基于模式增长的高效用序列模式挖掘算法

唐辉军 [1] 王 乐 [2] 樊成立 [1]

摘 要 高效用序列模式挖掘是数据挖掘领域的一项重要内容, 在生物信息学、消费行为分析等方面具有重要的应用. 与传统基于频繁项模式挖掘方法不同, 高效用序列模式挖掘不仅考虑项集的内外效用, 更突出项集的时间序列含义, 计算复杂度较高. 尽管已经有一定数量的算法被提出应用于解决该类问题, 挖掘算法的时空效率依然成为该领域的主要研究热点问题. 鉴于此, 本文提出一个基于模式增长的高效用序列模式挖掘算法 HUSP-FP. 依据高效用序列项集必须满足事务效用闭包属性要求, 算法首先在去除无用项后建立全局树, 进而采用模式增长方法从全局树上获取全部高效用序列模式, 避免产生候选项集. 在实验环节与目前效率较好的 HUSP-Miner、USPAN、HUS-Span 三类算法进行了时空计算对比, 实验结果表明本文给出算法在较小阈值下仍能有效挖掘到相关序列模式, 并且在计算时间和空间使用效率两方面取得了较大的提高.

关键词 高效用序列模式, 模式增长, 闭包属性, 数据挖掘

引用格式 唐辉军, 王乐, 樊成立. 基于模式增长的高效用序列模式挖掘算法. 自动化学报, 2021, **47**(4): 943−954

DOI 10.16383/j.aas.c180660

A New Algorithm for Mining High Utility Sequential Patterns Based on Pattern-growth

TANG Hui-Jun[1] WANG Le[2] FAN Cheng-Li[1]

Abstract High utility sequential pattern mining is an important research topic in data mining. It plays an important role in many applications, such as bioinformatics and consumer behavior analysis. Different from traditional itemsets mining methods which take the appear numbers or the utility of the itemsets into account, high utility sequential pattern mining not only concerns the inner and outer utility but also the sequences of the items in the transactions, its computational complexity increases compared to the single frequent itemsets mining or high utility itemsets mining. Although a number of algorithms have been proposed to solve such problem, the efficiency of mining algorithms is still the main research topic in this field. In view of this, this paper proposes an efficient high utility sequential pattern mining algorithm named HUSP-FP based on pattern growth. Because the transaction utility value of the sequence itemset should satisfy the downward closure property, a global tree is established based on the sequential items in the transactions of the dataset after removing useless items. the HUSP-FP algorithm can efficiently extract sequential patterns from global tree without generating candidate itemsets. Comparing with the state-of-the-art high-utility sequential pattern mining algorithms of HUSP-Miner、USPAN and HUS-Span in our experiments, the proposed HUSP-FP out-performed its counterpart significantly.

Key words High utility sequential pattern, pattern growth, downward closure property, data mining

Citation Tang Hui-Jun, Wang Le, Fan Cheng-Li. A new algorithm for mining high utility sequential patterns based on pattern-growth. *Acta Automatica Sinica*, 2021, **47**(4): 943−954

序列模式挖掘作为数据挖掘领域的一项重要研究内容, 在风险管理、生物信息学、基因分析等方面具有重要的应用[1−3]. 频繁序列模式挖掘作为其应用的典型代表, 突出了序列数据集中的高频模式, 传统数据集上的频繁项集闭包属性表明: 任一个频繁项集的非空子集都是频繁项集, 非频繁项集的任一个超集都不是频繁项集, 基于上述闭包属性计算原则形成了序列集频繁模式挖掘算法[4−5]. 然而频繁模式没有考虑序列集中各项的效用值. 例如在零售业, 冰箱的销售要比牙刷卖的次数少的多, 但其利润可能比牙刷高的多, 因此在频繁序列模式的基础上, 提出了高效用序列模式挖掘, 即将内外效用值引入到序列项集中, 目前, 高效用序列模式已经成为数据挖掘领域的一个热门研究内容[6−8]. 2010 年,

收稿日期 2018-10-11 录用日期 2019-04-15
Manuscript received October 11, 2018; accepted April 15, 2019
浙江省公益技术应用研究计划项目 (LGF19H180002, 2017C35014), 宁波市自然科学基金项目 (2017A610122), 慈溪市社会发展科技计划项目 (CN2018001) 资助
Supported by Zhejiang Public Beneficial Technology Research Project (LGF19H180002, 2017C35014), Ningbo Natural Science Foundation Project (2017A610122), Cixi Social Development Science and Technology Project (CN2018001)
本文责任编委 张军平
Recommended by Associate Editor ZHANG Jun-Ping
1. 宁波财经学院金融与信息学院 宁波 315175 2. 宁波财经学院数字技术与工程学院 宁波 315175
1. School of Finance and Information, Ningbo University of Finance and Economics, Ningbo 315175 2. School of Digital Technology and Engineering, Ningbo University of Finance and Economics, Ningbo 315175

Ahmed 等[9] 给出了高效用序列模式挖掘的完整定义和数学模型. 其主要任务为在具有一定效用值的序列数据库中找出效用值不小于预先给定阈值的所有有序项集. 在定义过程中, 一个项集 X 的总效用值为 X 在该数据集中有包含项集 X 的事务 t 上的效用值 $U(X,t)$ 的总和. 而在单一事务 t 中可能存在多个相同序列集, 文献选择其中较大效用值作为项集 X 在该事务中的效用. 同时, 该文献还提出两种挖掘算法 UtilityLevel 和 UtilitySpan, 其中 UtilityLevel 通过事务中项集按水平次序组合生成候选项集, 进而判断和识别其中高效用序列. UtilitySpan 通过项集前缀垂直查找, 搜索高效用项集. 相比之下, UtilitySpan 的实现效率较好, 但当用户设定的阈值比较低或数据集中的不同项比较多时, 两类算法会产生众多的候选项集, 造成产生高效用序列项集的时空代价较大. 2012 年, Yin 等[10] 在 KDD 会议上提出的 USPAN 算法, 被认为能较大地提升高效用序列模式挖掘效率. 其主要基于项集树模型设计了数据库中序列模式, 所有序列模式均可从树中的结点得到, 并且基于高效用挖掘中的闭包属性原则给出了一个 SWU (Sequence-weighted utilization) 模型进行剪枝操作. 即一个项集的 SWU 值不小于用户定义的最小效用值, 该项集即为一个候选项集; 反之, 则该项集则不为高效用项集. 另外, 该文还通过设置事务中的项集的 Remain-utility 进行深度剪枝操作, 即一个项集的现有效用加上 Remain-utility 如果小于最小效用值, 则停止在树中搜索该项集. USPAN 的算法时空效率高于文献 [9] 中的算法. 但其算法基于枚举而产生, 时空效率代价也相对较大. Wang 等[11] 考虑到在较小阈值下, SWU 通常比最小效用值要大, 提出了一个不基于闭包属性原则的 HUS-Span 算法, 该算法主要设计了一种新的数据结构表示方法 - 效用链 (Utility-chain), 通过添加效用链剪枝达到计算效用值的目的. 不过, 该算法只在最小效用值较低或稀疏数据集时, 其性能优于 USPAN. 首先采用层次方法或者一定的数据结构 (树、链) 找出候选项集, 然后扫描数据集计算每个候选项集的真实效用值来判断该候选项集是否为高效用序列模式是 USPAN、HUS-Span 等算法的核心思想, 很多算法在此基础上进行数据结构设计改进[12−14], 2017 年, Zihayat 等[15] 在 USPAN 的基础上, 提出了 HUSP-Miner 算法, 该算法主要设计了两个新的数据结构 ItemUtilLists (Item utility lists) 和 HUSP-Tree (High utility sequential pattern tree), 其通过树形链式模式降低挖掘过程中的候选项个数, 在与 USPAN 的实验比较结果中可以看到, 算法的时空效率得到了一定的提升, 时间效率能提高到 3 倍. 高效用序列模式挖掘算法的时空效率有了很大的提高, 但算法的代价还比较大, 特别是挖掘时间效率的提升仍是本领域的一个挑战, 本文在基于高效用模式增长挖掘方法的基础上, 给出一个高效用序列模式新算法 HUSP-FP (High utility sequential pattern based on FP-growth). 该算法采用模式增长的方法来挖掘高效用序列模式, 所有的高效用序列项集均可从树上得的, 而不用再扫描原始数据集. 基于模式增长的挖掘算法已经在高效用无序数据集模式挖掘上表现出了独特的优势. HUI-Miner[16]、FHM[17]、HUPM-FP[18] 等算法相继提出, 在时空效率上得到了较大提升, 目前在序列数据集上开展高效用模式增长方法的研究成果还及其罕见. 本文首先给出高效用序列模式的相关数学定义; 然后, 给出 HUSP-FP 中的全局树建立过程, 重点突出从树中挖掘高效用序列模式的过程, 分析算法的时空效率优势; 最后, 开展相关实验, 与前述现有较好的 HUSP-Miner、UPSAN 和 HUS-Span 算法进行计算效率对比分析, 以验证新算法的优越性, 并开展总结分析提出下一步工作方向.

