基于索引树的带通配符序列模式挖掘算法

王 乐[1),2)] 王 水[1)] 刘胜蓝[2)] 王辉兵[2)]

[1)](宁波财经学院信息工程学院 浙江 宁波 315175)

[2)](大连理工大学创新创业学院 辽宁 大连 116024)

摘 要 随着有序时间序列数据的出现,序列模式挖掘成为数据挖掘领域的一个分支.其中带通配符的序列模式挖掘又是该领域中一个重要的研究问题,同时随着数据规模越来越大,算法的挖掘效率尤为重要.现有算法多采用树型结构来实现数据的压缩表示,树的结构和模式匹配方法对挖掘效率有决定性的影响.该文首先设计一个新的树结构索引树 I-Tree(Index-Tree)来维护原始序列数据以及序列模式和模式索引信息;然后在索引树的基础上,提出一个新的带通配符的序列模式挖掘算法 ITM(Index-Tree based sequential pattern Mining).算法 ITM 主要用 4 个策略提高算法的挖掘效率:(1)将原始序列中相同项压缩到一个节点上,该节点只记录项在原始序列中的索引;(2)采用迭代的方式,长度 $k+1$ 的序列模式是用长度 $k(k>0)$ 的候选序列模式产生;(3)采用前缀树的结构,逐层将 $k+1$ 的候选序列模式压缩到索引树上,叶子节点上记录序列模式最后一项的索引;(4)整个挖掘过程,只用一棵索引树.算法 ITM 通过采用以上索引树压缩原始序列数据以及存储候选序列模式,有效地缩小搜索空间,从而算法效率得到显著提升.另一种提高挖掘效率的思路,是在挖掘过程中允许有小部分的模式丢失,来换取挖掘效率的大幅度提升,即所谓的近似模式挖掘.该文也给出了一个近似序列模式挖掘算法 AITM(Approximate Index-Tree based sequential pattern Mining),该近似算法通过估计超序列模式的支持数,将非候选节点提前删掉,减少索引树上的节点个数,从而提高算法的时空效率;但是也因为估计的支持数可能会小于实际值,从而丢失了部分频繁的序列模式.该文实验中,提出的两个算法分别与算法 MGCS、MAPB 和 MAPD 进行了对比实验,采用 3 个典型数据序列进行测试,并设计了 3 组实验:(1)不同的最小支持度对算法的效率影响;(2)算法的扩展性;(3)通配符长度对算法效率的影响.实验结果验证了该文提出算法的有效性,时空效率得到一定的提高;针对不同的阈值,最小支持度越小、原始序列长度越长、通配符长度越长,算法的时间效率提高幅度越大;同时近似挖掘算法的精确度接近 100%.

关键词 数据挖掘;序列模式;通配符;模式匹配;索引树

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2019.00554

An Algorithm of Mining Sequential Pattern with Wildcards Based on Index-Tree

WANG Le[1),2)] WANG Shui[1)] LIU Sheng-Lan[2)] WANG Hui-Bing[2)]

[1)](*School of Information Engineering, Ningbo University of Finance & Economics, Ningbo, Zhejiang* 315175)

[2)](*School of Innovation and Entrepreneurship, Dalian University of Technology, Dalian, Liaoning* 116024)

Abstract Sequential Pattern Mining, which arose as a sub-field of data mining, focuses on sequences of items or events occurring in an ordered metric space. Mining sequential pattern with wildcards is one important issue in this field, and for higher volume data, more efficient algorithms are needed. Most of the previous algorithms generally adopt tree structures for compressing data and searching for sequential patterns, therefore the structural design and the pattern matching methodology affect mining efficiency significantly. In this paper, we first propose a novel tree structure Index-Tree(I-Tree), which stores compressed original data series and sequential

收稿日期:2016-04-05;在线出版日期:2016-11-29.本课题得到国家自然科学基金项目(1120165702)、中国博士后面上基金项目(ZX20150629)、浙江省科技厅计划项目(2016C31128,2017C35014)、宁波市自然科学基金项目(2015A610135)、2015 年度浙江省统计研究重点课题(201522)资助.**王 乐**,女,1978 年生,博士,讲师,主要研究方向为金融数据挖掘.E-mail:wangleboro@163.com.**王 水**(通信作者),男,1967 年生,教授,主要研究领域为数据挖掘.E-mail:seawan@163.com.**刘胜蓝**,男,1984 年生,博士,主要研究领域为计算机视觉、机器学习.**王辉兵**,男,1989 年生,博士研究生,主要研究方向为计算机视觉、机器学习.

patterns as well as pattern indexing information; second, we develop an efficient algorithm ITM (Index-Tree based sequential pattern Mining) based on index-tree for mining complete set of frequent sequential patterns. Efficiency of the proposed algorithm ITM is improved by four techniques: (1) the same candidate items of the original data series are compressed to the same node that records index of items, (2) sequential patterns of length $k+1$ are generated by scanning candidate sequential patterns of length k ($k>0$) and their index information on the index-tree, (3) adopting the prefix-tree structure, candidate sequential pattern of length $k+1$ are stored at the same index-tree, and leaf-nodes record the index of last item of the candidates, (4) the algorithm ITM creates one index-tree in the whole mining process. The index-tree reduces search space effectively by compressing the original data series & candidate sequential patterns, and by recording index of the last item of sequential patterns, therefore the performance of algorithm ITM is improved significantly. To achieve higher mining performance, we discuss another approach that allows a small portion of sequential patterns to be omitted during the mining process in exchange for a substantial increase of mining efficiency, this kind of approach is what we call approximate algorithm. In this paper, we develop an approximate algorithm named AITM (Approximate Index-Tree based sequential pattern Mining) based on the algorithm ITM; by estimating the support number of super sequential pattern, non-candidate nodes are deleted prematurely to reduce the number of nodes on index-tree, and therefore to reduce the search space of the mining process and promote efficiency of pattern matching; the algorithm AITM may lose some sequential patterns because the estimated support number of patterns may be less than their actual values, but the space-time efficiency of algorithm AITM is much better than that of algorithm ITM. Our experimental results show that the proposed algorithms are efficient and scalable for mining sequential pattern with varied parameter on different data series. Three classical datasets are used in our experiments for comparing with the state-of-art algorithms MGCS, MAPB and MAPD; 3 different experiments are designed: (1) under varied minimum support threshold, (2) under varied volume of data, and (3) under varied length of wildcards; the experimental results verify the validity of the proposed algorithms with increase of time/space efficiency; the smaller the minimum support is, or the longer the length of the original data series or the wildcards is the more significant boosting on performance the proposed algorithms would have; and the accuracy of approximate algorithm AITM is up to 100%.

Keywords　data mining; sequential pattern; wildcard gaps; pattern matching; index-tree

1　引　言

模式挖掘是数据挖掘领域一种重要和基础的技术，由最初的传统频繁模式[1-3](frequent pattern)挖掘不断扩展到不确定数据集(uncertain dataset)中的频繁模式挖掘[4-7]、带权重的频繁模式挖掘[8]、高效用模式(high utility pattern)挖掘[9-12]、大数据下的频繁模式挖掘[13-14]以及序列模式(sequential pattern)挖掘[15-16]等.

序列模式挖掘最早由 Agrawal 和 Srikant 提出[17]，其任务是从序列数据集中发现所有频繁的模式.序列模式挖掘广泛出现在股票分析及走势预测，DNA 序列分析、疾病预测、大型购物数据分析等应用领域中[18-19].如何在序列数据中找到频繁出现的序列模式，是序列模式挖掘的主要任务之一[15-16,20-24].

Wu 等人[15]指出带有通配符的序列模式挖掘具有重要的研究价值，它允许挖掘出序列模式中含有灵活的通配符，不仅使序列模式挖掘具有理论研究价值，也使其具有巨大的应用价值，如在文本索引和生物信息学等领域.

