基于频繁闭情节及其生成子的无冗余情节规则抽取

朱辉生[1),2)] 汪 卫[2)] 施伯乐[2)]

[1)](泰州师范高等专科学校 江苏 泰州 225300)

[2)](复旦大学计算机科学技术学院 上海 200433)

摘 要 情节规则挖掘旨在发现频繁情节之间的因果关联,已广泛应用于传感器数据处理、网络安全监控、金融证券管理、事务日志分析等众多领域.针对一个事件序列上的无冗余情节规则挖掘,提出了算法Extractor.该算法采用最小且非重叠发生的支持度定义和深度优先的搜索策略来发现频繁闭情节及其生成子,保证了频繁闭情节及其生成子的挖掘质量和挖掘效率;利用非生成子情节的Apriori性质,避免了冗余的情节生成子判断;直接由频繁闭情节及其生成子产生无冗余情节规则,提高了情节规则的生成质量和生成效率.所进行的实验证实了该情节规则抽取算法的有效性.

关键词 事件序列;频繁闭情节;情节生成子;情节规则

中图法分类号 TP311 **DOI号**: 10.3724/SP.J.1016.2012.00053

Extracting Non-Redundant Episode Rules Based on Frequent Closed Episodes and Their Generators

ZHU Hui-Sheng[1),2)] WANG Wei[2)] SHI Bai-Le[2)]

[1)](*Taizhou Teachers College*, *Taizhou*, *Jiangsu* 225300)

[2)](*School of Computer Science*, *Fudan University*, *Shanghai* 200433)

Abstract Aiming at discovering causal relationships between frequent episodes, episode rule mining has been broadly applied in many fields such as sensor data processing, network security monitoring, finance & securities managing, transaction log analyzing, and so on. To mine the non-redundant episode rules from an event sequence, an algorithm called Extractor is proposed in this paper. Extractor discovers all frequent closed episodes and their generators by employing the support definition of both minimal and non-overlapping occurrences and the depth-first search strategy, which assures the quality and efficiency of mining frequent closed episodes and their generators. Moreover, Extractor avoids redundant generator checking by utilizing the Apriori Property of non-generators. In addition, Extractor generates non-redundant episode rules directly from frequent closed episodes and their generators, which improves the quality and efficiency of generating episode rules. Experiments have proved the effectiveness of the proposed method.

Keywords event sequence; frequent closed episode; episode generator; episode rule

1 引 言

情节刻画了事件类型之间的紧随关系,而情节规则描述了频繁情节的因果关联.自Mannila等人[1]引入情节规则的概念以来,情节规则挖掘的问题一直是数据挖掘领域研究的热点之一,许多学者对此展开了深入研究,提出了TASA[2]、WinMiner[3]等经

收稿日期:2011-03-28;最终修改稿收到日期:2011-08-15.本课题得到国家"九七三"重点基础研究发展规划项目基金(2005CB321905)、国家自然科学基金(90818023,61003001,61103009)资助.**朱辉生**,男,1968年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为数据流与数据挖掘.E-mail: zhs@fudan.edu.cn. **汪 卫**,男,1970年生,教授,博士生导师,主要研究领域为数据挖掘、隐私保护、复杂结构数据管理等. **施伯乐**,男,1936年生,教授,博士生导师,主要研究领域为数据库与知识库.

典算法,并将它们应用于传感器数据处理[4]、网络安全监控[5]、金融证券管理[6]、软件规范挖掘[7]、事务日志分析[8]等众多领域.

然而,现有的情节规则挖掘算法存在以下不足:

(1)基于滑动窗口或最小发生来计算一个情节的支持度,致使频繁情节的挖掘质量不高;

(2)直接由频繁情节集产生所有的情节规则,导致规则数量过于庞大且存在冗余;

(3)尽管利用一些修剪技术来筛选冗余的情节规则,但这种后期的修剪处理增加了算法的时间代价.

那么,直接由频繁闭情节集来抽取情节规则,是否可以避免产生冗余规则呢?不幸的是,冗余问题依然存在.例如,假设事件序列 $ES1=\langle(A,1),(A,2),(B,3),(D,4),(A,5),(C,6),(B,7),(B,8),(E,9),(A,10),(B,11),(A,12),(C,13),(E,14),(F,15)\rangle$是某图书馆 Web 服务器上记载的一个涉及多个读者对多个文档的阅读序列,其中字母符号表示该图书馆提供的某个文档,数字表示某读者对相应文档的开始阅读时间.若使用最小且非重叠发生的情节支持度定义,则在支持度阈值 $min\_sup=2$ 时,$ES1$ 上所有的频繁情节和频繁闭情节分别如表 1 和表 2 所示,这些情节刻画了读者们的阅读行为,从而有助于图书馆人员发现文献之间的关联并向读者提供个性化的推荐服务.尽管频繁闭情节的个数远远少于频繁情节的个数,但直接由频繁闭情节集仍能产生如表 3 所示的 50 个情节规则,其中存在许多冗余规则.

表 1 *ES1* 上的频繁情节

情节	支持度	情节	支持度	情节	支持度
⟨*A*⟩	5	⟨*ABAE*⟩	2	⟨*BAC*⟩	2
⟨*AA*⟩	2	⟨*ABC*⟩	2	⟨*BACE*⟩	2
⟨*AAB*⟩	2	⟨*ABCE*⟩	2	⟨*BAE*⟩	2
⟨*AAC*⟩	2	⟨*ABE*⟩	2	⟨*BB*⟩	2
⟨*AACE*⟩	2	⟨*AC*⟩	2	⟨*BC*⟩	2
⟨*AAE*⟩	2	⟨*ACE*⟩	2	⟨*BCE*⟩	2
⟨*AB*⟩	3	⟨*AE*⟩	2	⟨*BE*⟩	2
⟨*ABA*⟩	2	⟨*B*⟩	2	⟨*C*⟩	2
⟨*ABAC*⟩	2	⟨*BA*⟩	2	⟨*CE*⟩	2
⟨*ABACE*⟩	2	⟨*BAB*⟩	2	⟨*E*⟩	2

表 2 *ES1* 上的频繁闭情节

情节	支持度	情节	支持度	情节	支持度
⟨*A*⟩	5	⟨*ABACE*⟩	2	⟨*BAB*⟩	2
⟨*AAB*⟩	2	⟨*B*⟩	4		
⟨*AB*⟩	3	⟨*BA*⟩	3		