1 相关定义

设 $D = t_1, t_2, \cdots, t_n$ 为一个序列数据库, t_j 为构成该数据库的各个事务, $I = i_1, i_2, \cdots, i_m$ 为构成该数据库的所有项的集合. 在每一个事务 t_j 中, 设置$(x_{j1}:\ c_{j1})$, $(x_{j2}:\ c_{j2}), \cdots,$ $(x_{jv}:\ c_{jv})$为该事务的具体项和内部效应表示. $x_{jk}\ (k = 1, 2, \cdots, v)$ 为 I 具体的项, $c_{jk}(k = 1, 2, \cdots, v) = q(x_{jk}, t_j)$ 为其相应的效用值. $p(x_{jk})$ 为事务中该项相应的外部效用值. 表 1、表 2 分别为某一序列数据库和外部效用表实例.

表 1 高效用数据集

Table 1 High utility dataset

TID	Transaction (item, quantity)
t_1	$(A,4)(B,3)(D,3)(E,1)$
t_2	$(B,2)(C,2)(B,1)(G,4)$
t_3	$(A,3)(C,4)$
t_4	$(B,1)(C,1)(C,2)$
t_5	$(A,2)(B,3)(D,3)(E,2)(F,9)$
t_6	$(C,2)(D,2)(C,1)(G,7)$
t_7	$(A,2)(B,1)(A,1)(A,1)(B,1)$

定义 1. 项 x 在事务 t 中的效用值, 记为 $u(x,t)$, 其定义如下:

$$u(x,t) = p(x)q(x,t) \tag{1}$$

例如项 A 在事务 t_1 中的效用值为 12. 由于项 B 在事务 t_2 中有两个效用值, 分别为 6, 3. 这里设定在

事务中出现多个相同项集情况下, 取较大值为该项的效用值, 即项 B 的效用值为 6.

表 2　利润表 (外部效用值)

Table 2　A Profit Table (External utility value)

TID	Transaction (item, quantity)
A	3
B	3
C	2
D	2
E	2
F	1
G	1

定义 2. 项集 X 在事务 t 中的效用值, 记为 u (X, t), 其定义如下:

$$u(X,t)=\begin{cases}0, & X\not\subset t\\ sum_{x\in X}u(x,t), & X\subset t\end{cases}\tag{2}$$

例如项集A, B在事务 t_1 的效用值为 $12+9=21$. 考虑到项集在某一事务中可能多次出现, 例如有序项集B, C在事务 t_4 中效用值分别为 $3+2=5$ 和 $3+4=7$, 这里按照定义 1 中的设定, 取较大值 7 为该有序项集在事务 t_4 中的效用值.

定义 3. 序列项集 X 在数据集 D 中的效用值, 设为 u (X).

$$u(X)=sum_{t\in D}\ u(X,t)\tag{3}$$

例如序列项集A, B在数据集中的效用为 $21+15+12=48$.

定义 4. 事务 t 的效用值设为 $stu(t)$. 其包含了所在事务中的项效用值的和, 定义公式如下:

$$stu(t)=sum_{x\in t}u(x,t)\tag{4}$$

例如事务 t_3 的效用值为 $9+8=17$.

定义 5. 在数据集 D 的事务效用值为 $swu(D)$. 其定义如下:

$$swu(D)=sum_{t\in D}stu(t)\tag{5}$$

例如表 1 数据集中的事务效用值为 $29+17+17+9+34+17+18=141$.

定义 6. 基于数据集 D, 一个项集 X 的事务效用集不小于最小效用值, 则该项集为候选项集, 否则, 则为非候选项集.

$$swu(X)=sum_{X\subset t}\ stu(t)\tag{6}$$

例如序列项集A, B在数据集中的事务效用值为 $29+34+18=81$.

定义 7. 基于数据集 D, 一个项集的效用值 $u(X)$ 大于或等于最小效用值, 则该项集为高效用项集, 否则为非高效用集.

性质 1. 高效用序列项集的事务效用值必须满足闭包属性: 任一候选项集的非空子集是一个候选项集, 任一非候选项集的超集是一个非候选项集. 证明: 假设序列项集 X 是序列项集 Y 的一个子集, 也即 Y 是 X 的一个超集. 因为 X 是 Y 的一个子集, 因此数据集 D 中包含 Y 的事务项集一定也包含 X, 而包含 X 的事务项集不一定包含 Y, 即包含 X 的事务项集个数不会小于包含 Y 的事务项集个数, 因此 $swu(X)$ 一定大于等于 $swu(Y)$. 所以若 Y 满足闭包属性是一个候选项集, 则 X 一定也是一个候选项集; 若 X 不满足闭包属性, 即不是一个候选项集, 则 Y 一定也不是一个候选项集.

2　HUSP-FP算法

在本节中, 我们主要介绍基于模式增长的 HUSP-FP 算法在高效用序列模式挖掘中的应用. 其中, 第 2.1 节给出了全局树的建立, 其可将效用事务集压缩到一棵树上, 树上存储了所有效用信息. 第 2.2 节描述了如何从全局树上挖掘到所有的高效用序列项集模式.

2.1　全局树的建立

以表 1 和表 2 的数据为例, 设置最小阈值为 35, 构建全局树步骤如下:

步骤 1. 扫描一遍数据集, 统计各项的效用值 (Utility), 事务效用值 (swu), 进而形成数据头表 H 计算结果如图 1 (a) 所示

步骤 2. 将头表中事务效用值 swu 小于 35 的项删除, 由于 F、G 的 swu 均小于 35, 故删除 F、G, 更新后的头表如图 1 (b) 所示.