但是序列模式中引入通配符后，使序列模式挖掘问题变得更为复杂，特别是随着数据规模的增加，挖掘算法的效率的提高是序列模式挖掘的一个重要

研究问题[15-16,20-22]，另外序列模式挖掘中的一个重要工作是模式匹配，此后，Wu 等人针对模式匹配问题进行了研究，同时也给出了有效的模式匹配算法[25-26].

在已有序列模式挖掘研究成果的基础上，本文首先给出一个树结构来存储原始序列索引信息和序列模式的索引信息，该树结构命名为索引树(I-Tree，Index-tree)；然后给出一个基于 I-Tree 的序列模式挖掘算法 ITM(I-Tree based sequential pattern Mining). 算法 ITM 通过将序列中各项的索引存放到一棵索引树 I-Tree 上，然后访问树上相应索引进行序列模式的匹配，不需要再访问原数据集，同时通过索引进行模式匹配也缩小了搜索空间，从而可以有效地提高算法的挖掘效率. 在算法 ITM 的基础上，利用估计模式支持数进行剪枝的策略，本文又给出一个近似的序列模式挖掘算法 AITM(Approximate ITM).

本文第 2 节给出序列模式挖掘的相关定义和相关工作；第 3 节描述算法 ITM；第 4 节介绍一个近似算法 AITM；第 5 节对算法进行复杂度分析；第 6 节给出实验结果及其分析；第 7 节给出结论.

2 相关定义与相关工作

2.1 相关定义

定义 1. 在一个目标序列 $S=s_1s_2\cdots s_n$ 中，其包含的字符个数 n 为其序列长度，序列 S 的字符集(也称为字符表)合记为 Σ，S 中不同字符的个数记为 $|\Sigma|$.

例 1. 一个 DNA 序列片段 $S=s_1s_2s_3s_4s_5s_6s_7s_8s_9s_{10}s_{11}s_{12}s_{13}s_{14}s_{15}$ = gaattcatcagccat，其序列长度为 15，字符表 $\Sigma=\{a,c,g,t\}$，$|\Sigma|=4$.

定义 2. 通配符可以代表字符集中任意字符，记为 Φ.

定义 3. 设长度为 m 的一个模式 $P=p_1g_1p_2g_2\cdots p_j\cdots g_{m-1}p_m$，其中 $g_i(i=1,2,\cdots,m-1)$ 是字符 p_i 和 p_{i+1} 之间的通配符个数，g_i 的取值区间为 $[M,N]$(即 g_i 最小取值为 M，最大取值为 N)，这里模式 P 就称为一个带通配符的序列模式或非固定间隔约束的序列模式，这里简称为序列模式.

定义 4. 假设 S 中依次存在 m 个字符 $s_{l_1}s_{l_2}\cdots s_{l_m}$，其中字符 $s_{l_i}(i=1,2,\cdots,m)$ 的下标 l_i 为该字符在 S 中位置(或索引)，只要前后两个字符下标值的范围在 $[M,N]$，则序列 $s_{l_1}s_{l_2}\cdots s_{l_m}$ 是一个长度为 m 的偏移序列(offset sequence)，序列 S 中模式 P 的所有偏移序列就是所有长度为 $|P|$ 的偏移序列，其个数记为 $ofs(|P|,S)$.

例 2. 在目标序列 $S=s_1s_2s_3s_4s_5s_6s_7s_8s_9$ = gaattcatc 中，当间隔约束 $[M,N]$ 中的 $M=3$，$N=4$，则序列 s_1s_5 = gt、s_1s_6 = gc、s_2s_6 = ac、s_2s_7 = aa、s_3s_7 = aa、s_3s_8 = at、s_4s_8 = tt、s_4s_9 = tc、s_5s_9 = tc 是长度为 2 的偏移序列，$ofs(2,S)=9$；序列 $s_1s_5s_9$ = gtc 是长度为 3 的偏移序列，$ofs(3,S)=1$.

定义 5. $S_1=s_{l_1}s_{l_2}\cdots s_{l_m}$ 是 S 中一个偏移序列，$P=p_1g_1p_2g_2\cdots p_j\cdots g_{m-1}p_m$ 是一个长度为 m 的序列模式，若 $s_{l_i}=p_i(i=1,2,\cdots,m)$，则字符串 $S_1=s_{l_1}s_{l_2}\cdots s_{l_m}$ 为模式 P 在 S 中的一次出现；同样满足上述条件的字符串 $S_2=s_{t_1}s_{t_2}\cdots s_{t_m}$，只要 S_1 和 S_2 中存在一个 i 使 $t_i\neq l_i$，则 S_2 就称为是模式 P 的另一次出现，即 S_1 和 S_2 是模式 P 在目标序列 S 中的两次出现；模式 P 在 S 中所有出现次数称为模式 P 的支持数，记为 $sup(P,S)$.

定义 6. 假设 ρ 为用户预定义的最小支持度，若 $r(P,S)=sup(P,S)/ofs(|P|,S)\geq\rho$，则模式 P 称为一个频繁序列模式；否则，模式 P 为非频繁序列模式. 本文所研究的模式挖掘就是从序列中找出所有 $r(P,S)\geq\rho$ 的频繁序列模式.

定义 7. 给定长度分别为 m_1 和 m_2 的模式 P_1 和 P_2，如果 P_1 是 P_2 的一个子串，则称 P_2 是 P_1 的超模式，P_1 是 P_2 的子模式；如果 P_1 和 P_2 的前 m_2-1 个字符依次相同，则称 P_1 是 P_2 的前缀模式；如果 P_1 和 P_2 的后 m_2-1 个字符依次相同，则称 P_1 是 P_2 的后缀模式.

从定义 4 和 5 可知，$sup(P,S)\leq ofs(|P|,S)$，即 $0\leq r(P,S)\leq 1$. 设定模式 P 的长度为 m，序列 S 的长度为 n，$W=N-M+1$，$l_1=\lfloor(n+M)/(M+1)\rfloor$，$l_2=\lfloor(n+N)/(N+1)\rfloor$，根据文献[15]，目标序列 S 中长度为 m 的偏移序列个数 $ofs(m,S)$ 的计算方法如式(1)：

$$ofs(m,S)=\begin{cases}0, & m>l_1\\ (n-(m-1)((M+N)/2+1))W^{(m-1)}, & m\leq l_2\\ can\ be\ calculated\ by\ a\ recursive\ formular, & l_2<m\leq l_1\end{cases}\tag{1}$$

引理 1. 对任意长度为 m 的模式 P_1 和其长度为 $m+1$ 的超模式 P_2，可得

$sup(P_2,S)\leq sup(P_1,S)\times W$[15].

证明.　令 $S_1=s_{l_1}s_{l_2}\cdots s_{l_m}$ 是模式 P_1 在目标序列 S 中的一个出现，模式 P_1 为模式 P_2 的一个前缀模式，则 P_2 在序列 S 中至多有 W 个以 S_1 为前缀的超模式，所以 $sup(P_2,S)\leq sup(P_1,S)W$. 同样地，当 P_1 为模式 P_2 的后缀模式时，也可以得到相同的结论.　　证毕.

引理 2.　如果一个长度为 m 的序列模式 P 的支持率小于 $\dfrac{n-(d-1)(w+1)}{n-(m-1)(w+1)}\rho(d>m)$，那么它所有的超模式都是非频繁的，其中 d 表示最长的频繁模式长度，n 表示主序列 S 的长度，$w=(N+M)/2$[15].

证明.　根据引理 2 的条件可知 $\dfrac{sup(P,S)}{ofs(|P|,S)}<\dfrac{n-(d-1)(w+1)}{n-(m-1)(w+1)}\rho$.