表 3 *ES1* 上的情节规则

情节规则	支持度	置信度/%	情节规则	支持度	置信度/%
⟨*A*⟩→⟨*A*⟩	2	40	⟨*AB*⟩→⟨*A*⟩	2	67
⟨*A*⟩→⟨*B*⟩	3	60	⟨*AB*⟩→⟨*C*⟩	2	67
⟨*A*⟩→⟨*AB*⟩	2	40	⟨*AB*⟩→⟨*E*⟩	2	67
⟨*AA*⟩→⟨*B*⟩	2	100	⟨*AB*⟩→⟨*AC*⟩	2	67
⟨*A*⟩→⟨*C*⟩	2	40	⟨*AB*⟩→⟨*AE*⟩	2	67
⟨*A*⟩→⟨*E*⟩	2	40	⟨*AB*⟩→⟨*CE*⟩	2	67
⟨*A*⟩→⟨*BA*⟩	2	40	⟨*AB*⟩→⟨*ACE*⟩	2	67
⟨*A*⟩→⟨*BC*⟩	2	40	⟨*AA*⟩→⟨*C*⟩	2	100
⟨*A*⟩→⟨*BE*⟩	2	40	⟨*AA*⟩→⟨*E*⟩	2	100
⟨*A*⟩→⟨*AC*⟩	2	40	⟨*AA*⟩→⟨*CE*⟩	2	100
⟨*A*⟩→⟨*AE*⟩	2	40	⟨*AC*⟩→⟨*E*⟩	2	100
⟨*A*⟩→⟨*CE*⟩	2	40	⟨*BA*⟩→⟨*C*⟩	2	67
⟨*A*⟩→⟨*BAC*⟩	2	40	⟨*BA*⟩→⟨*E*⟩	2	67
⟨*A*⟩→⟨*BAE*⟩	2	40	⟨*BA*⟩→⟨*CE*⟩	2	67
⟨*A*⟩→⟨*BCE*⟩	2	40	⟨*BC*⟩→⟨*E*⟩	2	100
⟨*A*⟩→⟨*ACE*⟩	2	40	⟨*ABA*⟩→⟨*C*⟩	2	100
⟨*A*⟩→⟨*BACE*⟩	2	40	⟨*ABA*⟩→⟨*E*⟩	2	100
⟨*B*⟩→⟨*A*⟩	3	75	⟨*ABA*⟩→⟨*CE*⟩	2	100
⟨*B*⟩→⟨*C*⟩	2	50	⟨*ABC*⟩→⟨*E*⟩	2	100
⟨*B*⟩→⟨*E*⟩	2	50	⟨*AAC*⟩→⟨*E*⟩	2	100
⟨*B*⟩→⟨*AC*⟩	2	50	⟨*BAC*⟩→⟨*E*⟩	2	100
⟨*B*⟩→⟨*AE*⟩	2	50	⟨*ABAC*⟩→⟨*E*⟩	2	100
⟨*B*⟩→⟨*CE*⟩	2	50	⟨*B*⟩→⟨*B*⟩	2	50
⟨*B*⟩→⟨*ACE*⟩	2	50	⟨*B*⟩→⟨*AB*⟩	2	50
⟨*C*⟩→⟨*E*⟩	2	100	⟨*BA*⟩→⟨*B*⟩	2	67

1999 年,Pasquier 等人[9]引入了项集生成子的概念:一个项集称为生成子,当且仅当该项集不存在与其支持度相同的任何一个真子集.在一个交易数据库中,包含在相同交易集的项集的集合称为一个等价类[10],生成子和闭项集可以将交易数据库中所有项集合的幂集划分为不同的等价类,在每个这样的等价类中,项集生成子是最小的元素,而闭项集则是最大的元素.根据最小描述长度 MDL 原理,Li 等人[11]证明了项集生成子对于其所属的等价类具有最小的描述长度,可以避免对训练数据集的“过拟合”和抗噪音能力差等问题,因此,项集生成子比频繁闭项集更适于分类应用.基于频繁闭项集及其生成子可以直接产生具有最小前件和最大后件的无冗余关联规则基,典型算法有 Gen[12]、DPMiner[13]等.

依据交易数据库等价类和项集生成子的定义,Lo 等人[14]引入了序列数据库等价类和序列模式生成子的概念:一个序列数据库中包含在相同序列集的序列模式的集合称为一个等价类;一个序列模式称为生成子,当且仅当该序列模式不存在与其支持度相同的任何一个真子序列模式.基于 MDL 原理,他们同样证明了序列模式生成子比闭序列模式更适于解决分类问题,并且提出了由搜索空间压缩、发现生成子超集、筛选非生成子 3 个步骤组成的序列模式生成子挖掘算法 GenMiner.

然而，关联规则和序列模式的挖掘是针对一个交易数据库或序列数据库，相关的算法并不能直接应用于针对一个事件序列的情节规则挖掘. 目前尚无文献报道利用情节生成子来抽取无冗余情节规则的相关研究. 能否结合项集生成子及序列模式生成子的概念来定义情节生成子？情节生成子是否具有与项集生成子一样的 Apriori 性质？能否借鉴无冗余关联规则基的挖掘方法来抽取无冗余情节规则？这些正是本文研究的主要动机.

为此，我们引入了情节生成子的概念和非生成子情节的 Apriori 性质，并提出了一个情节规则的抽取算法 Extractor. 该算法采用最小且非重叠发生的支持度定义和深度优先的搜索策略来发现频繁闭情节及其生成子，保证了频繁闭情节及其生成子的挖掘质量和挖掘效率；利用非生成子情节的 Apriori 性质，避免了冗余的情节生成子判断；直接由频繁闭情节及其生成子产生无冗余情节规则，提高了情节规则的生成质量和生成效率. 理论分析和实验评估证明该算法能有效地抽取给定事件序列上的所有无冗余情节规则.

2　相关工作

关联规则的挖掘研究由来已久，针对如何控制由频繁项集产生的关联规则数量问题，Pasquier 等人[9]引入了项集生成子的概念：不存在与其支持度相同的任何一个真子集的项集，并提出了频繁闭项集的挖掘算法 A-Close；Bastide 等人[12]根据 Galois 连接的闭包定义了两种关联规则基：具有最小前件和最大后件的精确关联规则（置信度等于 100% 的关联规则）组成的无冗余精确关联规则基、具有最小前件和最大后件的近似关联规则（置信度小于 100% 的关联规则）组成的无冗余近似关联规则基，并提出了直接由频繁闭项集及其生成子来产生无冗余关联规则基的算法 Gen. 与 Gen 算法中的广度优先搜索策略不同，Li 等人[13]提出的算法 DPMiner 采用了深度优先的搜索策略，利用倒置的 FP 树来发现频繁闭项集及其生成子，进而产生交易数据库中具有 δ 差异的等价类.

为了拓展生成子在序列模式挖掘中的应用，Lo 等人[14]引入了序列数据库等价类和序列模式生成子的概念，并结合序列数据库等价类的特性，提出了序列模式生成子的挖掘算法 GenMiner. 该算法首先采用深度优先的搜索策略来创建存储了所有序列模式的前缀搜索树 PSL，然后通过遍历 PSL 得到包含所有序列模式生成子的超集 Gen-S，最后根据比较 Gen-S 中 2 个序列模式的支持度来筛选非生成子的序列模式.