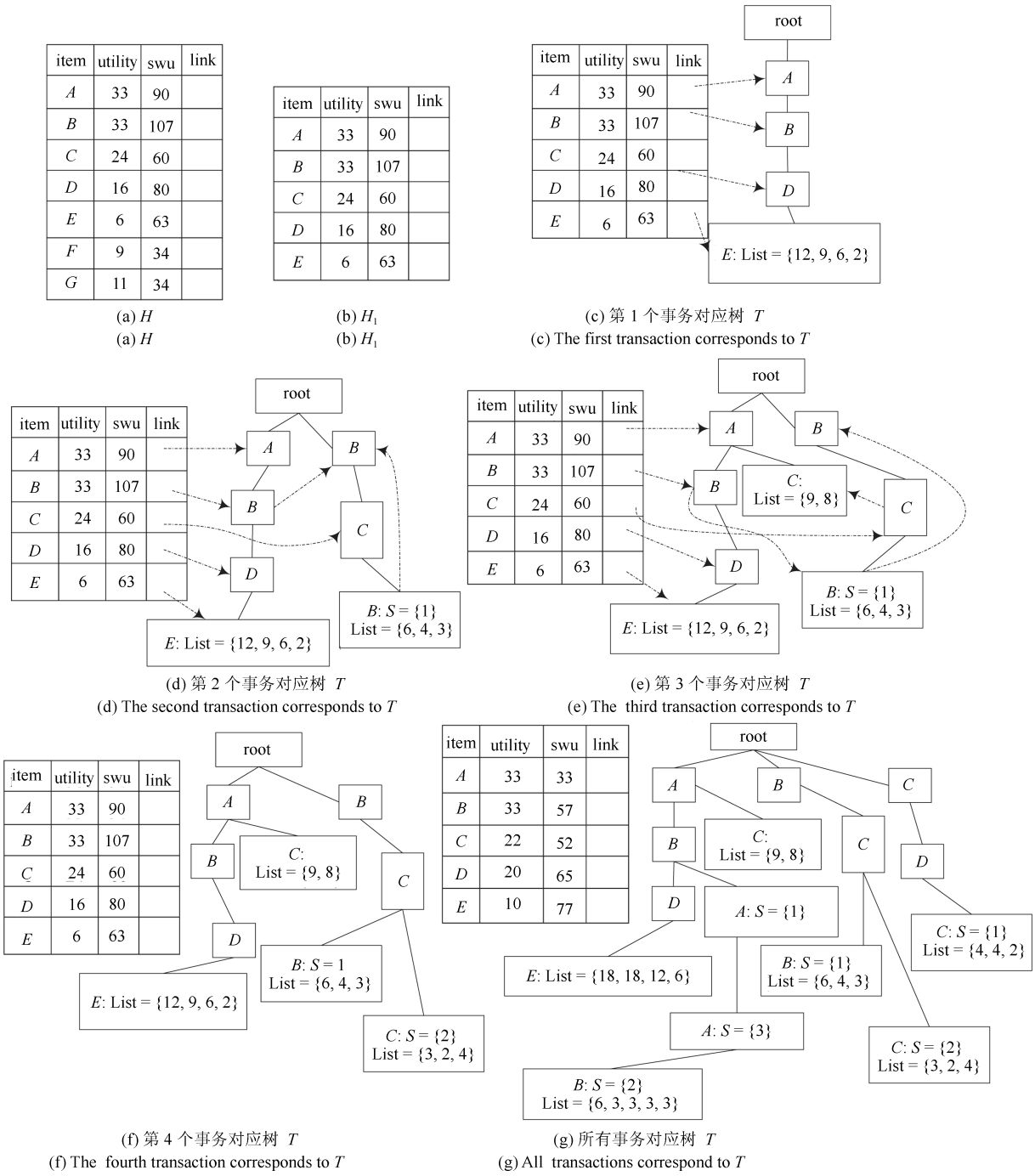
步骤 3. 再次扫描原始数据集, 将不在头表中出现的项删除, 并按原始项出现顺序依次添加到一颗树中.

表 1 中的第一个序列集 $(A,4)(B,3)(D,3)(E,1)$ 添加到树中的结果如图 1 (c) 所示. 叶子节点保留了全部项的效用值 List = [12, 9, 6, 2]. 图 1 (c) 表示了第一个事务添加后的结果. 图 1 (d) 为添加表 1 中第 2 个事务项集后的结果, 由于在该事务中出现了两个 B 项, 在最后出现的 B 项中记录下前一个 B 项在路径中出现的位置, 这里用 $S=1$ 来表示, 并设置链接指针到前一项. 在添加的过程中, 如果项集对应的最后一个节点在树上已经存在了, 则只需要将各项对应的效用值累加到 List 的对应项上. 第 4 个事务项集添加到树上后的结果如图 1 (f) 所示, 依次添加第 5、第 6 和第 7 个事务集, 在添加

第 7 个事务集中, A、B 项出现了多项, 第二个 A 中记录第一个 A 出现的下标位置, 第三个 A 记录第二个 A 的下标位置, 依次类推, 将所有效用值保留在叶子节点中. 所有事务项集添加后的结果如图 1 (g) 所示. 头表中的 link 对相同节点按照事务出现顺序进行链接, 以便于计算相同项集的效用值. 相关事务内容项指针链接如图 1 (c) ∼ (e) 所示, 为了图例看起来清晰, 图 1 (f) ∼ (g) 上没有画出 link 对应的事务内容项之间的链接线.

2.2 从全局树上挖掘所有序列模式

上述过程对序列全局树进行了有效构建, 将序列数据集维护到了一颗树和一张头表中, 算法 HUSP-FP 采用模式增长方式挖掘全部高效用序列集. 具体算法过程如算法 1 所示.



(a) H (b) H_1

(c) 第 1 个事务对应树 T

(c) The first transaction corresponds to T

(d) 第 2 个事务对应树 T

(d) The second transaction corresponds to T

(e) 第 3 个事务对应树 T

(e) The third transaction corresponds to T

(f) 第 4 个事务对应树 T

(f) The fourth transaction corresponds to T

(g) 所有事务对应树 T

(g) All transactions correspond to T

图 1 序列全局树构建实例

Fig. 1 An example of constructing a sequential global tree

算法 1. Mining(T, H, X, minUti)

输入: 全局树 T, 头表 H, 初始值为空的项集 X, 最小效用值 minUti.

输出: 高效用序列模式集 HUSPs.

Begin

1) For each item Q in H do /*from the order of appearance in the path */

2) If (Q.swu $\geq$ minUti) then

3) $X = X \cup Q$;

//判断项集 X 是否是高效用模式

4) If (Q.Ut $\geq$ minUti) then

5) 将项集 X 添加到 HUSPs ;

6) End if

//因为项集 X 的 swu 值不小于最小效用值, 可以为该项集创建子树

7) 为项集 X 产生子树 sT 和子头表 sH;

8) If (sH 不为空) then

9) Mining (sT, sH, X, minUti);

10) End if

11) 将项 Q 从项集 X 中删除;

12) End if

13) End for

End

算法 1 在头表和子头表的创建过程中, 将项集 X 的效用值存放在头表中的字段 Utility, 因此在算法 1 的第 4 行中可以直接判断项集 X 是否是高效用模式.

从头表出发, 计算包含以头表中每一项为尾节点的项集效用值. 具体到全局树的路径中, 则对应以路径中的叶子节点为出发点, 依次处理每一项得到其相应的子表和子树, 处理完每一项即可挖掘到以该项为尾项的序列项集. 在计算过程中, 先通过判断事务效用值 swu 是否满足闭包属性进而计算其序列模式是否为高效用序列模式. 在图 1 (g) 的路径出现过程中, A-B-D-E 为第一个路径, 包含 C 的路径 B-C-B 为第二个路径, 由于头表中只有这 5 项已经全部出现, 则挖掘头表中项的顺序为 E-D-B-A-C.

这里以图 1 (g) 中的全局树和头表为例, 说明算法 1 构建子树和子头表的过程. 由于项 E、D 的包含项集比较简单, 这里选择挖掘以项 B 为尾节点的高效用序列项集作为演示例子. 当处理头表 (图 1 (g)) 中项 B 时, 因为头表中项 B 的 swu 值不小于最小阈值 35, 因此可以为项集 $X = B$ 创建子树和子头表, 构建步骤如下:

步骤 1. 按照从头表指针链中 B 点出发链接形成的路径, 分别按照从下往上的顺序从路径 A-B 上分析带有 B 项的项集为 A、B. 从路径 B-C-B 项获取项集 C、B、B、B. 从路径 A-B-A-A-B (这里指针链接到第二个 B 项) 上获取项集 A、B、B、B. 按照项集效用和事务效用值计算规则, 得到 B 的子头表如图 2 (a) 所示. 按照前述第 2.1 节全局树建立规则, 删除 swu 不满足最小效用值要求的无用项集 B、B 和 C、B 得到优化后的子头表如图 2 (b) 所示.

步骤 2. 由于子头表不为空, 继续以子头表中的某一项集为基项, 构建以该子头表为基础的子树. 按照原始序列挖掘顺序添加相关节点, 添加第一条路径后得到项集 B 的子树如图 2 (c) 所示, 其中 Base 表示了路径基项 B 的效用值. 所有路径添加完毕后的结果如图 2 (d) 所示. 继续以子头表中的某一项为基项得到后续子集 A、A、B 和 B、A、B, 发现其 swu 均小于最小效用值, 基项 B 的序列项集处理完毕, 计算得到 A、B, A、A、B 和 B、A、B 为高效用序列项集. 依次类推, 图 2 (e) 则表示了处理 D 为基项后的子树和子头表.

2.3 HUSP-FP 理论分析

算法 HUSP-FP 将所有项的效用值维护在全局树的叶子节点上, 并且其存储的是所有的真实效用值而不是估计值. 这使得其计算中不需要产生候选项集. 在子树的创建过程中 (如图 2), 依据路径叶子节点, 可以得到子树节点中每一项的真实效用值. 依据事务项集闭包属性要求, 一旦子树中的项集事务效用值 swu 小于最小阈值, 则抛弃该项集, 最终使得创建子树和子头表的次数得到了大幅度减少, 计算效率得到了有效提升.