假设模式 Q 是 P 的一个长度为 $k(m<k\leq d)$ 的超模式，根据偏移序列计算公式(1)可以推断出 $ofs(|P|,S)=(n-(m-1)(w+1))W^{m-1}$ 和 $ofs(|Q|,S)=(n-(k-1)(w+1))W^{k-1}$. 由引理 1 可得到 $sup(Q,S)\leq sup(P,S)W^{k-m}$，所以

$$\frac{sup(Q,S)}{ofs(|Q|,S)}\leq\frac{sup(P,S)}{ofs(|P|,S)}\cdot\frac{n-(m-1)(w+1)}{n-(k-1)(w+1)}\leq\frac{sup(P,S)}{ofs(|P|,S)}\cdot\frac{n-(m-1)(w+1)}{n-(d-1)(w+1)}<\rho.$$

由此引理 2 得证.　　证毕.

引理 3.　假设 P 是一个长度为 m 的序列模式、目标序列 S 中最长频繁序列模式的长度为 d、$\beta=\dfrac{n-(d-2)w}{n-(d-1)w}>1(d>m)$，如果 $r(P,S)\beta^{d-m}<\rho$，则模式 P 不存在超序列模式(超频繁序列模式).

证明.　假设 Q 是 P 的一个长度为 $k(m<k\leq d)$ 的超模式，则

$$\begin{aligned}r(Q,S)&=\frac{sup(Q,S)}{ofs(|Q|,S)}\leq\frac{sup(P,S)w^{k-m}}{n-(k-1)w^{k-1}}\\&=r(P,S)\cdot\frac{n-(m-1)w}{n-(k-1)w}\\&=r(P,S)\cdot\frac{n-(m-1)\cdot w}{n-m\cdot w}\cdot\frac{n-mw}{n-(m+1)w}\cdot\\&\quad\frac{n-(m+1)w}{n-(m+2)w}\cdots\frac{n-(k-2)w}{n-(k-1)w}\\&<r(P,S)\beta^{k-m}.\end{aligned}$$

其中，由于 $\alpha=\dfrac{n-(m-1)w}{n-mw}>1$，所以 $r(P,S)\beta<r(P,S)\beta^2<\cdots<r(P,S)\beta^{k-m}$. 因此可以推出当 $r(P,S)\beta^{d-m}<\rho$，模式 P 不存在超频繁序列模式.　　证毕.

定义 8.　一个长度为 m 的序列模式 P 的支持率小于 $\dfrac{n-(d-1)(w+1)}{n-(m-1)(w+1)}\cdot\rho$ 或 $r(P,S)\beta^{d-m}$(β 见引理 3)，则该序列模式称为非候选序列模式(当 $m=1$ 时，称为非候选项)，否则称为候选序列模式(当 $m=1$ 时，称为候选项).

定义 9.　一个长度为 m 的序列模式的候选支持率记为 $\rho(m)=\dfrac{n-(d-1)(w+1)}{n-(m-1)(w+1)}\cdot\rho$.

2.2　相关工作

序列模式挖掘的约束条件主要有 3 个[25]：间隔约束(gap condition/wildcards condition)、模式重叠约束(overlapping condition)和一次出现约束(one-off condition). 本文主要研究具有间隔约束、可以重叠、并且满足非一次出现的序列模式挖掘. 而这类序列模式的支持数计算复杂度较高，算法的效率尤为重要.

Zhang 等人[16]最先提出了带有间隔约束的序列模式挖掘问题，并给出了第一个挖掘算法 MPP. 算法 MPP 采用类 Apriori 性质进行序列模式挖掘，首先扫描序列 S 产生长度为 3 的频繁序列模式集 L_3 和候选序列模式集 CL_3；然后用长度为 $k(k>2)$ 的候选序列模式来产生长度为 $k+1$ 的候选序列模式集 CL_{k+1}，再计算候选模式集中每个模式的支持率，将不小于最小支持率的模式放到频繁序列模式集合中；直到某个 k 值使候选序列模式集 CL_{k+1} 为空为止. 算法 MPP 利用一种称为部分索引列表的结构来维护每个模式第一项对应的所有坐标及每个坐标下该模式对应的支持数，因此该算法计算每个候选项的支持率的时候不需要再次扫描原始序列，因此该算法的主要缺点为耗费大量空间来存储所有模式的索引列表，同时也影响了算法的时间效率.

算法 GCS[21]将一个长度为 m 的模式 P 与长度为 n 的序列 S 的匹配关系用一个 $m\times n$ 的矩阵来表示，尽管计算模式的支持数的效率提高了，但是算法的效率仍不理想. 在算法 GCS 的基础上，算法 MGCS 给出了改进，在模式 P 及其 $m\times n$ 的矩阵的基础上，同时计算模式 P 的所有以 P 为前缀、长度为 $m+1$ 的超模式的支持数，因此算法 MGCS 的时空效率得到了较大的提升.

算法 Amin[20]采用类 Apriori 算法的层次挖掘特点，首先产生长度为 1 的频繁项；然后用频繁项与序列中所有的项组合产生长度为 2 的频繁序列模式；然后重复地用长度为 k 的频繁序列模式分别与序列中所有项组合产生长度为 $k+1$ 的频繁序列模

式集 L_{k+1}，直到某个 k 值使 L_{k+1} 为空为止. 相比算法 MPP 和 MGCS，算法 Amin 很大程度上降低了检索空间，但同时也会导致一些频繁序列模式没有被挖掘到.

Wu 等人[15]提出算法 MAPB 和 MAPD，这两种算法都是利用网树来提高计算模式支持数的效率，其中算法 MAPB 是采用广度优先搜索及创建新的网树，算法 MAPD 是采用深度优先搜索及创建新的网树. 由于算法 MAPB 在挖掘的过程中，需要同时维护较多的网树；相对 MAPB 算法，MAPD 在挖掘的过程中，同时维护树的棵数较少，导致算法 MAPD 的空间消耗较少，并且 MAPD 的算法时间效率也比 MAPB 也较高. 和算法 MPP、MGCS 和 AMIN 相比，算法 MAPD 的时间效率提高幅度也较高.

通过以上算法的分析，目前算法 MAPD 的时空效率是较高的. 本文提出的算法也将和该算法进行实验对比.

3 算法 ITM

本文给出的算法 ITM 首先通过一遍原始序列的扫描，将序列中各项的索引存放到一棵索引树 I-Tree 上；然后通过 I-Tree，即可挖掘到所有的频繁序列模式，不再需要扫描原数据集.

3.1 树 I-Tree 的节点结构

树 I-Tree 中的节点结构分为 3 种：

(1) 根节点 root(第 1 层节点)包含 1 个字段：child 记录孩子节点.

(2) 根节点的孩子节点(第 2 层节点)包含 4 个字段：① item 记录节点的名字；② SN 记录支持数；③ IND 记录该节点对应项在原始序列中的下标列表；④ child 记录孩子节点.

(3) 其余层节点记录包含 4 个字段：① item 记录节点的名字；② SN 记录支持数；③ IND 记录该节点对应项在原始序列中的下标信息列表(下标信息包含下标 ind 及其下标对应的支持数 sn)；④ child 记录孩子节点. 这里的 IND 字段和第 1 层节点结构中 IND 不同，这里的 IND 包含下标和下标的相应支持数.

3.2 算法 ITM

描述算法之前，先给出算法中用到相关术语的定义.

定义 10. 根节点 root 到每个节点 N 路径上有序项组成的序列称为节点 N 的模式，其中第 d ($d>0$)层节点对应的序列长度为 $d-1$.

定义 11. 第 d 层上节点 N 的支持率等于 $N.SN/ofs(d-1,S)$.

定义 12. 一个节点的模式不是频繁的，则该节点称为非频繁节点；若节点的模式是频繁的，则该节点称为频繁节点；若节点的模式不是候选序列模式，则该节点也称为非候选节点；若节点的序列是候选序列模式，则该节点也称为候选节点.