然而，情节规则的挖掘有别于关联规则与序列模式的挖掘，前者挖掘的对象是一个事件序列，并非是一个事务数据库和序列数据库. Mannila 等人[1]首先引入了情节规则的概念：带有两个时间约束的情节规则可以表示为 $\beta[win1]\rightarrow\alpha[win2]$，其中 $\beta\subseteq\alpha$，即 β 是 α 的子情节，$win1$、$win2$ 分别是两个时间窗口. 该规则的语义是若情节 β 在一个不超过 $win1$ 的最小区间 $[t_s,t_e]$ 上发生，则情节 α 将会在不超过 $win2$ 的最小区间 $[t_s,t'_e]$ 上发生. 随后，Hatonen 等人[2]基于通信故障诊断的应用背景，提出了挖掘给定事件序列上所有情节规则的算法 TASA，该算法分为 3 个步骤：首先，基于广度优先的搜索策略和滑动窗口的情节支持度定义，通过多遍扫描事件序列，以候选、剪枝再测试的处理方式发现频繁情节；其次，由频繁情节集产生情节规则集；最后，使用 Pruning、Ordering、Grouping 等技术来筛选冗余的情节规则. 为了在产生情节规则的同时，能够发现使情节规则具有最大置信度的最小时间窗口，Meger 等人[3]于 2004 年提出了算法 WinMiner，该算法采用了深度优先的搜索策略和最小发生的情节支持度定义，只需单遍扫描事件序列，且不会产生任何候选情节，但其情节规则集的产生也是直接基于频繁情节集.

近年来，流数据的日益普及使得上述情节规则挖掘方法得到了广泛的应用，如基于传感器网络的大规模数据管理[4]、集误用检测和异常检测于一体的网络安全监控[5]、金融事件关联与股票价格趋势的分析[6]、基于程序验证与缺陷发现的软件规范挖掘[7]、动态系统中事件日志的概要分析[8]等.

针对现有情节规则挖掘算法存在的不足，本文提出了一个基于事件序列上的频繁闭情节及其生成子来抽取所有无冗余情节规则的算法 Extractor. 表 4 列出了本文提出的算法 Extractor 与现有相关算法的主要区别.

表 4　Extractor 与现有相关算法的区别

算法	搜索策略	情节的支持度计算方法	挖掘结果
TASA[2]	广度优先	滑动窗口（可能重叠）	所有情节规则
WinMiner[3]	深度优先	最小发生（可能重叠）	所有情节规则
Gen[12]	广度优先		无冗余关联规则基
DPMiner[13]	深度优先		交易数据库等价类
GenMiner[14]	深度优先		序列模式生成子集
Extractor（本文）	深度优先	最小且非重叠发生	无冗余情节规则集

3 预备知识

3.1 基本概念

定义 1(事件,事件序列,子序列). 给定一个事件类型集 $\varepsilon=\{E_1,E_2,\cdots,E_n\}$,一个事件就是一个二元组$(E,t)$,其中,$E\in\varepsilon$,$t$ 表示该事件的发生时间. 定义在 ε 上的一个事件序列 ES 是由若干事件按发生时间先后排列的序列,表示为 $ES=\langle(E_1,t_1),(E_2,t_2),\cdots,(E_s,t_s)\rangle$,其中 $t_i<t_j(1\leq i<j\leq s)$. 给定两个序列 ES 和 ES',若 $ES'=ES$ 或 ES' 是由 ES 删除某些事件后得到的剩余序列,则称 ES' 是 ES 的子序列.

定义 2(情节,子情节). 一个情节 α 是由若干事件类型组成的序列,表示为 $\alpha=\langle E_1E_2\cdots E_k\rangle$,其中元素 $E_i(1\leq i\leq k)\in\varepsilon$ 且对于所有的 i 和 $j(1\leq i<j\leq k)$满足 E_i 总是排列在 E_j 之前. 情节 α 中的元素个数称为 α 的长度,记为$|\alpha|$. 长度为 k 的情节也称为 k-情节. 给定情节 $\alpha=\langle E_1E_2\cdots E_m\rangle$ 和 $\beta=\langle E'_1E'_2\cdots E'_k\rangle$,若至少存在一个整数序列 $1\leq l_1<l_2<\cdots\leq l_k$,满足 $E'_i=E_{l_i}(1\leq i\leq k)$,则称 β 是 α 的一个子情节,或 α 是 β 的一个超情节,记为 $\beta\subseteq\alpha$ 或 $\alpha\supseteq\beta$. 若$\beta\subseteq\alpha$ 且$\beta\neq\alpha$,则称 β 是 α 的一个真子情节,或 α 是 β 的一个真超情节,记为 $\beta\subset\alpha$ 或 $\alpha\supset\beta$.

定义 3(前缀,后缀,串接,投影). 给定情节 $\alpha=\langle E_1E_2\cdots E_n\rangle$,若对于所有的 $i(1\leq i\leq m<n)$,满足 $E'_i=E_i$,则称情节 $\beta=\langle E'_1E'_2\cdots E'_m\rangle$ 是 α 的 m-前缀,记为 $prefix(\alpha,m)$. 给定情节 $\alpha=\langle E_1E_2\cdots E_k\rangle(k>1)$,称情节$\langle E_iE_{i+1}\cdots E_k\rangle(2\leq i\leq k)$是 α 的 i-后缀,记为 $suffix(\alpha,i)$. 给定情节 $\alpha=\langle E_1E_2\cdots E_m\rangle$ 和 $\beta=\langle E'_1E'_2\cdots E'_k\rangle$,则$\langle E_1E_2\cdots E_mE'_1E'_2\cdots E'_k\rangle$称为 α 和 β 的串接,记为 $concat(\alpha,\beta)$. 设 $\beta\subseteq\alpha$,j 是 β 在 α 中首次出现的结束位置,则从 α 中删除第 1 至第 j 个事件类型后剩余的情节称为 β 在 α 上的投影,记为 $project(\alpha,\beta)$.

定义 4(前向扩展,后向扩展,中间扩展). 设 k-情节 $\alpha=\langle E_1E_2\cdots E_k\rangle$,$e$ 为 1-情节,则$\langle E_1E_2\cdots E_ke\rangle$、$\langle eE_1E_2\cdots E_k\rangle$和$\langle E_1\cdots E_ieE_{i+1}\cdots E_k\rangle$分别称为 α 的前向扩展、后向扩展和中间扩展. 实际上,α 的前向扩展也是以 α 为前缀、e 为后缀进行情节增长后得到的情节 $concat(\alpha,e)$.