以下证明 HUSP-FP 能够挖掘到所有的高效用序列模式. 首先假设项集 Z 是任何一个高效用序列模式. 因为 Z 是一个高效用序列模式, 则其任何一个非空子集的 swu 值都不会小于最小阈值. 根据算法 HUSP-FP, 序列项集 Z 中的所有项都会出现在全局树对应的头表中, 当 Z 中第一个项 $Z1$ 被处理, 并为之创建子头表时, 因为项集 $Z1$ 和 Z 中其余各项的组合的 swu 值都不小于最小阈值, 因此这些序列组合的项集都会出现在子头表中, 当处理子头表中项 $Z1$ 和 Z 中的第二项 $Z2$ 的序列组合时, 同样项集 $Z1$、$Z2$ 和 Z 中其余各项的序列组合会出现在新的子头表中, 依次处理下去, 一定能够得到 Z 所包含的全部高效用序列模式.

与前述基于候选集的已有算法 HUSP-Miner、USPAN 和 HUS-Span 相比, HUSP-FP 能够依据模式增长方法挖掘得到所有高效用序列模式, 并且在挖掘过程中无候选项.

全局树中的事务效用存储结构也相对简单, 这与其他三类基于候选项剪枝进而得到高效用序列模式的方法有本质不同, 这也是 HUSP-FP 能够取得较好时空效率的主要原因, 相关算法的复杂度比较内容可见表 3 所示.

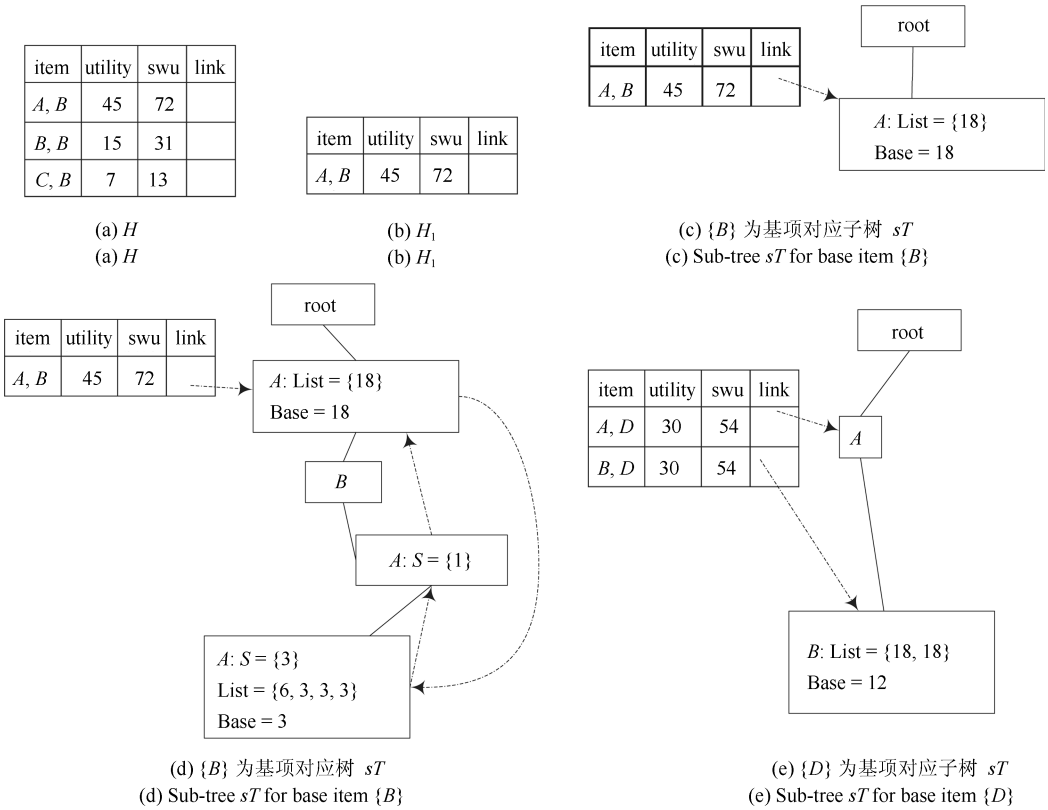


图 2 序列子树构建实例

Fig. 2 An example of constructing a sequential sub-tree

表 3 算法复杂度比较

Table 3 Comparisons of algorithm complexity

算法	事务存储	效用存储	挖掘方式	候选项
HUSP-FP	全局树	叶子结点	模式增长, 一次性完成	无
HUSP-Miner	序列表格	序列树结点	表格搜索, 深度剪枝	有
USPAN	序列字典树	效用矩阵	枚举搜索, 剩余量剪枝	有
HUS-Span	序列枚举树	链表	效用上界深度搜索	有

3 实验与结果

3.1 实验算法与平台

算法 HUSP-Miner、USPAN 和 HUS-Span 是目前时空效率较好的高效用序列模式挖掘算法, 本文新提出的算法 HUSP-FP 和这三类算法做了对比分析. 算法均采用 java 编程语言实现, 实验平台: Windows 7, 10 GB Memory (Java heap size is 1.5 GB), Intel (R) Core (TM) i5-3320 CPU @ 2.60 GHz.

3.2 实验数据

本节采用的数据来自 SPNF[19] 和文献 [20], 共 4 个真实数据集, 包括 Bible, FIFA, Kosarak 10k, SIGN. 其中, Bible 是一个具有中等序列长度的稠密数据集, FIFA 是一个具有较多长序列的稠密数据集, Kosarak 10k 则是一个包含短序列的稀疏数据集, SIGN 是一个具有超长序列类型的稠密数据集, 数据集的类型丰富性是足够的, 关于数据集的具体内在特征, 可见表 4 所示.

表 4 算法复杂度比较

Table 4 Data characteristics

数据集	项数	序列事务项集平均长度	序列事务总数
Bible	13 905	21.64	36 369
FIFA	13 749	45.32	573 060
Kosarak 10k	39 998	11.64	638 811
SIGN	267	93.00	730

3.3 时空效率

算法运行时间会随着最小效用值 (最小阈值) 的增加而降低. 第一个实验测试了不同最小阈值下, 算法的运行时间和内存消耗对比. 图 3 给出了在不同阈值要求下的 4 种算法在 4 个数据集的运行时间对比.

项的外部效用值采用文献 [21−22] 中的方法, 设定为随机产生的一个数值 (大于等于 0.0100, 小于等于 10.0000). 从图 3 中可以看到, 新算法的时