算法 1. ITM 步骤如下：

步骤 1. 首先创建一个根节点 root；

步骤 2. 扫描一遍原始序列，将序列中所有不同的项添加到 root 的孩子节点中，同时将每项在原始序列中的下标以及支持数存放到节点中；

步骤 3. 扫描一遍第 2 层的节点，将频繁节点对应的序列模式保存到频繁序列模式集中，同时删除非候选节点；

步骤 4. 处理第 2 层中每个节点 N，假设 N 上的项为 X：

子步骤 4.1. 项 X 与第 2 层每个节点 N' 上项 Y 组合，产生序列 XY；

子步骤 4.2. 序列 XY 的支持数可以根据节点 N 和 N' 上索引信息进行计算(详见子算法 CSN)；

子步骤 4.3. 如果序列 XY 满足候选序列模式，则在项 Y 的孩子节点上增加节点 N，并将支持数、对应的下标和其支持数保存在节点 N 上；

子步骤 4.4. 如果序列 XY 是一个序列模式，将其保存到序列模式集合中.

步骤 5. 依次处理第 k($k>1$)层中每个节点 N，假设节点 N 的序列模式为 X，X' 为 X 的后缀模式：

子步骤 5.1. 序列模式 X 与 X' 的每个孩子节点 N' 上项 Y 组合，产生模式 XY；

子步骤 5.2. 序列模式 XY 的支持数可以根据节点 N 和 N' 上索引下标进行计算；

子步骤 5.3. 如果序列模式 XY 满足候选项集，则节点 N 上增加存放项 Y 的孩子节点，并将支持数、对应的下标和其支持数保存在节点上；

子步骤 5.4. 如果序列模式 XY 的支持率不小于最小支持率，则该序列就是一个频繁序列模式，将其保存到频繁序列模式集合中；

步骤 6. 循环执行第 5 步，直到第 k 层没有节点为止.

子算法 CSN. CSN($IND1$,$IND2$,$IND3$,D).

输入：节点 $N1$ 和 $N2$ 的下标列表 $IND1$ 和 $IND2$，初始值为空的下标列表 $IND3$，节点 $N1$ 的层次 D

输出：支持数 Tsn

$j1=0$，$Tsn=0$；

for $i=0$ to $IND2$.size() //$IND2$.size()表示列表 $IND2$ 中下标个数

$ind2=IND2[i]$；

$sn=0$；

for $j=j1$ to $IND1$.size()

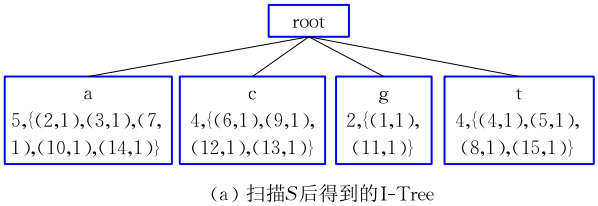
```
        ind1=IND1[j];
        //IND2[i].ind 表示列表 IND2 中第 i 个位置存放的下标
        if(IND2[i].ind-IND1[j].ind<M) break;
        else if (IND2[i].ind-IND1[j].ind>N)
          j1=j;
          continue;
        else
          if (D==1) sn++;
          else sn=sn+IND1[i].sn;
          //IND2[i].sn 表示列表 IND2 中第 i 个下标对应的支持数
      if (sn>0)
        IND3.add(ind2,sn); //将新产生的下标及其支持数存放到 IND3 中
        Tsn=Tsn+sn;
    Return Tsn;
```

3.3　算法 ITM 的实例

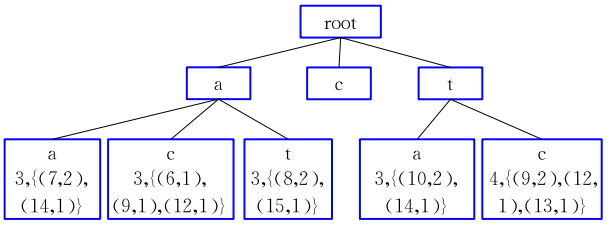
这里以 $S=s_1\,s_2\,s_3\,s_4\,s_5\,s_6\,s_7\,s_8\,s_9\,s_{10}\,s_{11}\,s_{12}\,s_{13}\,s_{14}\,s_{15}=$ gaattcatcagcca 为例说明 ITM 的挖掘过程. 这里设定 $M=3$，$N=5$，$\rho=14\%$.

步骤 1. 创建一根节点 root；

步骤 2. 采用 3.1 节中步骤 2 的方式，得到一棵如图 1(a)所示的树，如节点“a”上的“5”表示项 a 的支持数，“{(2,1),(3,1),(7,1),(10,1),(14,1)}”表示这 5 次分别出现的下标位置和相应位置的支持数(第 2 层节点上的支持数 1 可以省掉).



(a) 扫描S后得到的I-Tree



(b) 处理(a)中第2层节点后的结果

图 1　I-Tree 的创建

步骤 3. “g”节点的支持率低于设定的阈值，则将该节点删除.

步骤 4. 采用 3.1 节中步骤 4 的方式，节点 a 上的项“a”分别和第 2 层节点上的项(即“a”、“c”、“t”)进行组合产生序列模式，如产生序列模式“aa”，然后根据节点“a”下标列表(图 1 中第 2 层)，用算法 CSN 计算序列模式“aa”的支持数、及该模式中第二个“a”出现的下标和相应下标位置的支持数. 如图 1(b)第 3 层节点“a”上的“3,{(7,2),(14,1)}”，其中数字“3”表示模式“aa”的支持数；“(7,2)”中的“7”表示模式“aa”中第 2 个“a”(s_7)的下标，(7,2)”中的“2”表示有 2 个“a”分别和 s_7 组合产生模式“aa”($s_2\,s_7$，$s_3\,s_7$).

如图 1(b)上第 3 层节点“c”，首先根据节点“a”和“c”上的下标列表(图 1 中第 2 层)，用算法 CSN 计算序列模式“ac”的支持数及该模式中最后一项“c”出现的下标和相应下标位置的支持数. “ac”的支持数为 3，该模式中最后一项“c”出现的下标和相应支持数为“{(6,1),(9,1),(12,1)}”，其中“(6,1)”中的第一个数字“6”表示下标，“1”表示该下标下对应 1 次“ac”的出现($s_2\,s_6$)；由于序列模式“ac”的支持率不低于候选支持率，即可能存在以该模式为前缀的超集序列模式是频繁的，因此节点“a”增加存放项“c”的孩子节点；根据模式“ac”的支持数可以计算到该模式的支持率低于最小支持率，因此该模式不是频繁序列模式. 同样的方法，节点“a”上增加了 3 个孩子节点，节点“t”上增加了 2 个孩子节点，结果如图 1(b)中第 3 层节点所示.

步骤 5. 采用 3.1 节中步骤 5 的方式，依次处理第 3 层节点. 如第 3 层最左边的节点“c”，该节点的序列模式为“ac”，其后缀模式“c”没有孩子节点(见图 1(b)第 2 层节点“c”)，即不可能有以“ac”为前缀的频繁序列模式，所以该节点“c”没有孩子节点. 第 3 层节点“t”的序列模式是“at”，其后缀模式“t”的孩子节点包含 2 个(见图 1(b)第 2 层节点“t”的 2 个孩子节点)，即以“at”为前缀的序列模式只可能是“atc”和“ata”；根据节点“t”(第 3 层)和“c”(第 2 层节点 t 的孩子节点“c”)上的下标列表，用算法 CSN 计算序列模式“atc”的支持数为 4，则该序列的支持率低于候选支持率，因此序列模式“atc”不是频繁的，同时也不会产生以该模式为前缀的频繁序列模式. 同样的方式处理第 3 层其余节点，结果每个节点都没有孩子节点产生，即没有第 4 层节点产生，则算法执行完毕.