定义 5(发生). 给定事件序列 ES 和情节 $\alpha=\langle E_1E_2\cdots E_k\rangle$,若至少存在 ES 的一个子序列 $ES'=\langle(E_1,t_1),(E_2,t_2),\cdots,(E_k,t_k)\rangle$,满足 $t_i<t_{i+1}(1\leq i\leq k-1)$,则称 ES 上发生(或出现)了情节 α,区间$[t_1,t_k]$称为 α 在 ES 上的一次发生,其中 t_1 和 t_k 分别称为该发生的起始时间和终止时间.

定义 6(最小发生). 设$[t_s,t_e]$是情节 α 在事件序列 ES 上的一次发生,若在 ES 上不存在 α 的另一次发生$[t'_s,t'_e]$,使得 $t_s<t'_s$ 且 $t'_e\leq t_e$,或 $t_s\leq t'_s$ 且 $t'_e<t_e$,即$[t'_s,t'_e]\subset[t_s,t_e]$,则称$[t_s,t_e]$是 α 在 ES 上的一次最小发生.

定义 7(最小且非重叠发生). 设$[t_s,t_e]$和$[t'_s,t'_e]$是情节 α 在事件序列 ES 上的两次发生,若(1)$t_e<t'_s$或$t'_e<t_s$;且(2)$[t_s,t_e]$和$[t'_s,t'_e]$都是 α 的最小发生,则$[t_s,t_e]$和$[t'_s,t'_e]$是 α 在 ES 上的最小且非重叠发生.

定义 8(支持度). 情节 α 在事件序列 ES 上所有最小且非重叠发生组成的最大集合的基数称为 α 的支持度,记为 $\alpha.sup$.

定义 9(频繁情节,频繁闭情节,情节生成子). 给定支持度阈值 min_sup,若情节 α 的支持度大于等于 min_sup,则 α 是一个频繁情节. 若情节 α 是频繁的,且 α 的任何一个真超情节的支持度均不等于 α 的支持度,则 α 是一个频繁闭情节. 设 f 是一个闭情节,$g\subseteq f$,若 g 的支持度等于 f 的支持度,且 g 不存在与其支持度相同的任何一个真子情节,则 g 称为闭情节 f 的一个情节生成子.

定义 10(情节规则). 一个情节规则 γ 是一个五元组(l,r,s,c,w). 其中,l、r、s、c、w 分别称为 γ 的前件、后件、支持度、置信度和窗口宽度. 情节规则 γ 的支持度用于衡量该规则在事件序列上的统计特性,它等于情节 $concat(\gamma.l,\gamma.r)$的支持度;情节规则 γ 的置信度用于衡量该规则的可信程度,它等于情节 $concat(\gamma.l,\gamma.r)$的支持度与情节 $\gamma.l$ 的支持度比值. 情节规则 γ 的窗口宽度用于约束该规则的前件和后件必须在这个指定的时间区间内先后发生,它等于情节 $concat(\gamma.l,\gamma.r)$的所有最小且非重叠发生的平均时间.

定义 11(无冗余情节规则). 给定情节规则 $\gamma(l,r,s,c,w)$,若不存在情节规则 $\gamma'(l,r,s,c,w)$,使得 $\gamma.s=\gamma'.s$,$\gamma.c=\gamma'.c$,$\gamma'.l\subseteq\gamma.l$,$\gamma'.r\supseteq\gamma.r$,则称 γ 是一个无冗余情节规则,否则是一个冗余情节规则. 在所有支持度相同且置信度相同的情节规则中,无冗余情节规则同时具有最小的前件和最大的后件.

3.2 问题描述

给定事件序列 ES、支持度阈值 min_sup 和置

信度阈值 min_conf，则问题可以描述为：设计一个能够抽取 ES 上所有无冗余情节规则的算法，要求：(1) 单遍扫描 ES，发现频繁情节的过程中不产生候选情节；(2) 直接由发现的频繁闭情节及其生成子产生无冗余情节规则.

4　无冗余情节规则抽取算法 Extractor

4.1　算法描述

文献[15]提出了算法 MANEPI 以发现给定事件序列上的频繁情节，该算法采用深度优先的搜索策略和最小且非重叠发生的支持度定义，只需单遍扫描事件序列，且挖掘过程避免了候选情节的产生和情节发生的“过计数”问题，具有较高的挖掘效率和挖掘质量. 为此，本文提出的算法 Extractor 采用了与 MANEPI 一样的搜索策略和支持度定义，但与 MANEPI 不同的是，算法 Extractor 旨在发现频繁闭情节及其生成子，并进而直接由频繁闭情节及其生成子产生无冗余情节规则. 下面是 Extractor 的伪代码.

Extractor(ES, min_sup, min_conf).

Input: ES: an event $sequence$=$\langle(E_1,t_1),(E_2,t_2),\cdots,(E_n,t_n)\rangle$;
min_sup: a support threshold;
min_conf: a confidence threshold

Output: R: a set of all non-redundant episode rules

1. Let $FE=\varnothing$
2. Scan ES once to find all frequent 1-episodes and Store them with $clo=0$ and $gen=0$ to FE in lexicographical order
3. For each frequent 1-episode e do
4. $MineGrow(e)$
5. For each $\alpha\in FE$ do
6. $CheckClosure(\alpha)$
7. $CheckGenerator(\alpha)$
8. $FormRule(FE, min\_conf, R)$
9. Return R

Procedure $MineGrow(\alpha)$

Input: α: an episode to be grown

Objective: grow the existing frequent episode α

1. For each frequent 1-episode e do
2. Let $\beta=concat(\alpha,e)$
3. If $\beta.s\geq min\_sup$
4. Let $\beta.clo=\beta.gen=0$
5. Let $FE=FE\cup\beta$
6. If $\beta.s=\alpha.s$
7. Let $\beta.gen=-1$
8. If $\alpha.clo=0$
9. Let $\alpha.clo=-1$
10. $MineGrow(\beta)$

4.2　生成子检查

算法 Extractor 的关键步骤之一是如何确定一个情节是否为生成子. 项集生成子具有如下 Apriori 性质[14]：“若一个项集是一个频繁闭项集的生成子，则该生成子的真子集也是这个频繁闭项集的生成子”，这个性质可以用来加快项集生成子的挖掘. 那么，情节生成子是否也具有类似的 Apriori 性质：若一个情节是一个频繁闭情节的生成子，则该生成子的真子情节也是这个频繁闭情节的生成子？以表 1 和表 2 为例，情节$\langle AA\rangle$是频繁闭情节$\langle ABACE\rangle$的一个生成子，因为$\langle AA\rangle$的支持度等于$\langle ABACE\rangle$的支持度，且$\langle AA\rangle$不存在与其具有相同支持度的任何真子情节. 然而，$\langle AA\rangle$的一个真子情节$\langle A\rangle$却不是$\langle ABACE\rangle$的生成子，这是因为$\langle A\rangle$的支持度并不等于$\langle ABACE\rangle$的支持度.