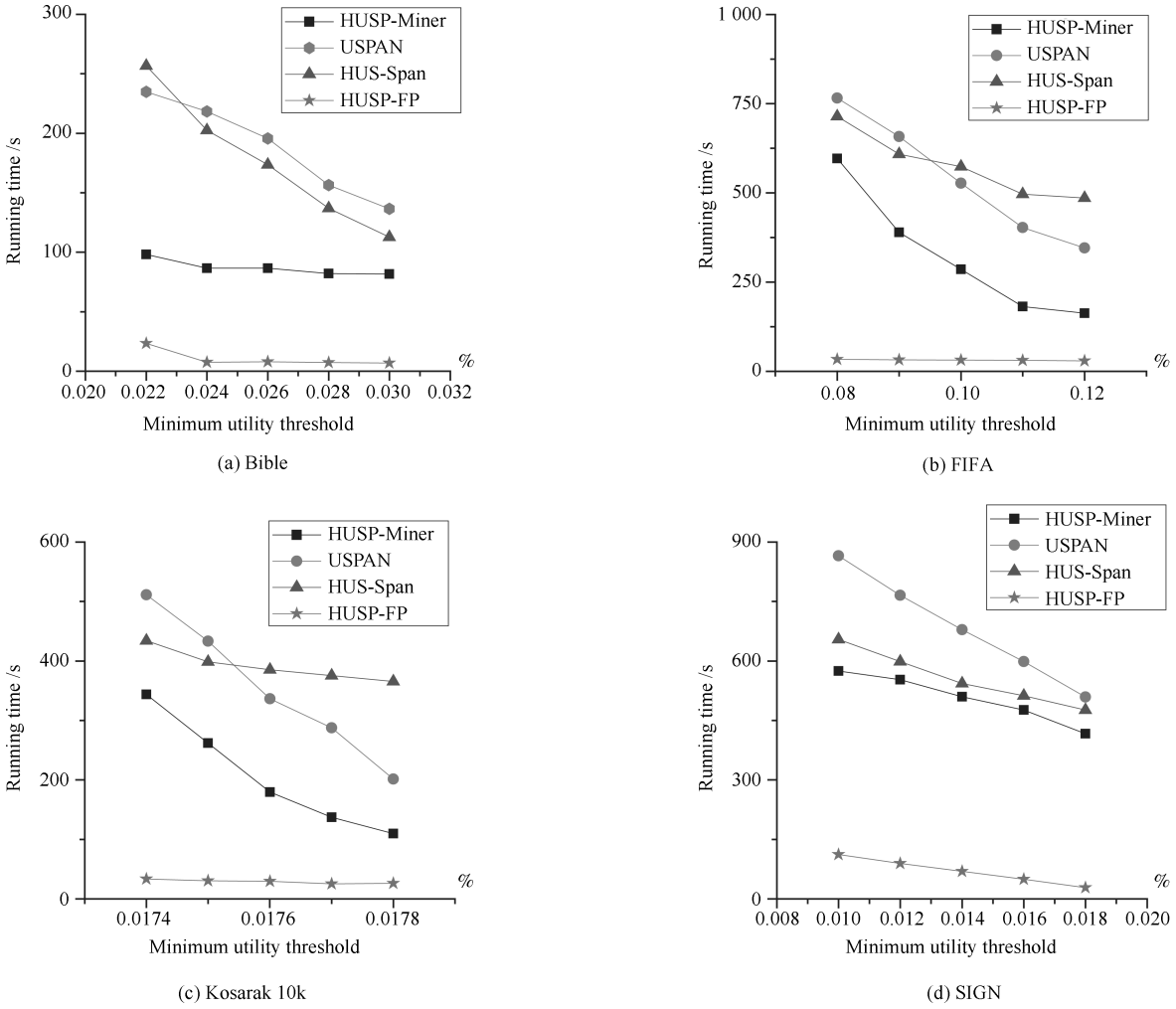


图 3　运行时间

Fig. 3　Running time

间效率得到了较大提升, 个别数据集上可以提高达到 1 个数量级以上. 建立全局树然后基于树进行高效用序列模式挖掘是 HUSP-FP 算法时间消耗的两大部分, 不过随着阈值的增加, 相比其他算法来说, HUSP-FP 的挖掘时间变化比较平稳. 对于短序列数据集 Bible 和 Kosarak 10k 来说, 由于 HUSP-Miner、USPAN 和 HUS-Span 在实现过程中产生了较多的候选项集, 从而导致算法的时间消耗较大. 对于长序列稠密数据集 FIFA 和 SIGN, 三类算法更是付出了较多的时间代价, 例如对于数据集 SIGN, 当最小阈值达到 0.018% 时, 此时已无高效用序列模式产生, 三类算法仍然消耗了大量时间进行候选项判断, 由此可见, 新算法 HUSP-FP 在时间运行效率上取得了良好的领先结果.

3.4　算法扩展性测试

第 2 个实验测试在不同数据量下算法的运行时间和内存消耗性能, 分别选取稠密数据集 Bible 和稀疏数据集 Kosarak 10k 作为测试对象, 数据选择方式为从 Bible 数据集中从头到尾顺序添加 5 000、10 000、15 000、20 000、25 000、30 000 事务数据量, 且设置其最小阈值为 0.026%. 从 Kosarak 10k 数据集中选择 100 000、200 000、300 000、400 000、500 000、600 000 事务数据量, 且设置其最小阈值为 0.0176%. 其中图 5、图 6 分别表示了 4 类算法在不同数据量下的时空效率. 从图中可以看到, 随着数据测试集数据量的增大, HUSP-FP 算法产生的高效用序列模式增多, 但是 HUSP-FP 的时间消耗量却相对较为稳定, 这是由于算法按照闭包属性, 在模式增长过程中及时的抛弃了一些无用序列项, 从而使得运行时间保持了平稳. HUSP-Miner、USPAN 和 HUS-Span 基于层次的搜索方法产生了较多的候选项, 虽然三类算法大都采用树形结构, 但它们在全局树建立过程中采用枚举搜索的方式增加了较大的计算时间. 通过对算法的内存消耗对比, 发现 4 类算法的内存消耗都随着数据量的增加成线性增加模