4　算法 AITM

根据定义 3，一个长度为 m 的序列模式 P(假设 P 的最后一个字符是“x”)的任一超序列模式 $Q=$“P-y”(Q 长度为 $m+1$)的个数最多为 $sup(P,S)\times W$，当 $sup(Q,S)=sup(P,S)\times W$，即在目标序列 S 中，任一模式 P 出现的后面 W 字符都是“y”，实际上，假定每个字符在 S 中出现概率相同，则任一个字符“x”后的 W 个字符中“y”个个数为 $W/|\Sigma|$，因此用 $sup(P,S)\times W$ 来估计 Q 的支持数会和实际值差距较大，从而算法 ITM 在挖掘序列模式的过程中，I-Tree 上会维护一些不会产生频繁序列模式的节点，影响降低了算法的时空效率.

为此，在算法 ITM 的基础上，给出一个近似算

法，当 I-Tree 每增加一层节点，都可以计算出模式“x”和“x-∅”（“∅”指原序列 S 中任一字符，即通配符）的支持数. 在给节点“x”（节点“x”的序列模式为 P）增加孩子节点后，存在一个支持数 $sup(P\text{-}y,S)$ 最大的孩子节点“y”，即可得到一个数值 $\alpha=sup(P\text{-}y,S)/sup(P,S)$（由于每处理完一个节点时，都会得到一个新的 α 值，如果新得到的值大于原来的值就进行更新来记录最大值）；当算法 ITM 再处理下一个节点时，就可以用 α 来代替引理 2 中的 W 来判断是否是候选模式，从而可以提前将一些非频繁的节点删除，减少空间需求，同时也减少算法下一步的计算量.

为了区别于 ITM，这里将改进后的近似算法记为 AITM.

5 算法复杂度分析

假设目标序列 S 的长度为 n，间隔约束为 $[N,M]$，设 $w=M-N+1$，以下分别从时间和空间复杂度对算法 1 进行分析.

(1) 算法的时间复杂度分析

在算法 ITM，产生一个 2 项模式，需要用两个项的下标列表进行计算，其时间复杂度为 $O(l_1(w/|\Sigma|))$，其中 l_1 表示这两项下标列表的最小长度；若产生一个 3 项模式，在算法 1 中，仍然需要两个下标列表计算，其时间复杂度为 $O(l_2(w/|\Sigma|))$，其中 l_2 表示这两项下标列表的最小长度；若一个模式长度为 d，假设 $l_1,l_2,l_3,\cdots,l_d$ 的均值为 L，则计算该长度为 d 的模式的时间复杂度为 $O(dL(w/|\Sigma|))$. 在算法 ITM 中，每一个最长的模式对应树 I-Tree 上的一个叶节点，假设叶子节点为 g 个，则算法 ITM 的时间复杂度为 $O\left(\sum_{i=1}^{g} d_i L_i w/|\Sigma|\right)$（$d_i$ 表示第 i 个模式的长度，L_i 表示产生第 i 个模式需要匹配下标列表的平均长度），一般情况下，从根到叶子的枝上包含多个频繁序列模式，即 g 小于频繁序列模式个数.

(2) 算法的空间复杂度分析

算法 ITM 的主要空间消耗在 I-Tree 树上，该树维护 2 类信息，一类是节点名、父节点、孩子节点，其空间复杂度为 $O(N)$，其中 N 表示树上的节点个数；另一类信息为下标列表信息，该类信息只存放到叶节点和第 2 层节点上，假设下标列表平均长度为 $n/|\Sigma|$，叶节点和第 2 层节点总数为 G，则其空间复杂度为 $O(Gn/|\Sigma|)$，即算法 1 的空间复杂度为 $O(N+Gn/|\Sigma|)$.

针对算法 AITM，引其提前对节点的模式进行估计，判断其是否会存在超序列模式，若不存在，则不需要保存下标列表，即是叶节点，也不需要保存下标列表，该节点也不会再继续参与模式挖掘，该节点也可从树上删除. 因此算法 AITM 的时空复杂度都会低于算法 ITM.

6 实 验

6.1 实验环境和参数设置

在本节中，对 ITM 和 AITM、MGCS、MAPB、MAPD 这 5 个算法进行实验验证和算法性能对比. 对比的所有算法都是采用 Java 编程语言实现. 实验平台：Windows 7 operating system，8GB Memory (Java heap size is 1.5G)，Intel(R) Core(TM) i7-3612 CPU@2.10GHz.

测试数据集采用文献[15]中的数据集，共用 3 个数据集 Homo Sapiens AX829174、Homo Sapiens AL158070 和 Homo Sapiens AB038490，数据集来自于网站 http://wuc.scse.hebut.edu.cn/msppwg/index.Htm. 另外从每个数据集第 1 个字符开始，分别取长度不同的字符串组成不同的序列片段，如表 1 所示. 在本实验中，针对目标序列的长度不同，对算法也进行性能测试.

表 1 生物数据序列

序列片段	取自数据	序列片段长度
S1	Homo Sapiens AX829174	1000
S2	Homo Sapiens AX829174	2000
S3	Homo Sapiens AX829174	4000
S4	Homo Sapiens AX829174	8000
S5	Homo Sapiens AX829174	10011
S6	Homo Sapiens AL158070	20000
S7	Homo Sapiens AL158070	40000
S8	Homo Sapiens AL158070	80000
S9	Homo Sapiens AL158070	167005
S10	Homo Sapiens AB038490	15000
S11	Homo Sapiens AB038490	30000
S12	Homo Sapiens AB038490	60000
S13	Homo Sapiens AB038490	131892

在本文的实验中，同文献[15]，将最大频繁模式长度估计为 13，序列模式的最小间隔 M 为 9、最大间隔 N 为 12.

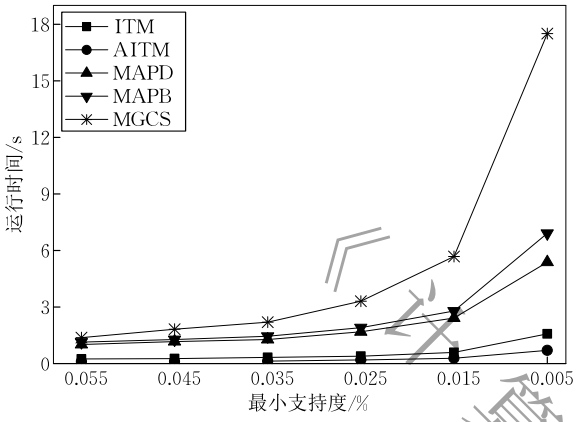
本文共设计了 3 组实验，一组测试最小支持度对算法的影响；第二组测试数据规模对算法的影响；最后测试通配符长度对算法效率的影响.

6.2 不同的最小支持度对算法的影响

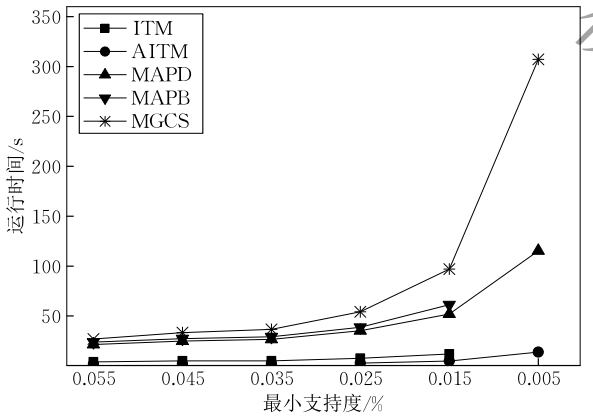
由于不同的最小支持度阈值会产生的序列模式个数不同，阈值越小，产生的个数会越多，同样的，算

法的运行时间也会随着阈值的降低而增加，第 1 个实验测试了算法在不同阈值下的表现情况. 数据集采用 3 个原始数据集序列(即序列片段 S5、S9 和 S13).

