了长度为k的前缀项集，就能够递归地用扩展项集中频繁的项来生成长度为$(k+1)$的前缀项集。整棵搜索树持续扩展，直到扩展项集中没有项或者没有频繁的项为止。比如，我们找到了长度为1的项集$\{a\}$，然后用$\{a\}$的扩展项集中频繁的项d、c来生成长度为2的项集$\{ad\}$和$\{ac\}$。因为在$\{ad\}$的扩展项集中没有频繁的项了，所以长度为2的项集$\{ad\}$停止扩展。前缀不同的项集都被分到不同的子树中，所以整个过程可以使用分治方法。通过这种方式，树中所有的结点都是频繁的，没有产生非频繁的候选项集。接下来将展示如何通过效用值和质量值上界的性质来进一步减小搜索空间。具体来说，给定一个结点，我们能有效地计算出以该结点为根结点的子树的相对效用值和质量值上界。

算法1展示了使用深度优先来挖掘top-k的高质量的项集。我们用列表L来存储当前找到的k个结点，L中k个结点最小的质量值用q^*来表示。首先检查当前结点X是否为有质量的结点，因为PrefixSpan算法仅生成频繁的候选项集，因此仅需要检查X的相对效用值，如果X是有质量的结点，那么需要将X的质量值和q^*进行比较，再决定是否更新L。然后再递归地检查以X的孩子结点为根结点的子树。若X相对效用值低于给定的阈值，则通过函数getQualUpperBound来计算以X为根结点的子树的质量值上界。若求得的上界值不大于q^*，则整棵子树都将被剪掉；反之，递归地检查以X的孩子结点为根结点的子树。我们将在下一节讨论如何快速计算出子树的质量上界值。

算法1 FHIMA

```
Input: the current node currentNode, the list L to maintain the top-k
qualified itemsets searched so far.
1. leastNode←the node with least quality in L
2. if currentNode.util/TU(D)≥β then
3.   if currentNode.quality>leastNode.quality then
4.     replace currentNode with leastNode in L
5.     leastNode←the node with least quality in L
6.     for node∈currentNode.children do
7.       FHIMA(node,L)
8.   else
9.     if qual_ub>leastNode.quality then
10.      for node∈currentNode.children do
11.        FHIMA(node,L)
12.    end if
13. end if
14. else
15.   qual_ub←getQualUpperBound(currentNode)
16.   if qual_ub>leastNode.quality then
17.     for node∈currentNode.children do
18.       FHIMA(node,L)
19.   end if
20. end for
```

3.3 质量的上界

本节将展示如何有效地计算出质量值的上界。表5列出了这一节中使用的一些符号。

表5 3.3节中使用的部分符号

X	当前子树的根结点
X′	当前子树的任一结点
$PI(X,t_d)$	结点X在支持事务t_d中的前缀项集
$EI(X,t_d)$	结点X在支持事务t_d中的扩展项集
$EL(X)$	结点X的所有扩展项集的长度
$eiu(X,i_p)$	项i_p在X的扩展项集上的效用值
$EIU(X)$	一个记录所有$eiu(X,i_p)$值的向量，其中$i_p\in PI(x,t_d)$，$t_d\in T_x$
$sel(X)$	结点X的最小可扩展长度
$lel(X)$	结点X的最大可扩展长度
$V^{\downarrow}$	向量V按照递减的顺序排列

这里使用$EL(X)$来记录X的所有扩展项集的长度。表6列出结点$\{a\}$的所有扩展项集的长度。

表6 结点$\{a\}$

t_{id}	Transaction	PI	EI	EL	$EL^{\downarrow}$
t_1	b(2)a(2)d(6)c(3)	a(2)	d(6)c(3)	2	2
t_2	b(8)a(7)c(2)	a(7)	c(2)	1	1
t_3	b(4)a(3)d(2)	a(3)	d(2)	1	1
t_4	a(1)d(3)	a(1)	d(3)	1	1