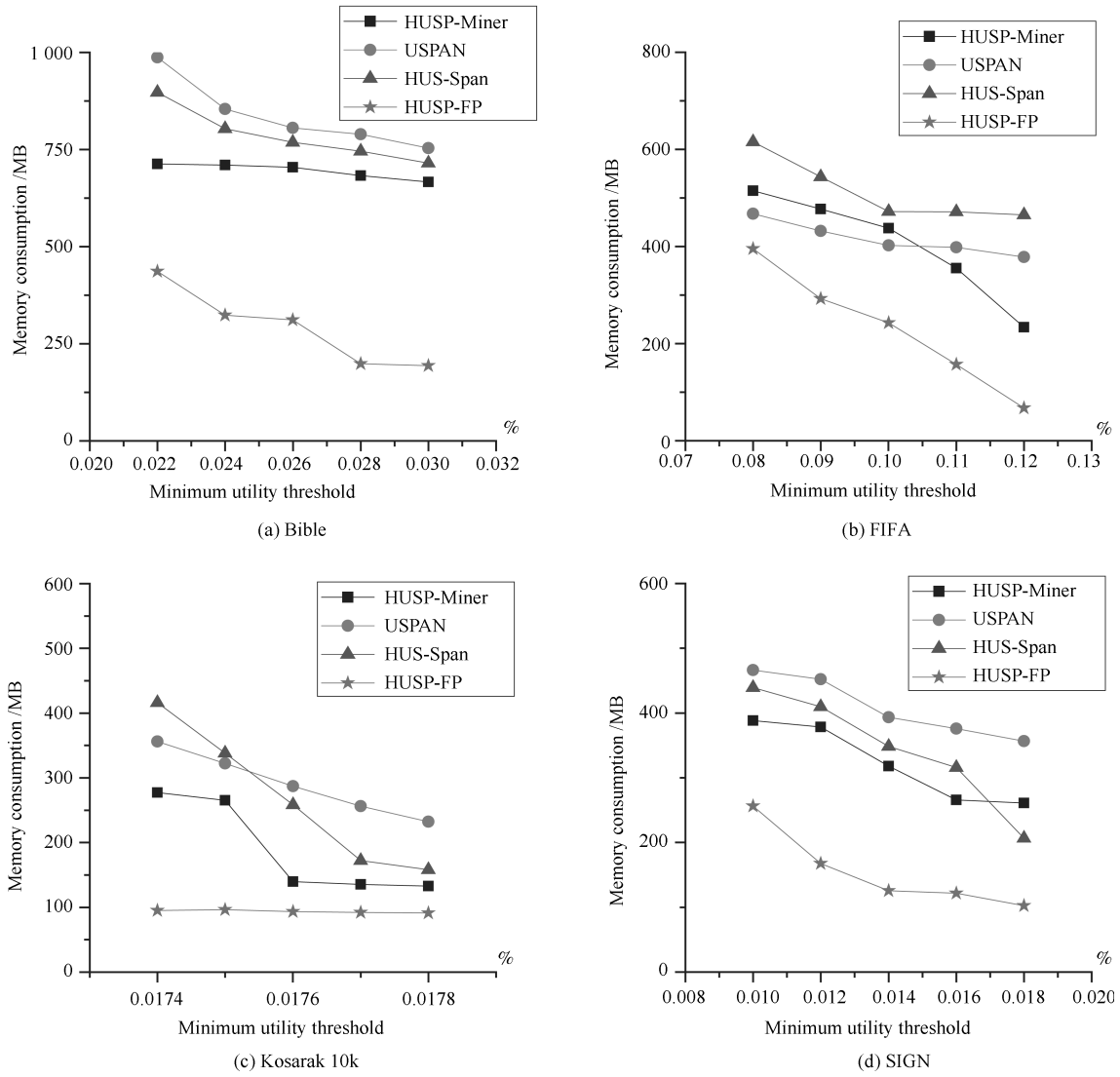


图 4 内存消耗

Fig. 4 Memory consumption

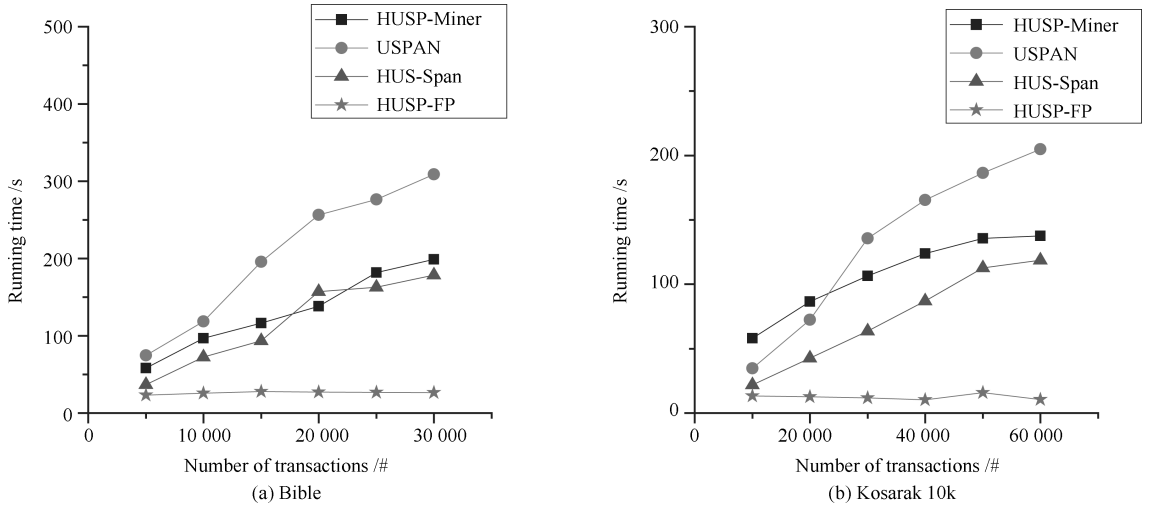


图 5 不同数据量运行时间

Fig. 5 Running time under different data size

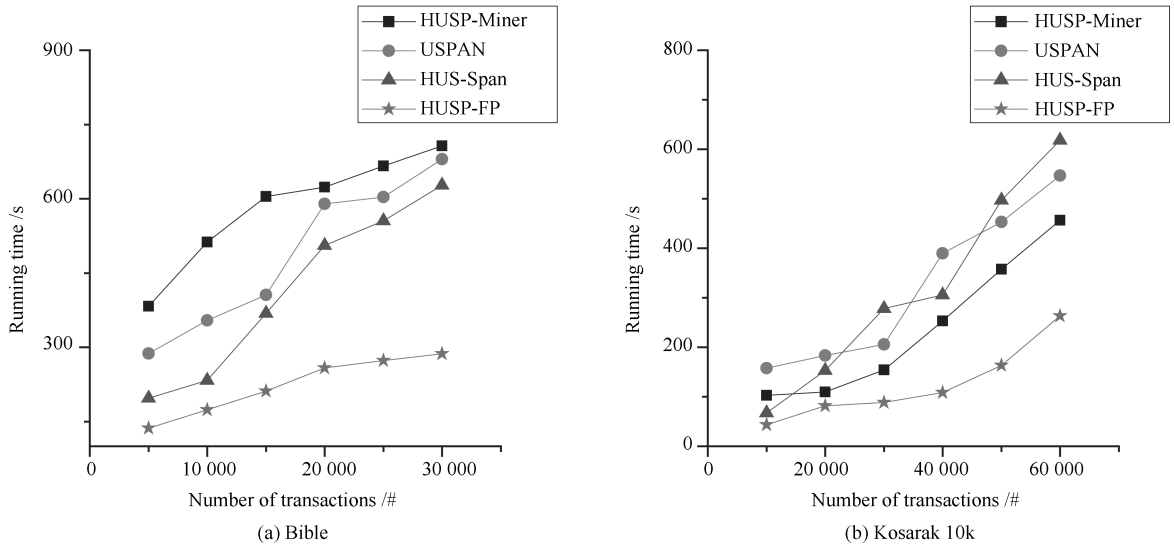


图 6　不同数据量内存消耗

Fig. 6　Memory consumption under different data size

式 (该测试结果不包含垃圾节点占用空间, 垃圾节点累积到一定程度, 系统会自动回收垃圾所占用的空间), 但总体上看, HUSP-FP 消耗的内存更少. 由此可见, 新算法在数据量增多的情形下时空效率都优于其他三类算法, 特别是计算时间上优势明显.

图 4 给出了在上述 4 个数据集运算中的内存消耗量对比. 本节实验中的内存消耗值的提取方法如下: 在算法运行过程中, 设置不同的点, 提取每个点上的所用内存量, 然后从所有不同点上取出最大的内存消耗量作为当前算法的内存消耗. 在不同的最小阈值下, 相比其他三类算法, 算法 HUSP-FP 在空间使用效率上也有较大的优势. 当最小阈值减小, 数据集中包含的高效用模式会增加, 从而使得新算法 HUSP-FP 创建的全局树占用空间增加, 同时创建的子树棵数也会增加, 所以算法 HUSP-FP 会随着最小阈值的降低而消耗更多的内存空间. 但与其他三类算法相比较, HUSP-FP 还是取得了较大的优势.

第 3 个实验进一步测试了在数据流窗口下的算法运算效率. 图 7 给出在不同最小阈值下, 4 个算法在上述 4 类典型数据集下的运行时间 (这里设置一个时间窗口包含 5 个序列批次, 每一批次包含事务容量不大于 20 k). 从图 7 的实验中可以发现, HUSP-FP 的运行时间明显小于其他三类算法, 随着实验中最小阈值的增大, 其他算法的运行时间也得到了一定的下降, 但下降后的值与 HUSP-FP 相比, 差距仍然较大. 这说明 HUSP-FP 能够在时间窗口数据流挖掘上具备较好的运行时间优势. 从图 8 的内存消耗比较中也可以看出, HUSP-FP 在数据流窗口下内存计算消耗量相对较小. 图 7 和图 8 表明新算法 HUSP-FP 在数据流窗口高效用序列模式挖掘中能够显示出较好的运行效率优势.

分别设置 Bible、FIFA、K0sarak 10k、SIGN 的最小阈值为 0.026 %、0.10 %、0.176 %、0.014 %, 图 9 给出了不同批处理数据容量 (例如 Bible-10k 表示划分 Bible 单一批处理数据集的大小不超过 10k) 和不同批处理数据数量 (Bible-3 表示一个时间窗口有 3 个批处理集) 下的运行时间对比, 从图 7 中可以看出, HUSP- Miner 相比 USPAN 和 HUS-Span, 运行时间更少, 故图 9 对 HUSP-Miner 和 HUSP-FP 开展了对比分析, 随着相关批处理数据容量和数量的增大, 在对同一数据集的分析中可以发现, HUSP-FP 和 HUSP-Miner 都会增长运行时间, 但 HUSP-FP 的增长值相对 HUSP-Miner 更小, 能够保持相对的稳定, 这更进一步说明了 HUSP-FP 在时间窗口高效用序列模式挖掘中的优越性.