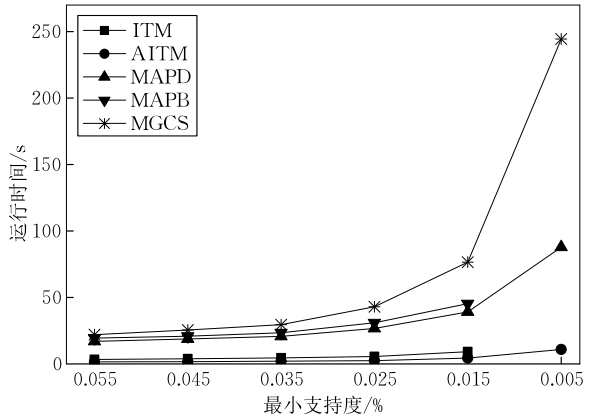
实验结果如图 2 所示，正如预期的结果，算法的运行时间都随着阈值降低而增加；当最小支持度阈值取 0.005%时，算法 MAPB 和 ITM 在执行过程中发生了内存溢出，因此这两个算法没有执行时间.



(a) AX829174(序列S5)



(b) AL158070(序列S9)



(c) AB038490(序列S13)

图 2　不同最小支持度下的运行时间对比

从图 2 可以明显看出，本文给出的算法 ITM 和 AITM 的时间效率较高，如最小支持度阈值为 0.015%时，在目标序列 S5 上，算法 ITM 的运行时间为 0.6 s，算法 AITM 的运行时间为 0.3 s；在序列 S9 上，算法 ITM 的运行时间为 11.9 s，算法 AITM 的运行时间为 4.9 s；在序列 S13 上，算法 ITM 的运行时间为 9.2 s，算法 AITM 的运行时间为 4.4 s.

本文给出算法主要有效记录序列模式最后一个字符在原序列中出现的位置，从而可以高效地计算出超模式的支持数，从而使算法的时间效率较高；同时本文给出的算法只需要记录树上最后一层节点上记录序列模式在原序列中出现的位置，从而也有效的提高了算法的空间效率；但是仍然在最小支持度阈值取比较低的时候，如 0.005%时，算法 ITM 在序列 S9 和 S13 上发生了内存溢出.

由于算法 ITM 内存主要消耗在维护一棵索引树上，而算法 AITM 通过估计一个节点是否存在超序列模式，就会提前将该节点从树上删掉，一方面降低了空间需求，另外一方面也不再需要计算该节点超序列模式的支持数，从而也提高算法的时间效率. 从实验结果上看，如图 2 所示，相对算法 ITM，算法AITM 的时间效率基本上提高了 1 倍. 尽管算法AITM是一个近似算法，该算法在 3 个数据集中的精确度都在 99.8%以上，如图 3 所示，特别在 AL167005(S9)和 AB131892(S13)上的精确度基本上都为 100%.

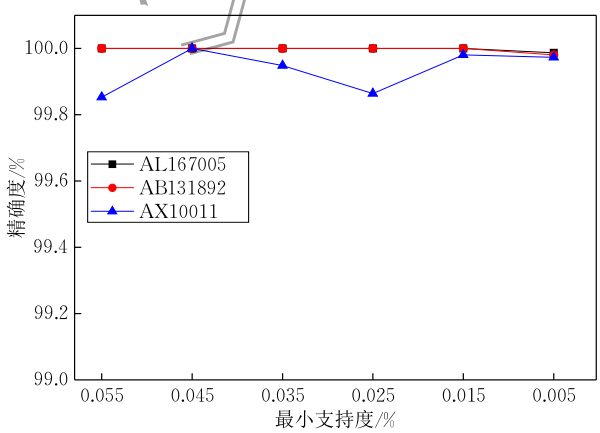
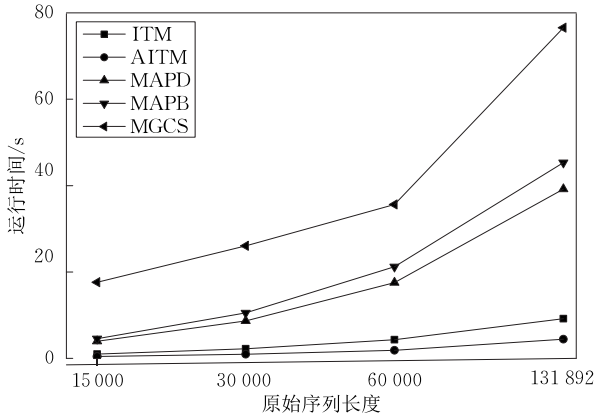


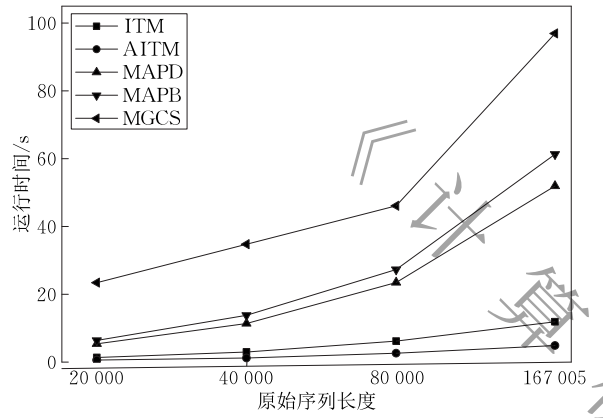
图 3　不同最小支持度下的精确度

6.3　算法扩展性

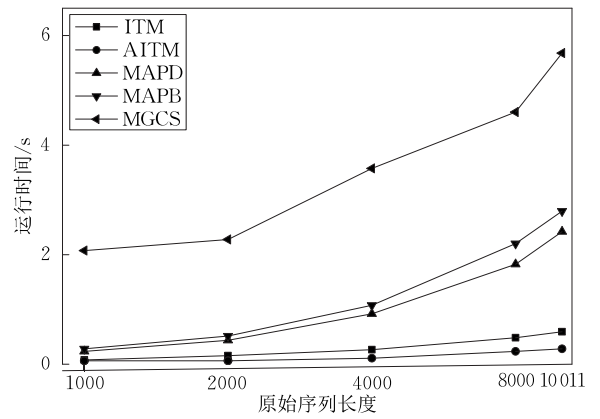
在当前大数据环境下，特别是数据量越来越大的情况下，算法的扩展性尤为重要，这里测试了算法在的扩展性，如表 1 中已经划分好的序列片段(不同大小)为测试数据，ρ=0.015%，测试结果如图 4 所示.



(a) AX829174(S1～S5)



(b) AL158070(S6～S9)



(c) AB038490(S10～S13)

图 4 算法扩展性

在相同的最小支持度阈值下,目标序列长度越长,算法执行的时间越长,如图 4 所示.从图 4 可明显看出,随着字符串长度的增加,算法 ITM 和 AITM 的运行时间增加较为缓慢,同时所需的执行时间较少.

尽管数据规模增加,数据集中包含的序列模式个数变化不大,如表 2 所示.由于索引树 I-Tree 上的节点个数和模式个数相关,当数据量变化时,树上节点个数变化不大,只是最后一层记录序列模式在原序列中位置会增多;因此随着数据量加大,本文给出算法的运行时间增加较为缓慢.

表 2 序列模式个数(S1～S5)

算法	序列长度				
	1000	2000	4000	8000	10011
ITM	5319	5191	5246	5176	5139
AITM	5304	5180	5230	5174	5138

在该实验中,由于算法 AITM 有效地删除树上的部分节点,算法 AITM 较 ITM 的运行时间效率提高了 1 倍以上,同时算法的精确度也比较高,如图 5 和表 2 所示,在 3 个数据集上的精确度都达到了 99.6%以上,如当在目标序列 S5 上,AITM 算法仅少挖掘到 1 个序列模式.