定义10 给定前缀X，项i_p在X的扩展项集上的效用值记为$eiu(X,i_p)$，定义为$\sum_{i_p\in PI(X,t_d)} u(i_p,t_d)$。

从表6中我们可以看到，项c和项d在结点$\{a\}$的扩展项集中，因为$c\in PI(\{a\},t_1)$且$c\in PI(\{a\},t_2)$，所以可以计算出$eiu(\{a\},c)=3+2=5$，同理可得$eiu(\{a\},d)=6+2+3=11$。用向量$EIU(X)$记录所有$eiu(X,i_p)$值，因此$EIU(\{a\})$中有两个值，即11和5。对一个向量V(V可能是EL或者是EIU)，$V^{\downarrow}$表示将向量V中的所有元素按照递减的顺序排列。因此$EIU^{\downarrow}(1)$是EIU中最大的元素；$EL^{\downarrow}(i)$则是EL中第i大的值。

给定子树根结点X，X'表示当前子树的任一结点，其中X'是将X的扩展项集中的项添加到X后形成的新的项集。其中扩展长度$v(|X'|-|X|)$表示从X的扩展项集向X添加的项的个数。接下来将讨论在给定X后，扩展长度v的范围。

定义11(最小有效扩展，SVE)[17] 给定子树根结点X，X的最小有效扩展长度记为$sel(X)$，定义为$\lceil(\beta * TU(D)-u(X))/EIU^{\downarrow}(1)\rceil$。

例如，$TU(D)=126$，$\beta=0.2$，$u(\{a\})=13$，以$\{a\}$为根结点的子树中最小有效扩展长度为$\lceil(0.2*126-13)/11\rceil=2$。换句话来说，如果只从$\{a\}$的扩展项集中取出1个项加入到$\{a\}$，形成新的项集$X'$，那么$X'$一定不是高效用项集。更多的细节可以参考文献[17]。

参照SVE的性质，提出了一个新的性质，来限制X的最大扩展长度，称作最大有效扩展长度。

定义12(最大有效扩展，LVE) 给定子树根结点X，X的最大有效扩展长度记为$lel(X)$，定义为$EL^{\downarrow}(\lceil|D|*\alpha\rceil)$。

例如，$\alpha=0.2$，$|D|=5$，$EL^{\downarrow}=\langle 2,1,1,1\rangle$，$\{a\}$的最大有效扩展长度为$EL^{\downarrow}(2)=1$。即，如果我们从$\{a\}$的扩展项集中取出2个或2个以上的项加入到$\{a\}$，形成新的项集$X'$，那么$X'$一定不是频繁的。

定义11和定义12给出了X的扩展长度的范围，如果$lel(X)<sel(X)$，那么以X为根结点的整棵子树都不可能存在有质量的项集，整棵子树都应该被剪掉。以$\{a\}$为根结点的

子树就是这种情况，所以整棵子树都应该被剪掉。若$lex(X)>sel(X)$，则通过计算相对效用值和质量值的上界来进行剪枝。

给定根结点 X，由于支持度的反单调性，整棵子树支持度的上界为 X 的支持度，即 $fre(X)/|D|$。Liu[12]提出了一个计算效用值上界的方法，即将 X 的效用值和所有项在 X 的扩展项集上的效用值相加。由效用值的上界，能很快得到相对效用值的上界，如下式所示：

$$\frac{\sum_{t_d\in T_X} u(X,t)+\sum_{i_p\in PI(X,t_d),t_d\in T_X} eiu(X,i_p)}{TU(D)} \quad (1)$$

因为质量值是支持度和相对效用值的加权和，由支持度和相对效用值的上界，能很快地得到质量值的一个比较松的上界。

$$\frac{fre(X)}{|D|}+\lambda\left(\frac{\sum_{t_d\in T_X} u(X,t)+\sum_{i_p\in PI(X,t_d),t_d\in T_X} eiu(X,i_p)}{TU(D)}\right) \quad (2)$$