尽管情节生成子不具有 Apriori 性质，但是一些非生成子情节却具有引理 1 所示的 Apriori 性质，它可以用来加快情节生成子的发现.

引理 1.　给定一个情节 α，如果存在 α 的一个真子情节 β，使得 $\beta.mano=\alpha.mano$，则情节 α 及以其为前缀的其它任何情节都是非生成子情节.

证明.　(1) 根据定义 9 可知，情节 α 是一个非生成子情节；(2) 因 $\alpha.mano=\beta.mano$，故对于已知事件序列上的任何一个频繁 1-情节 e 而言，只要能以 α 为前缀、e 为后缀进行情节增长得到 $concat(\alpha,e)$，也就能以 β 为前缀、e 为后缀进行情节增长而得到 $concat(\beta,e)$，并使得 $concat(\beta,e).mano=concat(\alpha,e).mano$，即 $concat(\beta,e)$ 的支持度等于 $concat(\alpha,e)$ 的支持度. 又因为 β 是 α 的真子情节，所以 $concat(\beta,e)$ 也是 $concat(\alpha,e)$ 的真子情节，从而根据定义 9 可知，以 α 为前缀的 $concat(\alpha,e)$ 不是生成子情节.　证毕.

以表 2 为例，因为$\langle ABA\rangle.mano=\langle AA\rangle.mano$且$\langle ABA\rangle\supset\langle AA\rangle$，所以情节$\langle ABA\rangle$及以其为前缀的任何其它情节(如$\langle ABACE\rangle$)都是非生成子情节.

由于算法 Extractor 在过程 MineGrow 中已通过比较频繁情节 α 与其前向扩展的支持度来检查 α 的当前前向扩展是否为生成子，所以在得到频繁情节集 FE 后，针对尚未确定为生成子的频繁情节 β 而言，我们只要通过比较 β 与 β 的其它真子情节的支持度就可以确定 β 是否为生成子. 下面是生成子

检查的伪代码.

```
Procedure CheckGenerator(α)
Input: α: a frequent episode
Objective: check whether episode α is a generator
1. If α.gen=-1
2.   If ∃β s.t. β.mano=α.mano and β⊂α
3.     Set the generators of α and any episode with α
       as its prefix to -1
4. Else
5.   For i=1 to |α|-1 do
6.     Let β=⟨E_1…E_{i-1}E_{i+1}…E_{|α|}⟩
7.     If β∈FE and β.s=α.s
8.       Let α.gen=-1
9.       Break
10. Return
```

4.3 规则产生

根据置信度的不同,无冗余情节规则可以分为如下两个类别.

类别 1. 无冗余精确情节规则,即置信度等于100%的无冗余情节规则. 设 C 为频繁闭情节集,G_f 为 C 中频繁闭情节 f 的所有情节生成子组成的集合,则所有的无冗余精确情节规则可以表示为集合:$RE=\{(l,r,s,c,w)\mid l\in G_f\wedge r=project(f,l)\wedge f\in C\wedge l\neq f\}$,式中:$l$ 是频繁闭情节 f 的一个情节生成子,约束条件 $l\neq f$ 保证了一个情节规则的前件不能为空.

定理 1. 无冗余精确情节规则集蕴含了所有精确情节规则的信息.

证明. 设 γ 是一个由频繁闭情节 f 得到的精确情节规则,因为 $\gamma.c=100\%$,所以 $\gamma.s=concat(\gamma.l,\gamma.r).s=(\gamma.l).s$. 对于 f 而言,必然存在一个无冗余精确情节规则 γ',满足 $\gamma'.l$ 是 f 的情节生成子,$\gamma'.s=concat(\gamma'.l,\gamma'.r).s=(\gamma'.l).s=f.s$,且 $\gamma'.l\subseteq\gamma.l\subset concat(\gamma.l,\gamma.r)\subseteq f$,即表示 γ 的前件和后件可由 γ' 得到. 因为 $\gamma'.l\subseteq\gamma.l\subset concat(\gamma.l,\gamma.r)\subseteq f$,所以 $f.s\leq concat(\gamma.l,\gamma.r).s\leq(\gamma.l).s\leq(\gamma'.l).s$. 由于 $f.s=(\gamma'.l).s=\gamma'.s$,所以 $\gamma.s=concat(\gamma.l,\gamma.r).s=\gamma'.s$,即表示 γ 的支持度可以由 γ' 得到. 由于 $\gamma.l\supseteq\gamma'.l$,$\gamma.r\subseteq\gamma'.r$,且情节规则主要用于在前件发生时预测后件的发生情况,所以 γ' 的窗口宽度蕴含了 γ 的窗口宽度. 证毕.

类别 2. 无冗余近似情节规则,即置信度小于100%的无冗余情节规则. 设 C、G 分别为频繁闭情节集和情节生成子集,$\delta(\alpha)$ 表示一个与频繁情节 α 具有相同支持度的频繁闭情节,则所有的无冗余近似情节规则可以表示为集合:$RA=\{(l,r,s,c,w)\mid l\in G\wedge r=project(f,l)\wedge f\in C\wedge\delta(l)\subset f\}$,式中:$l$ 是频繁闭情节 $\delta(l)$ 的情节生成子,$\delta(l)$ 是另一个频繁闭情节 f 的真子情节.

定理 2. 无冗余近似情节规则集蕴含了所有近似情节规则的信息.

证明. 设 γ 是一个近似情节规则,因为 $\gamma.c<100\%$,所以 $\delta(\gamma.l)\subset\delta(concat(\gamma.l,\gamma.r))$. 对于 $\gamma.l$ 和 $concat(\gamma.l,\gamma.r)$ 而言,存在两个情节生成子 g_1 和 g_2,满足 $g_1\subseteq\gamma.l\subseteq\delta(\gamma.l)=\delta(g_1)$,$g_2\subseteq concat(\gamma.l,\gamma.r)\subseteq\delta(concat(\gamma.l,\gamma.r))=\delta(g_2)$. 因为 $\gamma.l\subset concat(\gamma.l,\gamma.r)$,所以 $\gamma.l\subset\delta(g_1)\subset concat(\gamma.l,\gamma.r)\subseteq\delta(g_2)$,这说明必然存在一个无冗余近似情节规则 γ' 满足 $\gamma'.l=g_1$,$\gamma'.r=project(\delta(g_2),g_1)$. 因为 $g_1\subseteq\gamma.l\subseteq\delta(g_1)\subset concat(\gamma.l,\gamma.r)\subseteq\delta(g_2)$,所以 γ 的前件和后件可由 γ' 得到. 由于 $\gamma.s=concat(\gamma.l,\gamma.r).s=\delta(g_2).s=\gamma'.s$,所以 γ 的支持度可由 γ' 得到. 因 $g_1\subseteq\gamma.l\subseteq\delta(g_1)$,故 $(g_1).s=(\gamma.l).s=\delta(g_1).s$,$\gamma.c=concat(\gamma.l,\gamma.r).s/(\gamma.l).s=\delta(g_2).s/\delta(g_1).s=concat(\gamma'.l,\gamma'.r).s/(\gamma'.l).s=\gamma'.c$,表示 γ 的置信度可由 γ' 得到. 证毕.