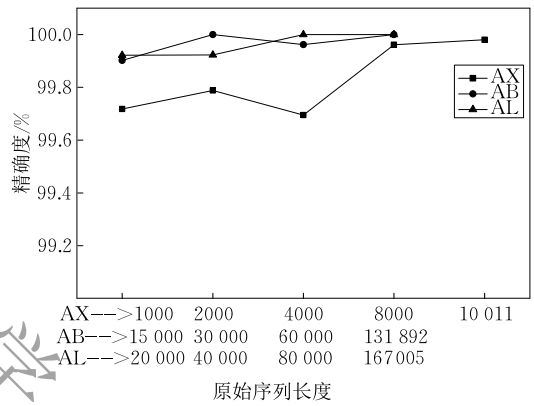


图 5 不同序列片段下近似算法的精确度

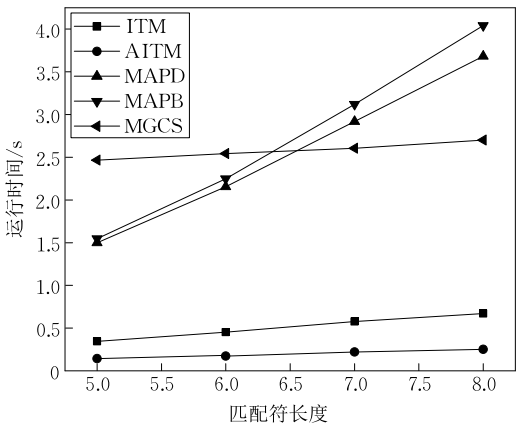
6.4 通配符长度对算法的影响

改变最小通配符个数 M,最大通配符个数 N 不变,即 $W=N-M+1$ 会随着 M 的减少而增大,即模式的支持数会随着 W 的增加而增加,从而会导致挖掘算法在计算序列模式支持数的计算量增大,理论上,挖掘算法的时间效率会随着 W 的增加而降低.

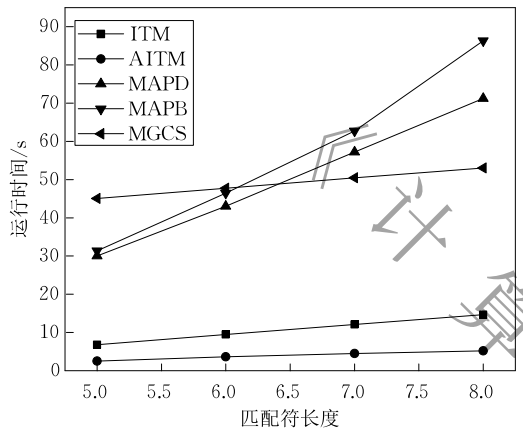
在本组实验中,N 值固定为 12,M 依次取 8、7、6 和 5,即 W 从 5 到 8,另外分别取了两个不同的最小支持度值进行了算法时间效率的实验对比,即 $\rho=0.035\%$ 和 $\rho=0.025\%$.

实验结果如图 6 和图 7 所示,随着 W 值的增加,所有算法运行时间都随着 W 值的增加而增加,但是算法 ITM、AITM 和 MGCS 运行时间增加比较平稳,算法 ITM 和 AITM 的时间效率仍然比较好.

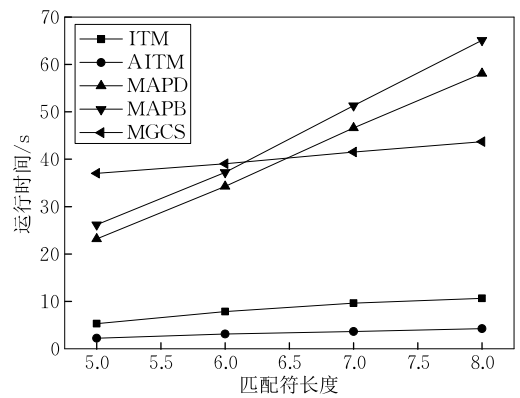
在不同最小支持度阈值上的实验结果也一致,当 W 越大,算法的运行时间都会随之增加;另外当



(a) AX829174(序列S5)

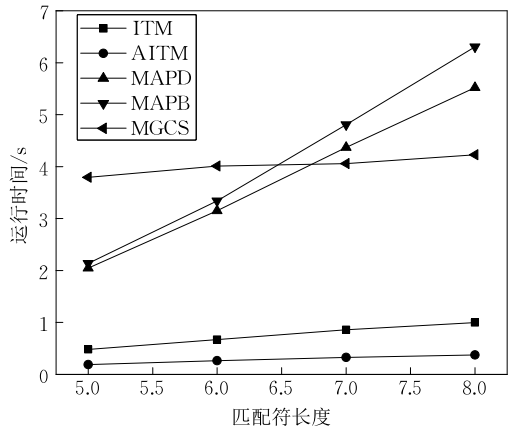


(b) AL158070(序列S9)

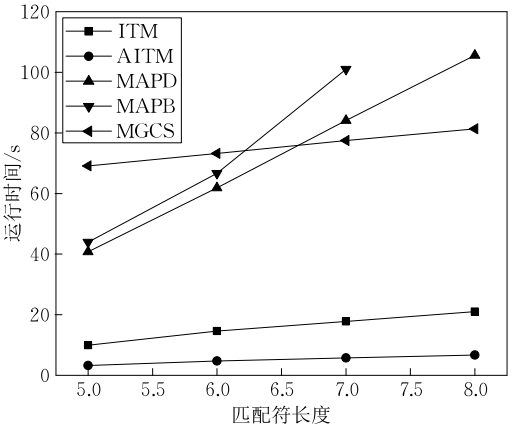


(c) AB038490(序列S13)

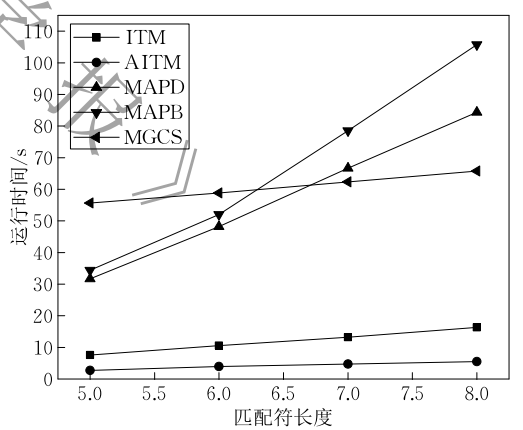
图 6　通配符长度对算法的影响($\rho=0.035\%$)



(a) AX829174(序列S5)



(b) AL158070(序列S9)



(c) AB038490(序列S13)

图 7　通配符长度对算法的影响($\rho=0.025\%$)

最小支持度阈值越小，挖掘算法的运行时间越长，这里的实验结果仍然同 6.2 节实验中的结果.

7　结　论

本文提出一个新的带通配符的序列模式挖掘算法 ITM，该算法通过一遍原序列的扫描，将产生的频繁 1 项集及其在原序列中的索引保存在树上，在此基础上产生长度为 2 的序列模式，并在原树上产生一层新的节点记录长度为 2 的序列模式及其序列模式索引信息；然后通过长度为 2 的序列模式和频繁 1 项集产生长度为 3 的序列模式，并将其维护在树上，依次类推，直到产生所有的序列模式. 该算法只在树上第 1 层和最后一层节点上记录序列模式的索引信息. 同时本文也给出一个剪枝策略，将估计不会产生超序列模式的节点提前删掉，从而提高算法的时空效率；并将该策略应用在算法 ITM，得到一

个近似算法 AITM. 实验验证了本文提出算法的有效性,时空效率有了一定的提高,给出的近似挖掘算法的结果精确度接近 100%.