以表 7 的结点$\{d\}$为例，$\{d\}$的支持度为 4/5=0.8。$\{d\}$的效用值为(6+2+3+9)=20，项 c 在$\{d\}$的扩展项集中，可得到 $eiu(\{d\},c)=12$。因为 $TU(D)=126$，得到其相对效用值的上界为(20+12)/126=0.254，设 $\lambda=2$，得到其质量值的上界为(0.8+2*0.254)=1.308。

表 7　结点$\{d\}$

t_{id}	Transaction	PI	EI	EL	EL↓
t_1	b(2)a(2)d(6)c(3)	d(6)	c(3)	1	1
t_2	b(8)a(7)c(2)	d(2)		0	1
t_3	b(4)a(3)d(2)	d(3)		0	0
t_4	d(9)c(9)	d(9)	c(9)	1	0

上面计算得到的质量值上界很松。换句话说，在挖掘的过程中，仍然有许多候选结点生成。这里将提出一个更紧的计算质量值上界的方法。

给定根结点 X、扩展长度 v，其中 $sel(X)\leqslant v\leqslant lel(X)$，则扩展后的项集 X'的支持事务 $T_{X'}$ 的上界是 X 的扩展项集中长度大于或等于 v 的事务，即满足 $EL(t_d)\geqslant v$，其中 $t_d\in T_X$。而 X'的效用值的上界则分为两部分，第一部分是项集 X 在 $T_{X'}$ 的效用值，第二部分则是新加入的项贡献的效用值。而新加入的项在 $T_{X'}$ 的效用值定小于或等于 $EIU^{\downarrow}$ 中前 k 个最大值之和。最终得到其相对效用值上界如下：

$$\frac{\sum_{t_d\in T_X,EL(t_d)\geqslant v} u(X,t_d)+\sum_{j=1}^{v} EIU^{\downarrow}(j)}{TU(D)} \quad (3)$$

证明：设 $T_{X'}$ 为 X'的支持事务，X^* 为新加入项的集合，$u(X')=\sum_{t_d\in T_{X'}} u(X,t_d)+\sum_{t_d\in T_{X'}} u(X^*,t_d)$，因为 $T_{X'}\subseteq\{t_d\mid EL(t_d)\geqslant v,t_d\in T_X\}$，所以 $\sum_{t_d\in T_{X'}} u(X,t_d)\leqslant\sum_{t_d\in T_{X'},EL(t_d)\geqslant v} u(X,t_d)$；又 $\sum_{t_d\in T_{X'}} u(X^*,t_d)\leqslant\sum_{i_p\in X^*} eiu(X,i_p)\leqslant\sum_{j=1}^{v} EIU^{\downarrow}(j)$。将上述两个不等式相加，即得证。

对于每个指定的 v，由其支持事务的上界易得到其支持度的上界为$|\{t_d\mid EL(t_d)\geqslant v,t_d\in T_X\}|$。对于每个 v，我们都能得到其对应的质量值的上界，通过枚举所有的 v，可得到最终质量值的上界为

$$\max_{sel(X)\leqslant v\leqslant lel(X)}\left(\frac{|\{t_d\mid EL(t_d)\geqslant v,t_d\in T_X\}|}{|D|}+\lambda\frac{\sum_{EL(t_d)\geqslant v,t_d\in T_X} u(X,t_d)+\sum_{j=1}^{v} EIU^{\downarrow}(j)}{TU(D)}\right) \quad (4)$$

例如，在表 7 中，我们计算得到 $sel(\{d\})=lel(\{d\})=1$，v 的可扩展长度为 1。其中只有 t_1 和 t_5 满足 $EL(t_d)\geqslant v$，所以其支持度上界为 2/5=0.4。$\sum_{EL(t_d)\geqslant 1} u(\{d\},t_d)=6+9=15$，$\sum_{j=1}^{1} EIU^{\downarrow}(1)=12$。所以其相对效用值为(12+15)/126=0.214，其质量值上界为(0.4+2*0.214)=0.828，比由式(2)得到的 1.308 小了很多。