下面是规则产生的伪代码.

```
Precedure FormRule(FE,min_conf,R)
Input: FE: a set of frequent episodes after checking
         closures and generators;
       min_conf: a confidence threshold;
       R: a set of non-redundant episode rules
Objective: generate all non-redundant episode rules
           directly from FE
1. Let R=∅
2. For each f∈FE with f.clo=0 do
3.   For each g∈FE with g.gen=0 do
4.     If g⊂f
5.       Let r=project(f,g)
6.       Let α=concat(g,r)
7.       If α.s/g.s≥min_conf
8.         Let R=R∪(g, r, α.s, α.s/g.s, α.w)
```

4.4 运行实例

设给定的事件序列为 $ES1$,$min\_sup=2$,$min\_conf=0$,现使用算法 Extractor 发现 $ES1$ 上所有的无冗余情节规则. 挖掘时得到的情节生成子和无冗余情节规则分别如表 5 和表 6 所示.

表 5 *ES1* 上的情节生成子

情节生成子	支持度	情节生成子	支持度
⟨*A*⟩	5	⟨*AA*⟩	2
⟨*B*⟩	4	⟨*AB*⟩	3
⟨*C*⟩	2	⟨*BA*⟩	3
⟨*E*⟩	2	⟨*BB*⟩	2

表 6　*ES*1 上的无冗余情节规则

前件	后件	支持度	置信度/%	窗口宽度
⟨*AA*⟩	⟨*B*⟩	2	100	5
⟨*AA*⟩	⟨*CE*⟩	2	100	7
⟨*C*⟩	⟨*E*⟩	2	100	3
⟨*A*⟩	⟨*AB*⟩	2	40	7
⟨*A*⟩	⟨*B*⟩	3	60	3
⟨*A*⟩	⟨*BACE*⟩	2	40	7
⟨*B*⟩	⟨*ACE*⟩	2	50	6
⟨*AB*⟩	⟨*ACE*⟩	2	67	7
⟨*BA*⟩	⟨*CE*⟩	2	67	6
⟨*B*⟩	⟨*A*⟩	3	75	3
⟨*B*⟩	⟨*AB*⟩	2	50	5
⟨*BA*⟩	⟨*B*⟩	2	67	5

由表 3 和表 6 可以看出，相对于所有情节规则，算法 Extractor 抽取的无冗余情节规则不仅在数据量上大大减少，而且没有丢失其它情节规则的信息.

4.5　算法复杂度分析

设 L 为事件序列为 ES 的长度，ε 为 ES 中的事件类型集，FE 为 ES 上所有频繁情节组成的集合，则算法 Extractor 的复杂度分析如下.

定理 3.　Extractor 的时间复杂度为 $O(|FE|\cdot|\varepsilon|\cdot L)$.

证明.　情节增长、闭合性检查、生成子检查、规则产生是 Extractor 的主要时间代价.

(1) 完成一次以已存在的频繁情节 α 为前缀、频繁 1-情节 e 为后缀的情节增长操作，只需单遍扫描 $mo(\alpha)$ 和 $mo(e)$，其时间复杂度为 $O(L)$. 而 Extractor 只需以 FE 中的每个情节为前缀进行情节增长，增长的次数至多为频繁 1-情节的个数，其上界为 $|\varepsilon|$，因此，Extractor 为完成全部的情节增长操作所需的时间复杂度为 $O(|FE|\cdot|\varepsilon|\cdot L)$.

(2) 在闭合性检查之前，由所有频繁情节组成的集合 FE 中至多存在 $|FE|$ 个闭合性待定的频繁情节，对于每一个这样的情节 α 而言，确定其闭合性在最坏的情况下需要比较 α 的支持度与其 $2|\varepsilon|$（$|\varepsilon|$ 为频繁 1-情节个数的上界）个中间扩展及后向扩展的支持度，而每一个这样的扩展又至多出现在情节 α 中的任一位置上，所以完成所有闭合性检查的最坏时间复杂度为 $O(|FE|\cdot|\varepsilon|\cdot|\alpha|)\leq O(|FE|\cdot|\varepsilon|\cdot L)$.

(3) 生成子检查的时间消耗与闭合性检查相同，其最坏时间复杂度也为 $O(|FE|\cdot|\varepsilon|\cdot L)$.

(4) 频繁闭情节的个数上界为 $|FE|$，所以规则产生过程所需的最坏时间复杂度为 $O(|FE|\cdot|FE|)\leq O(|FE|\cdot|\varepsilon|\cdot L)$.

综上，算法的时间复杂度为 $O(|FE|\cdot|\varepsilon|\cdot L)$.

证毕.

定理 4.　Extractor 的空间复杂度为 $O(|FE|^2)$.

证明.　算法需要维护所有频繁闭情节及情节规则，所以空间复杂度为 $O(|FE|^2)$.　证毕.

5　实验评估

我们通过 9 组实验对比了算法 Extractor、TASA[2] 和 WinMiner[3] 的时空性能和挖掘质量. 为了能够客观地评价 3 个算法，我们对 TASA 和 WinMiner 进行了适当调整，即均使用与 Extractor 一样的最小且非重叠发生的情节支持度定义，修改后的 TASA 和 WinMiner 分别表示为 TASA_mano 和 WinMiner_mano. 实验采用的硬件环境为 2 GHz Intel(R) Core(TM) 2 Duo CPU，内存 2GB，操作系统为 Windows XP，程序采用 Java 实现.

5.1　数据集

对于合成数据集，我们首先使用 IBM 合成数据生成器 Quest Market-Basket 的修改版生成了每个交易为单个项的交易序列，通过设置 $D=0.001$，$C=300000$，$N=20$，$S=300000$，其中参数 D 表示交易序列的个数（单位为 1000），C 表示每个交易序列中交易的平均个数，N 表示所有交易项的类型种数（单位为 1000），S 为最长交易序列中交易的平均个数，这样就得到了一个 20 000 种交易项类型上的由 300 000 个交易组成的交易序列. 然后，为该交易序列中的每个交易依次赋上一个连续的正整数以作为每个交易发生的时间戳，这样，我们就构造了一个 20 K 种事件类型上的由 300 K 个事件组成的事件序列.