4　总结

本文提出一种新的高效用序列模式挖掘算法 HUSP-FP, 该算法基于模式增长方式可直接挖掘高效用序列模式, 而不是传统的先得到候选项集再筛选的过程, 因此算法的时空效率都得到了提升. 本文采用 4 个典型真实数据集 (包含短序列、长序列、稀疏数据集、稠密数据集) 进行了静态和时间窗口动态实验验证, 实验结果也表明给出算法的时间和空间效率相比目前较好的 HUSP-Miner、USPAN 和 HUS-Span 都有较大程度的提高, 特别是时间效率得到了较大幅度的提高, 证明该方法是可行的. 同时通过实验运行, 发现 HUSP-FP 算法即使在最小阈值较大情况下, 在维护全局和子树环节仍然需要消

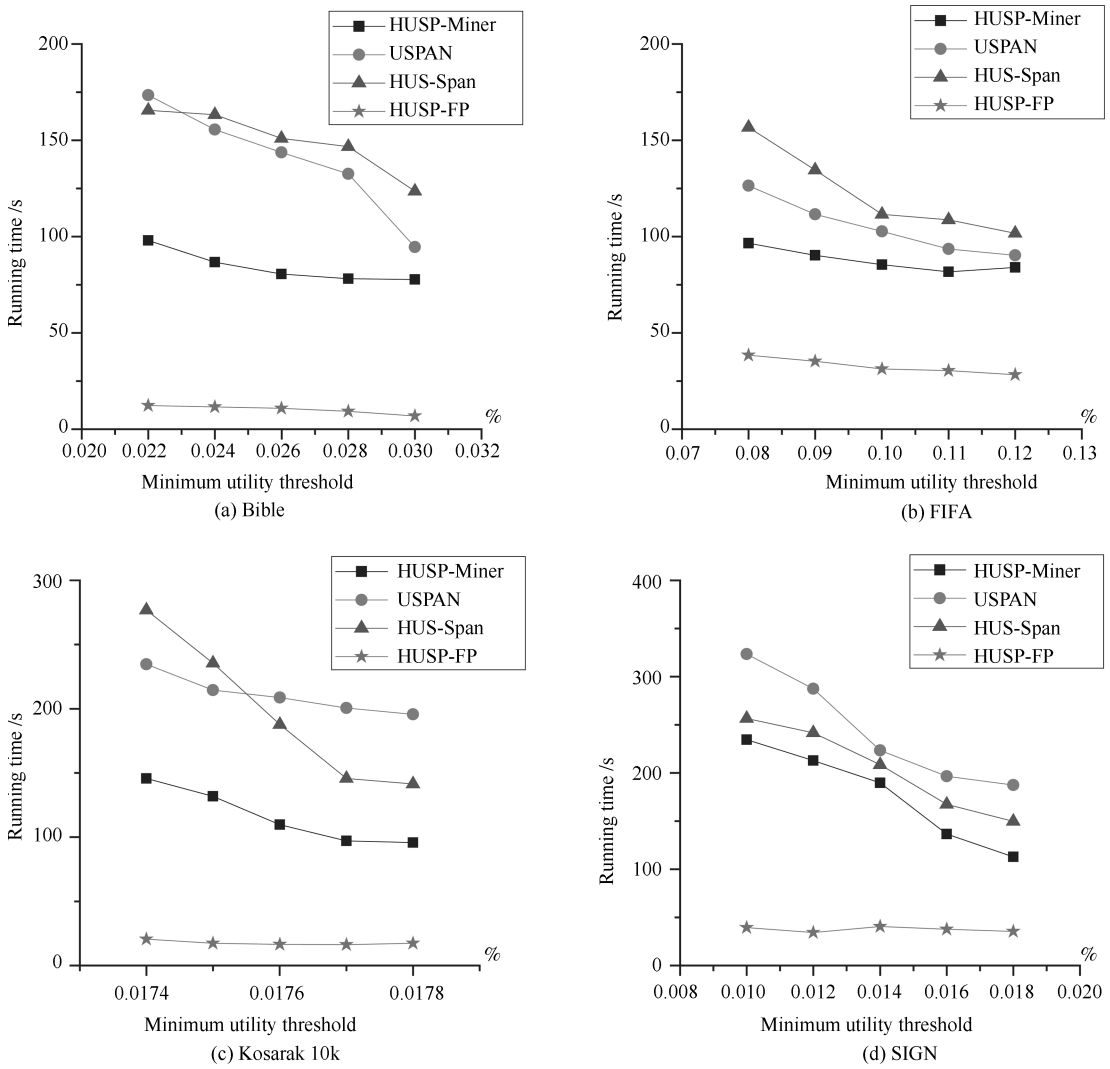
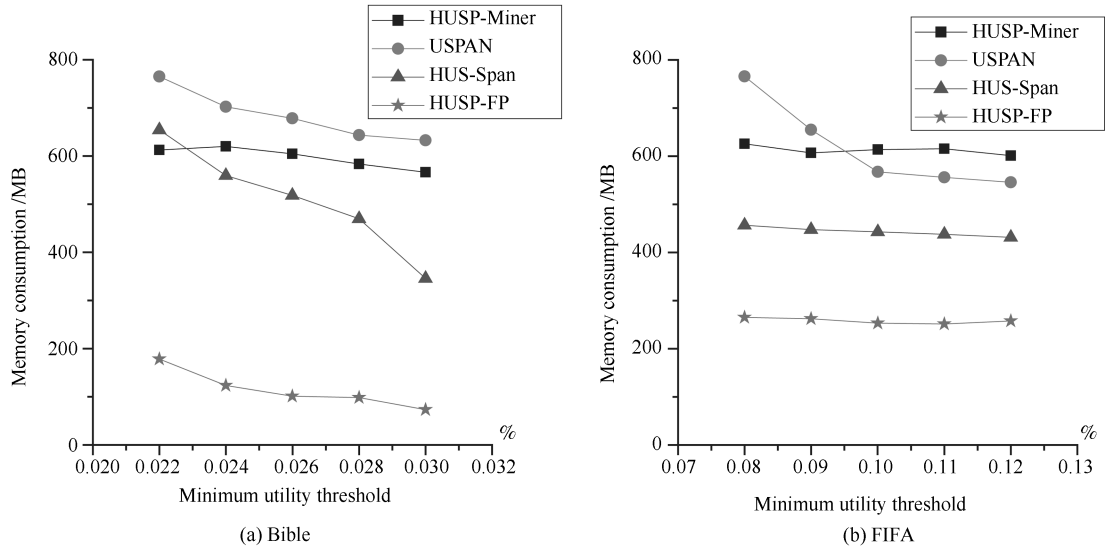
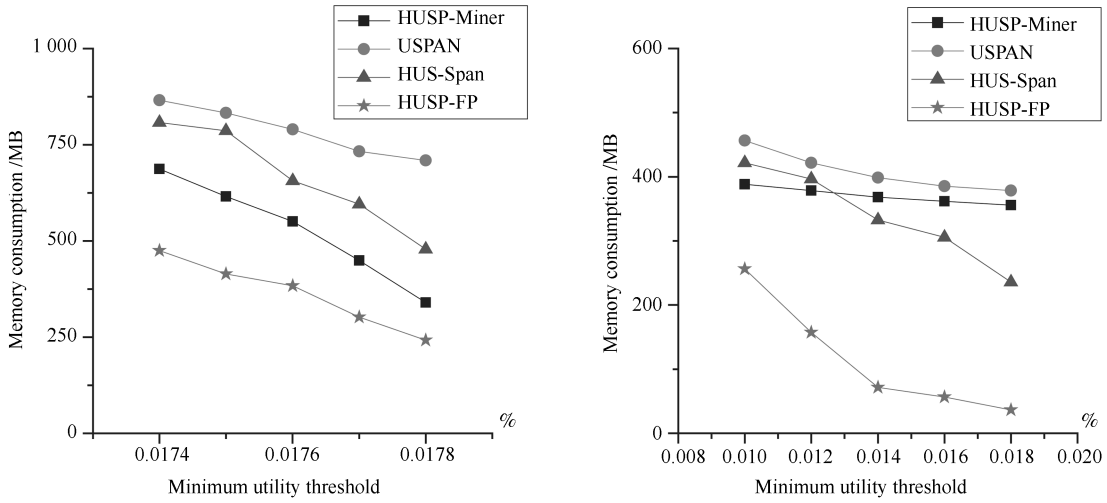