这里提出了两种计算质量上界值的方法，可以看到采用根结点的支持度和文献[12]采用的计算效用值的方法结合起来计算上界，得到的上界值很松。下一节的实验将在不同的数据集上对两种计算上界的方法进行比较。

4　实验评估

这一节将验证 FHIMA 通过用上界值进行剪枝带来的高效率，同时还对两种上界值的策略进行比较。

4.1　实验设置

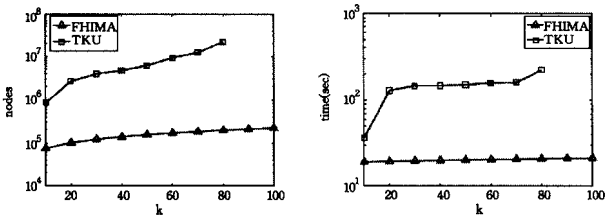
在这个实验中，一共使用了 4 个数据集，包括真实数据集 accident、chess 和 mushroom，以及由 IBM 模拟数据生成器生成的模拟数据集 T40I10D100K。所有的数据集均从 FIMI 库中下载[24]。使用类似于文献[2,12]的评估方法，项的利润值是通过对数正态分布生成的 1 到 1000 之间的数。项在不同事务中的数量是通过随机分布生成的 1 到 10 之间的数。

因为没有其它的算法同时将支持度和相对效用值作为衡量标准，所以没有其它的基准算法可以直接与 FHIMA 进行比较。但是可以通过对参数进行设置，来与已有的算法进行比较。例如，通过令 $\alpha=0,\beta=0$，和 $\lambda=+\infty$，我们将挖掘 top-k 的有质量的项集的问题转化成挖掘 top-k 的高效用的项集，用最近提出的用来挖掘 top-k 的高效用的项集的 TKU[21]算法作为基准算法。TKU 算法分为两步，第一步找出所有潜在的高效用项集，第二步是计算出所有潜在的高效用项集的真实效用值。TKU 算法需要维护大量的候选项集，并且需要计算候选项集的真实效用值；而 FHIMA 算法采用上界剪枝的思想，所以在效率上好于 TKU 算法。

所有的算法都是用 C++实现的，运行在一个 3.3GHz (Intel core i3-2120)，内存 8G，操作系统为 Windows 7 的个人电脑上。

4.2　FHIMA 和 TKU 的比较

图 2 给出了 FHIMA 和 TKU 在不同 k 值上搜索时间和搜索结点数量的比较结果。当 k 大于等于 20 时，FHIMA 算法差不多比 TKU 算法快了一个量级，同时 FHIMA 算法搜索结点的数目差不多比 TKU 少了两个量级。当 k 很小时，潜在的候选项集很小，TKU 在第二步中花费的时间很少，同时 TKU 采用了 SE 策略来进一步减少在第二步中所要检查项集的个数。因此在 k 很小时，两种算法的搜索时间很相近。随着 k 的增加，FHIMA 算法搜索时间和搜索结点数的增加非常缓和。例如，当 k 从 20 增加到 80 时，FHIMA 的搜索时间从 19.4s 增加到了 20.8s，而 TKU 算法的搜索时间从 128.3s增加到了 219.8s。



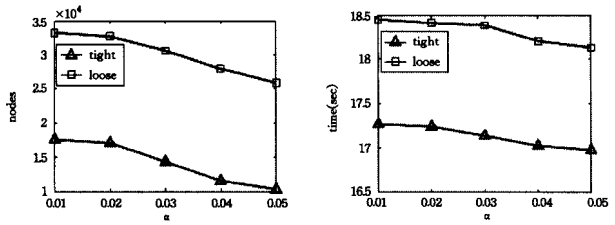
(a)结点数量　　(b)搜索时间

图 2　mushroom 上 k 不同时的搜索结点和搜索时间

4.3　不同参数设置上的衡量

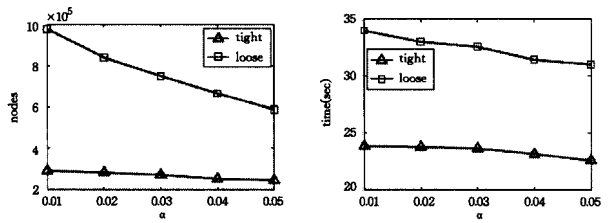
在这部分，为了测试提出的相对效用值和质量值上界的效率，将提出的方法（见式(4)）记为紧的上界，基准算法用文献[12]提出的计算相对效用值的方法（见式(2)）记为松的上界。