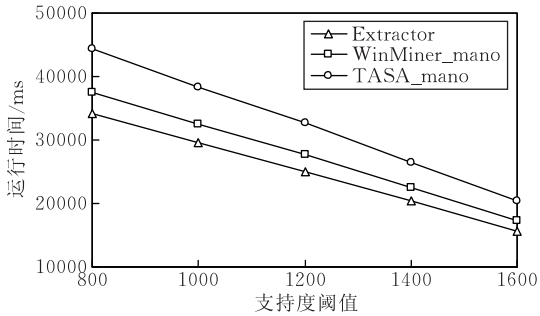
对于真实数据集，考虑到作为国内最具影响力的知识传播与数字化学习平台，中国知网 CNKI① 为全社会提供了最丰富、最全面的文献资源，为了能够发现 CNKI 中相关文献之间的引用关系，并为广大学者展开相关研究提供个性化的推荐服务，我们选用了 CNKI 的一个 Web 服务器上从 2010 年 11 月 1 日至 2010 年 11 月 30 日的日志数据，该日志数据包括了相关读者对 132 885 种不同文献的 211 665 个阅读序列.

5.2　实验结果

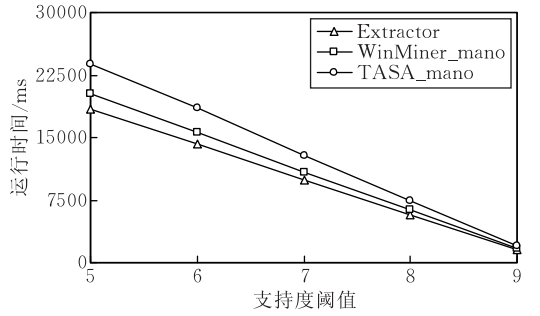
实验 1.　运行时间与支持度阈值的关系. 选择 300 K 合成数据集和 30 天真实数据集，在设定置信度阈值为 60% 的前提下，通过改变支持度阈值，得

① http://www.cnki.net

到了如图 1 所示的 3 个算法在两个数据集上的运行时间(以 ms 为单位).



(a) 合成数据集



(b) 真实数据集

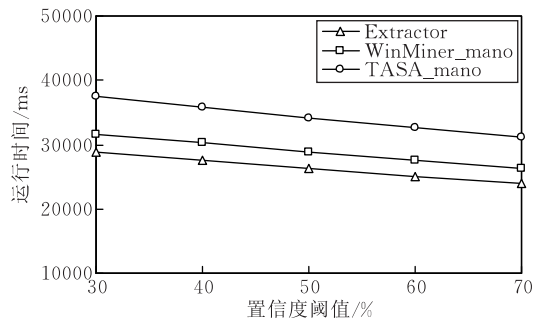
图 1 运行时间与支持度阈值的关系

可以看出,随着支持度阈值的减小,3 个算法的运行时间都线性增加,但 Extractor 要优于 TASA_mano 和 WinMiner_mano,且它们的运行时间之差随着支持度阈值的减小而不断增加,导致上述事实的主要原因是:TASA_mano 和 WinMiner_mano 旨在发现所有的情节规则,并分别采用了广度优先和深度优先的搜索策略,而 Extractor 基于深度优先的搜索策略,并内嵌了一个优化技术,旨在发现无冗余的情节规则.

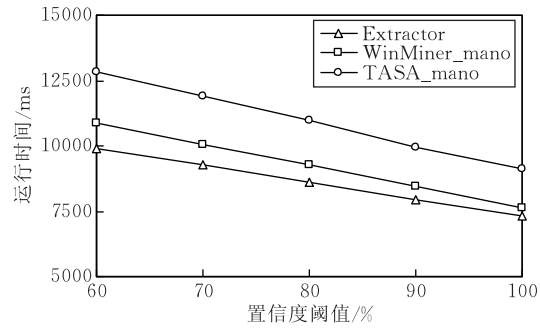
实验 2. 运行时间与置信度阈值的关系. 选择 300 K 合成数据集和 30 天真实数据集,在设定这两个数据集的支持度阈值分别为 800 和 7 的前提下,通过改变置信度阈值,得到了如图 2 所示的 3 个算法在两个数据集上的运行时间(以 ms 为单位).

可以看出,随着置信度阈值的减小,3 个算法的运行时间也在线性增加,但 Extractor 要优于 TASA_mano 和 WinMiner_mano,原因同实验 1 的解释. 我们同时还观察到:相对于支持度而言,置信度对 3 个算法运行时间的影响要相对平稳,这是因为 3 个算法均是首先基于支持度阈值发现频繁情节,然后再基于置信度阈值来产生情节规则.

实验 3. 运行时间与序列长度的关系. 首先从 300K 合成数据集中选择前 100 K、前 150 K、前 200 K、前 250 K 个事件作为 4 个合成子序列;再从 30 天真实数据集中选择前 6 天、前 12 天、前 18 天、前 24 天的事件作为 4 个真实子序列;然后通过设定合成数据集和真实数据集的支持度阈值分别为 800 和 7,置信度阈值均为 60%,得到了如图 3 所示的序列长度对算法时间性能的影响.