图 7 数据流运行时间

Fig. 7 Running time based on data stream

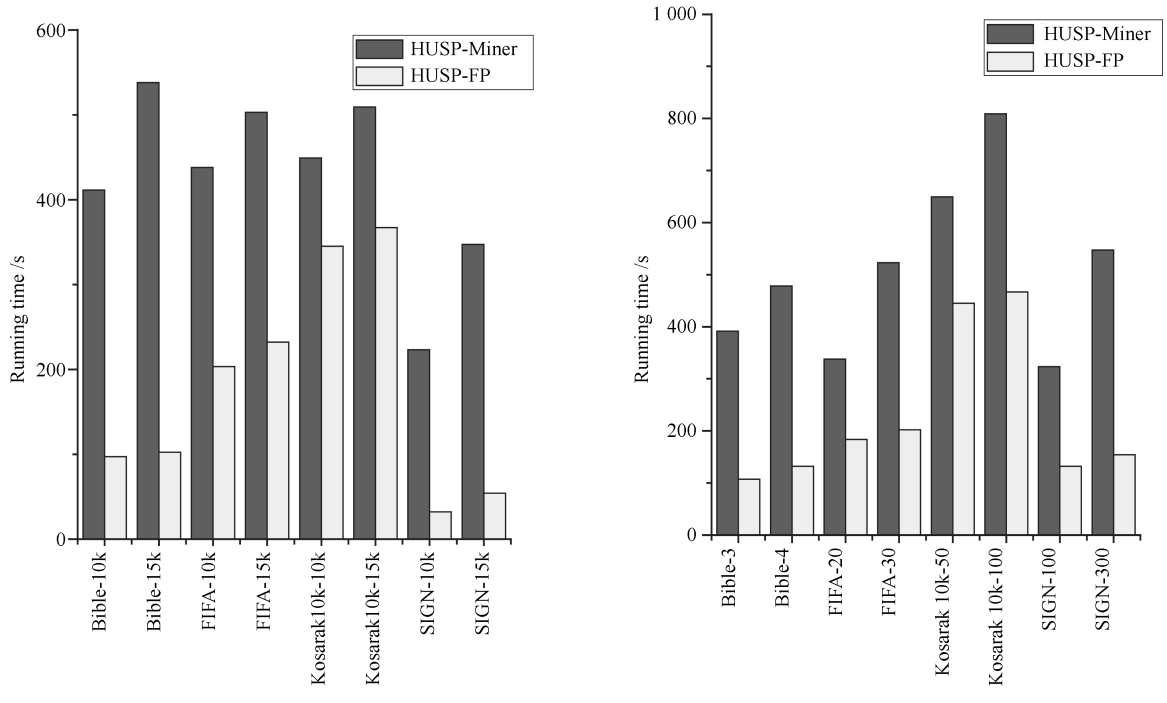




(c) Kosarak 10k　　(d) SIGN

图 8　数据流内存消耗

Fig. 8　Memory consumption based on data stream



(a) 不同数据容量下的对比
(a) Comparisons under different capacity

(b) 不同数据数量下的对比
(b) Comparisons under different quantity

图 9　HUSP-FP 和 HUSP-Miner 的对比评价

Fig. 9　Evaluation of HUSP-FP and HUSP-Miner

耗大量时间和空间, 下一步将针对此问题进行优化.

References

1 Michele D, Fabrizio B, Jacopo L, Jacopo R, Salvatore T, Federico. Sequential pattern mining for ICT risk assessment and management. *Journal of Logical and Algebraic Methods in Programming*, 2019, **102**(1): 1−16

2 Maryam A, Leyli M, Raffaela M. A sequential pattern mining model for application workload prediction in cloud environment. *Journal of Network and Computer Applications*,

2018, **105**(3): 21−62

3 Xue Yun, Li Tie-Chen, Liu Zhi-Wen, Pang Chao-Yi, Li Mei-Hang, Liao Zheng-Ling. A new approach for the deep order preserving submatrix problem based on sequential pattern mining. *International Journal of Machine Learning and Cybernetics*, 2018, **9**(2): 263−279

4 Trang V, Bay V, Bac L. Mining sequential patterns with itemset constraints. *Knowledge and Information Systems*, 2018, **57**(2): 311−330

5 Shou Zhen-Yu, Di Xuan. Similarity analysis of frequent sequential activity pattern mining. *Emerging Technologies*, 2018, **96**(11): 122−143

6 Bac L, Duy T, Van N, Quang M, Philippe F V. An efficient algorithm for hiding high utility sequential patterns. *International Journal of Approximate Reasoning*, 2018, **95**: 77−92

7 Liu J Q Zhang X X, Benjamin C, Li J Y, Farkhund I. Opportunistic mining of top-n high utility patterns. *Information Sciences*, 2018, **441**(5): 171−186

8 Song W, Rong K K. Mining high utility sequential patterns using maximal remaining utility. In: Proceedings of Data Mining and Big Data. Shanghai, China: Springer, 2018. 466−477

9 Ahmed, Chowdhury F. A novel approach for mining high-utility sequential patterns in sequence databases. *Electronics and Telecommunications Research Institute*, 2010, **32**(5): 676−686

10 Yin J F, Zheng Z G, Cao L B. Uspan: An efficient algorithm for mining high utility sequential patterns. In: Proceeding of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: IEEE, 2012. 660−666

11 Wang J Z, Huang J L, Chen Y C. On efficiently mining high utility sequential patterns. *Knowledge and Information Systems*, 2016, **49**(2): 597−627

12 Deng Z H, Ma S L, Li H. An efficient data structure for fast mining high utility itemset. *Computer Science*, 2015, **41**: 214−223

13 Zhang B B, Jerry L, Philippe F V. Mining of high utility-probability sequential patterns from uncertain databases. *Plos One*, 2017, **12**(7): e0180931

14 Zihayat M, Chen Y, An A. Memory-adaptive high utility sequential pattern mining over data streams. *Machine Learning*, 2017, **106**: 799−836

15 Zihayat M, Wu Cheng-Wei, An A. Efficiently mining high utility sequential patterns in static and streaming data. *Intelligent Data Analysis*, 2017, **21**: 103−135

16 Liu Meng-Chi, Qu Jun-Feng. Mining high utility itemsets without candidate generation. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management. Maui. USA: ACM, 2012. 55−64

17 Philippe F V, Wu Cheng-Wei, Souleymane Z, Vincent S. FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning. Intelligent Data Analysis, Switzerland: Springer, 2014. 83−92

18 Wang Le, Xiong Song-Quan, Chang Yan-Fen, Wang Shui. An algorithm for mining high utility patterns based on pattern-growth. *Acta Automatica Sinica*, 2015, **41**(9): 1616−1626
(王乐, 熊松泉, 常艳芬, 王水. 基于模式增长方式的高效用模式挖掘算法. 自动化学报, 2015, **41**(9): 1616−1626)

19 Philippe F V, Antonio G, Ted G. Spmf: a java open source pattern mining library. *Journal of Machine Learning Research*, 2014, **15**: 3389−3393

20 Zihayat M, Wu Cheng-Wei, An A. Efficient mining of high-utility sequential rule. In: Proceedings of Machine Learning and Data Mining. Hamburg, Germany: Springer, 2015. 157-171

21 Martínez B M, Martínezá F, Troncoso A, Mining quantitative association rules based on evolutionary computation and its application to atmospheric pollution. *Integrated Computer Aided Engineering*, 2010, **17**(3): 227−242

22 Vincent S T, Wu Cheng-Wei, Bai E S. UP-Growth: An efficient algorithm for high utility itemset mining. In: Proceedings of Knowledge Discovery and Data Mining. New York, USA: ACM, 2010. 253−262

唐辉军 宁波财经学院金融与信息学院副教授. 2008 年获得浙江工业大学硕士学位. 主要研究方向为数据挖掘技术. 本文通信作者.
E-mail: totti_2018@sina.com
(**TANG Hui-Jun** Associate professor at the School of Finance and Information, Ningbo University of Finance and Economics. He received his master degree from Zhejiang University of Techology in 2008. His main research interest is data mining. Corresponding author of this paper.)

王 乐 宁波财经学院数字技术与工程学院副教授. 2013 年获得大连理工大学博士学位. 主要研究方向为数据挖掘.
E-mail: wangleboro@163.com
(**WANG Le** Associate professor at the School of Digital Technology and Engineering, Ningbo University of Finance and Economics. She received her Ph. D. degree in computer application from Dalian University of Technology in 2013. Her main research interest is data mining.)

樊成立 宁波财经学院金融与信息学院副教授. 2007 年获得北京化工大学硕士学位. 主要研究方向为信息管理.
E-mail: 380577275@qq.com
(**FAN Cheng-Li** Associate professor at the School of Finance and Information, Ningbo University of Finance and Economics. He received his master degree from Beijing University of Chemical Technology in 2007. His main research interest is information management.)