支持度阈值。支持度的阈值越小，挖掘到的项集的数量就越多，将会导致搜索时间和搜索结点数量的增长。如图 3—图 6 所示，与松的上界相比，紧的上界搜索的结点和花费的时间更少。另外，因为依据相对效用值和质量值上界来剪枝，所以在支持度阈值很小时，FHIMA 算法也能保持高效。



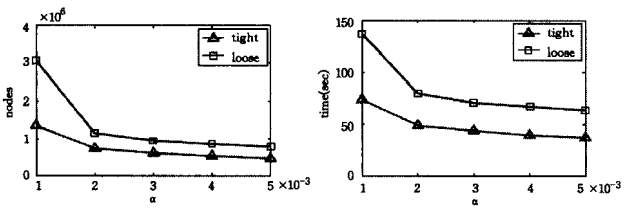
(a)结点数量　　(b)搜索时间

图 3　mushroom 上 α 不同时的搜索结点和搜索时间



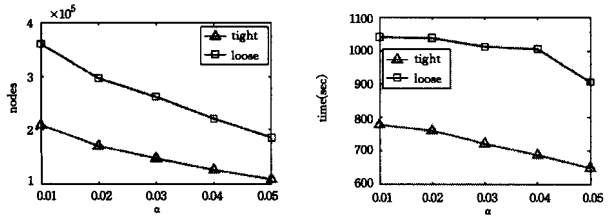
(a)结点数量　　(b)搜索时间

图 4　chess 上 α 不同时的搜索结点和搜索时间



(a)结点数量　　(b)搜索时间

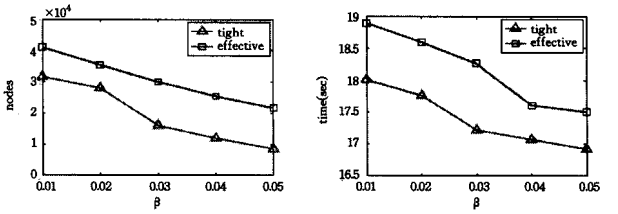
图 5　T40I10D100K 上 α 不同时的搜索结点和搜索时间



(a)结点数量　　(b)搜索时间

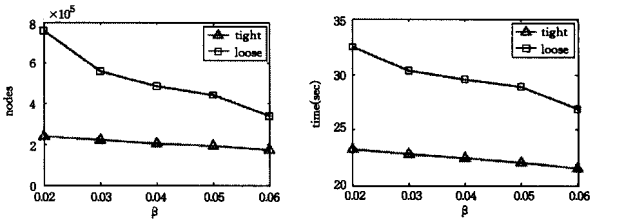
图 6　accident 上 α 不同时的搜索结点和搜索时间

相对效用值阈值。如图 7—图 10 所示，紧的上界值在效率上始终超过松的上界值。因为本文采用 top-k 高质量项集挖掘，列表 L 中存储的当前第 k 大的结点，能有效减小搜索空间。即使相对效用阈值设得很低，FHIMA 仍能够保持很高的效率。