致 谢 感谢武优西老师在个人网站(http://wuc.scse.hebut.edu.cn/resume/)上提供了部分算法代码和测试数据!

参 考 文 献

[1] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation//Proceedings of the ACM SIGMOD International Conference on Management of Data. Dallas, USA, 2000: 1-12

[2] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases//Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. Washington, USA, 1993: 207-216

[3] Wang L, Feng L, Jin B. Sliding window-based frequent itemsets mining over data streams using tail pointer table. International Journal of Computational Intelligence Systems, 2014, 7(1): 25-36

[4] Liao Guo-Qiong, Wu Ling-Qin, Wan Chang-Xuan. Frequent patterns mining over uncertain data streams based on probability decay window model. Journal of Computer Research and Development, 2012, 49(5): 1105-1115(in Chinese)
(廖国琼, 吴凌琴, 万常选. 基于概率衰减窗口模型的不确定数据流频繁模式挖掘. 计算机研究与发展, 2012, 49(5): 1105-1115)

[5] Liu Yin-Lei, Liu Yu-Bao, Chen Cheng. Efficient algorithm for mining of frequent itemsets over uncertain data streams. Journal of Computer Research and Development, 2011, 2011(S3): 1-7(in Chinese)
(刘殷雷, 刘玉葆, 陈程. 不确定性数据流上频繁项集挖掘的有效算法. 计算机研究与发展, 2011, 2011(S3): 1-7)

[6] Wang L, Feng L, Wu M. AT-mine: An efficient algorithm of frequent itemset mining on uncertain dataset. Journal of Computers , 2013, 8(6): 1417-1426

[7] Wan L, Chen L, Zhang C. Mining dependent frequent serial episodes from uncertain sequence data//Proceedings of the IEEE 13th International Conference on Data Mining (ICDM). Dallas, USA, 2013: 1211-1216

[8] Wang L, Wang S, Feng L. High expected weight itemsets mining on uncertain transaction datasets. International Journal of Advancements in Computing Technology, 2012, 4(20): 625-632

[9] Wang Le, Feng Lin, Wang Shui. An algorithm of mining TOP-*K* high utility patterns without generating candidates. Journal of Computer Research and Development, 2015, 52(2): 445-455(in Chinese)
(王乐, 冯林, 王水. 不产生候选项集的 TOP-*K* 高效用模式挖掘算法. 计算机研究与发展, 2015, 52(2): 445-455)

[10] Wang Le, Xiong Song-Quan, Chang Yan-Fen, et al. An algorithm for mining high utility patterns based on pattern-growth. Acta Automatica Sinica, 2015: 1616-1626(in Chinese)
(王乐, 熊松泉, 常艳芬等. 基于模式增长方式的高效用模式挖掘算法. 自动化学报, 2015: 1616-1626)

[11] Feng L, Wang L, Jin B. UT-Tree: Efficient mining of high utility itemsets from data streams. Intelligent Data Analysis (SCI&EI), 2013, 17(4): 585-602

[12] Tseng V S, Shie B, Wu C, et al. Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(8): 1772-1786

[13] Riondato M, Debrabant J A, Fonseca R, et al. PARMA: A parallel randomized algorithm for approximate association rules mining in MapReduce//Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 2012). Maui, USA, 2012: 85-94

[14] Lin M, Lee P, Hsueh S. Apriori-based frequent itemset mining algorithms on MapReduce//Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication. Kuala Lumpur, Malaysia, 2012: 76

[15] Wu Y, Wang L, Ren J, et al. Mining sequential patterns with periodic wildcard gaps. Applied Intelligence, 2014, 41(1): 99-116

[16] Zhang M, Kao B, Cheung D W, et al. Mining periodic patterns with gap requirement from sequences. ACM Transactions on Knowledge Discovery from Data, 2007, 1(2): 7

[17] Agrawal R, Srikant R. Mining sequential patterns//Proceedings of the 11th International Conference on Data Engineering. Taipei, China, 1995: 3-14

[18] Huang T C. Mining the change of customer behavior in fuzzy time-interval sequential patterns. Applied Soft Computing, 2012, 12(3): 1068-1086

[19] Li Z, Han J, Ji M, et al. MoveMine: Mining moving object data for discovery of animal movement patterns. ACM Transactions on Intelligent Systems and Technology, 2011, 2(4): 37

[20] Min F, Wu Y, Wu X. The Apriori property of sequence pattern mining with wildcard gaps. International Journal of Functional Informatics and Personalised Medicine, 2012, 4(1): 15-31

[21] Zhu X, Wu X. Mining complex patterns across sequences with gap requirements. International Joint Conference on Artificial Intelligence, 2007, 1(S2): S3

[22] Wu Y, Tang Z, Jiang H, et al. Approximate pattern matching with gap constraints. Journal of Information Science, 2016, 42(5): 639-658

[23] Yang Hao, Duan Lei, Hu Bin, et al. Mining Top-*k* distinguishing sequential patterns with gap constraint. Journal of Software, 2015, (11): 2994-3009(in Chinese)

(杨皓，段磊，胡斌等. 带间隔约束的 Top-k 对比序列模式挖掘. 软件学报，2015，(11)：2994-3009)

[24] Chai Xin, Jia Xiao-Fei, Wu You-Xi, et al. Strict pattern matching with general gaps and one-off condition. Journal of Software, 2015, (5): 1096-1112(in Chinese)
(柴欣，贾晓菲，武优西等. 一般间隙及一次性条件的严格模式匹配. 软件学报，2015，(5)：1096-1112)

[25] Wu Y, Shen C, Jiang H, et al. Strict pattern matching under non-overlapping condition. Science China Information Sciences, 2015, 58: 1-15

[26] Wu Y, Fu S, Jiang H, et al. Strict approximate pattern matching with general gaps. Applied Intelligence, 2015, 42(3): 566-580

WANG Le, born in 1978, Ph. D., lecturer. Her research interests include data mining, statistics learning theory and financial data analysis.

WANG Shui, born in 1967, professor. His research interests include data mining and financial data analysis.

LIU Sheng-Lan, born in 1984, Ph. D. His research interests include computer vision, machine learning and neural computing.

WANG Hui-Bing, born in 1989, Ph. D. candidate. His research interests include computer vision and machine learning.

Background

Mining sequential pattern with wildcards is one important issue in data mining; and for higher volume data, more efficient algorithm is needed.

Zhang et al.[16] raised the issue of mining sequential patterns with periodic restraints, and proposed the first algorithm MPP. Algorithm MGCS[21] used an $m \times n$ matrix to represent the matching relationship of pattern P and pattern S of length m and n respectively; although it improved the calculation for support, the overall efficiency was not satisfactory. The algorithm AMIN[20] adopted layered mining technics from Apriori, and compared with MPP and MGCS, decreased searching space significantly, but with the disadvantage of missing some of the frequent sequential patterns. Wu et al.[15] proposed algorithms MAPB and MAPD; both algorithms utilized Nettree to get more efficient mining on support, where MAPB utilized breadth-first search and MAPD utilized depth-first search, and both with creating new Nettrees. MAPD maintained less Nettrees than MAPB; it was more efficient in both space & time than MAPB, as well as MPP, MGCS and AMIN.

We propose a new algorithm dealing this problem; first we create a tree structure (we call "index-tree") to maintain original data series as well as sequential patterns and pattern indexing information; by scanning patterns of length $k(k>0)$ and their index information, sequential patterns of length $k+1$ are generated tier by tier. An approximate mining algorithm is also proposed; by estimating the support number of super sequential pattern, non-candidate nodes are deleted prematurely to reduce the number of nodes on index-tree, and therefore to promote algorithm efficiency. Experimental results verify the validity of these algorithms with increase of time/space efficiency, and the accuracy of approximate algorithm boosts up to 100%.