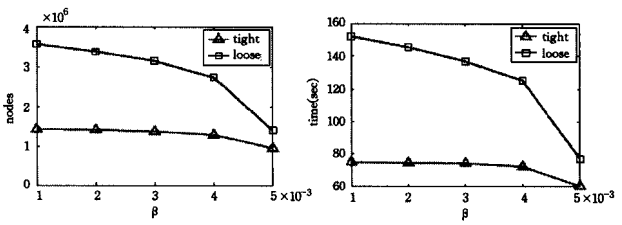
(a)结点数量　　(b)搜索时间

图 7　mushroom 上 β 不同时的搜索结点和搜索时间



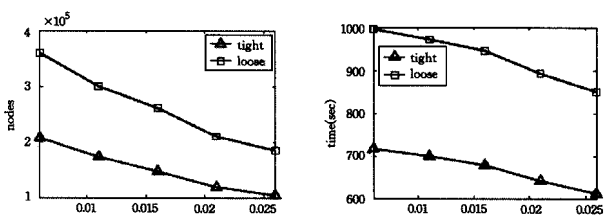
(a)结点数量　　(b)搜索时间

图 8　chess 上 β 不同时的搜索结点和搜索时间



(a)结点数量　　(b)搜索时间

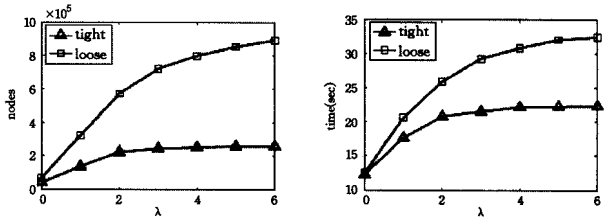
图 9　T40I10D100K 上 β 不同时的搜索结点和搜索时间



(a)结点数量　　(b)搜索时间

图 10　accident 上 β 不同时的搜索结点和搜索时间

相对效用权重参数。相对效用权重 λ 反映了相对效用值的重要度。从图 11 中可以看出，当把 λ 从 0 增加到 3 时，紧的上界的搜索时间从 12.3s 增加到了 21.6s。当 $\lambda \geqslant 3$ 时，搜索时间基本上收敛了，因为这时相对效用值已经在质量值中起主导作用，再继续增加 λ 对 FHIMA 的影响已经很小了。



(a)结点数量　　(b)搜索时间

图 11　chess 上 λ 不同时的搜索结点和搜索时间

（下转第 123 页）

Canada,2008

[7] Li Wen-bo,Sun Le,Zhang Da-kun. Text classification based on labeled-LDA model[J]. Chinese Journal of Computers,2008,31(4):620-627

[8] 江雨燕,李平,王清. 基于共享背景主题的 LabeledLDA 模型[J]. 电子学报,2013,41(9):1794-1799

[9] Ville H T,Henry T. Combining Topic Models and Social Networks for Chat Data Mining[C]// IEEE/WIC/ACM International Conference on Web Intelligence. Los Alamitos, USA: IEEE Computer Society Press,2004:206-213

[10] Tan Xu, Douglas W O. Wikipedia-based Topic Clustering for Microblogs[J]. American Society for Information Science and Technology,2011,48(1):1-10

[11] Wagstaff K,Cardie C. Clustering with instance-level constraints [C]// Proceedings of the 17rd international conference on Machine learning. Morgan Kaufmann,2000:1103-1110

[12] Brian K, Sugato B, Inderjit S D, et al. Semi-supervised graph clustering: a kernel approach[J]. Machine Learning, 2009,74(1):1-22

[13] Kass R,Wasserman L. A reference Bayesian test for nested hypotheses and its relationship to the Schwarzcriterion[J]. Journal of the American Statistical Association,1995(10):928-934

[14] 郑苗苗,吉根林. 一种基于密度的分布式聚类算法[J]. 南京大学学报,2008,44(5):536-543

[15] 刘群,李素建. 基于《知网》的词汇语义相似度计算[C]//第三届汉语词汇语义学研讨会. 台北,2002

(上接第 87 页)

结束语 本文中,我们将以前独立的两种衡量方式综合考虑,希望找到那些高质量即既频繁又高效用的项集。另外,我们将问题转化为 top-k 的频繁和高效用项集的挖掘。由于效用值和质量值都既不具有单调性也不具有反单调性,因此提出了利用相对效用值和质量值上界来剪枝的 FHIMA 算法。同时,FHIMA 利用了 Prefixspan 算法的思想,避免了非频繁候选项集的产生。实验结果证明,FHIMA 算法通过利用上界值剪枝能大大提高算法的效率,同时紧的上界值在效率上要优于松的上界值。

参 考 文 献

[1] Agrawl R, Srikant R. Fast algorithms for mining association rules[C]// VLDB. 1994:487-499

[2] Ahmed C F,Tanbeer S K,Jeong B-S,et al. Efficient tree structures for high utility pattern mining in incremental databases[C]// TKDE. 2009:1708-1721

[3] Ahmed C F,Tanbeer S K,Jeong B-S,et al. Efficient tree structures for high utility pattern mining in incremental databases [J]. TKDE,2009,21(12):1708-1721

[4] Chan R, Yang Q, Shen Y. Ming high-utility itemsets[C]// ICDM. 2003:19-26

[5] Cheung Y L,Fu A W. Mining frequent itemsets without support threshold:with and without item constraints[C]// TKDE. 2004: 1052-1069

[6] Chuang K,Huang J,Chen M. Mining top-k frequent patterns in the presence of the memory constraint[J]. VLDB Journal,2008, 17:1321-1488

[7] Fu A W,Kwong R W,Tang J. Mining n-most interesting itemsets[C]// ISMIS. 2000:59-67

[8] Gade K, Wang J, Karypis G. Efficient closed pattern mining in the presence of tough block constraints[C]// KDD. ACM,2004: 138-147

[9] Han J,Pei J,Yin Y. Mining frequent patterns without candidate generation[C]// SIGMOD. 2000:1-12

[10] Han J, Wang J, Liu Y, et al. Mining top-k frequent closed patterns without minimum support[C]// ICDM. 2002:211-218

[11] Li H F,Huang H Y,Chen Y C,et al. Fast and memory efficient mining of high utility itemsets in data streams[C]// ICDM. 2008:881-886

[12] Liu M,Qu J. Mining high utility itemsets without candidate generation[C]// CIKM. ACM,2012:55-64

[13] Liu Y,Liao W K,Choudhary A. A fast high utility itemsets mining algorithm[C]// Workshop on WUDM. 2005:90-99

[14] Lucchese C,Orlando S,Palmerini P,et al. KDCI:A multi-strategy algorithm for mining frequent itemsets[C]// ICDM Workshop FIMI. 2003:372-390

[15] Pei J,Han J,Mortazaviasl B,et al. Prefixspan:Mining sequential patterns efficiently by prefix-projected pattern growth[C]// ICDE. 2001:215-224

[16] Savasere A,Omiecinski E R,Navathe S B. An efficient algorithm for mining association rules in large databases[C]// VLDB. 1995:432-443

[17] Seno M,Karypis G. Lpminer:An algorithm for finding frequent itemsets using length-decreasing support constraint[C]// ICDM. 2011:505-512

[18] Tseng V S,Chu C J,Liang T. Efficient mining of temporal high-utility itemsets from data streams[C]// KDDWorkshop on UBDM. 2006:18

[19] Tseng V S,Wu C W,Shie B E,et al. Up-growth:an efficient algorithm for high utility itemset mining[C]// KDD. 2010:253-262

[20] Vo B,Nguyen H,Ho T B,et al. Parallel method for mining high utility itemsets from vertically partitioned distributed databases [C]// KES 2009. 2009:251-260

[21] Wu C,Shie B,Tseng V S,et al. Mining top-k high utility itemsets[C]// KDD. 2012:78-86

[22] Yao H, Hamilton H J, Buts C J. A foundational approach to mining itemset utilities from databases[C]// SDM. 2004:251-260

[23] Zaki M. Scalable algorithms for association mining [J]. Knowledge and Data Engineering,2000,12(2):372-390

[24] Frequent itemset mining dataset repository,2012[OL]. http://fimi.ua.ac.be