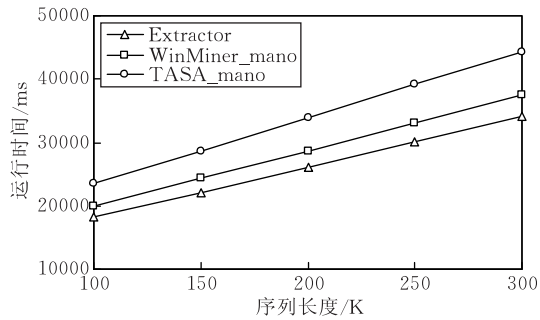


(a) 合成数据集

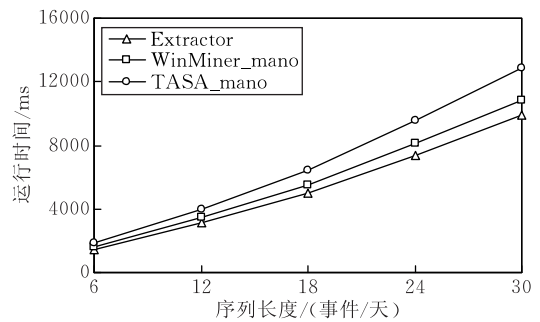


(b) 真实数据集

图 2 运行时间与置信度阈值的关系



(a) 合成数据集



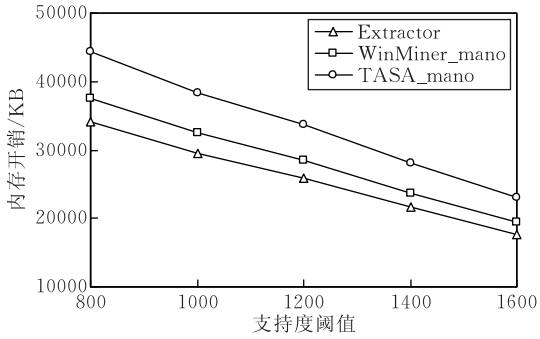
(b) 真实数据集

图 3 运行时间与序列长度的关系

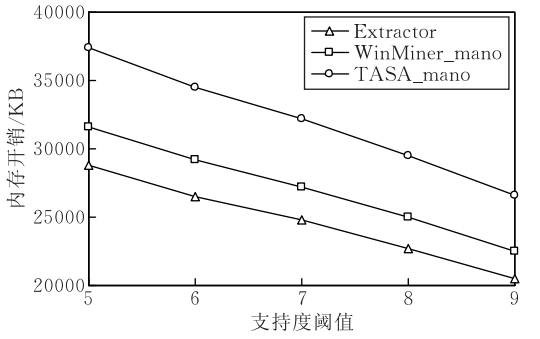
可以看出,3 个算法的运行时间都随着序列长

度的增加而线性增加，且 3 个算法的时间性能对比同实验 1，产生这些现象的原因与实验 1 的解释相同.

实验 4. 内存开销与支持度阈值的关系. 选择 300 K 合成数据集和 30 天真实数据集，在设定置信度阈值为 60% 的前提下，通过改变支持度阈值，得到了得到了如图 4 所示的 3 个算法在两个数据集上的内存开销(以 KB 为单位).



(a) 合成数据集



(b) 真实数据集

图 4　内存开销与支持度阈值的关系

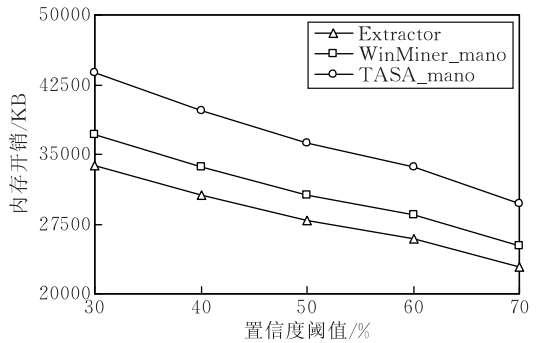
可以看出，随着支持度阈值的减小，3 个算法的内存开销都在线性增加，但 Extractor 要优于 TASA_mano 和 WinMiner_mano，原因与实验 1 的解释相同.

实验 5. 内存开销与置信度阈值的关系. 选择 300K 合成数据集和 30 天真实数据集，在设定它们的支持度阈值分别为 800 和 7 的前提下，通过改变置信度阈值，得到了如图 5 所示的算法在两个数据集上的内存开销(以 KB 为单位).

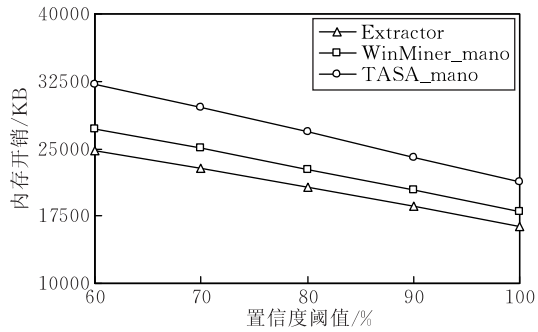
可以看出，随着置信度阈值的减小，算法的内存开销都在线性增加，但 Extractor 要优于 TASA_mano 和 WinMiner_mano，原因与实验 1 的解释相同.

实验 6. 内存开销与序列长度的关系. 通过设定合成数据集和真实数据集的支持度阈值分别为 800 和 7，置信度阈值均为 60%，得到了如图 6 所示的序列长度对 3 个算法内存开销的影响.

可以看出，3 个算法的内存开销都随着序列长度的

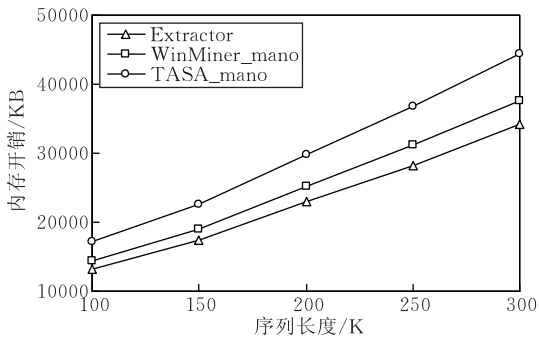


(a) 合成数据集

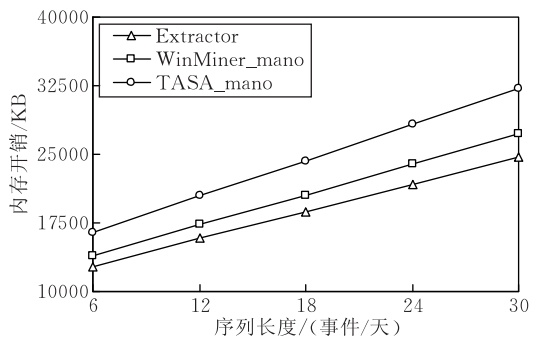


(b) 真实数据集

图 5　内存开销与置信度阈值的关系



(a) 合成数据集



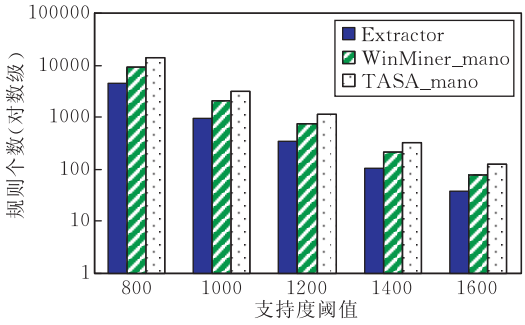
(b) 真实数据集

图 6　内存开销与序列长度的关系

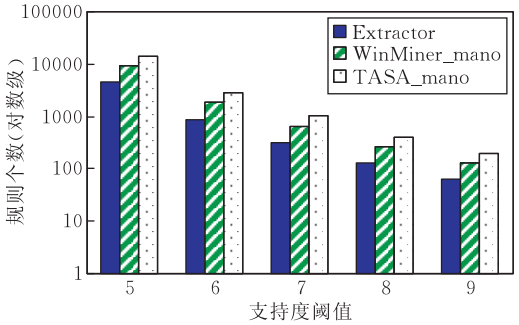
增加而线性增加，但 Extractor 要优于 TASA_mano 和 WinMiner_mano，原因与实验 1 的解释相同.

实验 7. 规则个数与支持度阈值的关系. 选择 300K 合成数据集和 30 天真实数据集，在设定置信度阈值为 60% 的前提下，通过改变支持度阈值，得

到了如图 7 所示的 3 个算法在两个数据集上发现的情节规则个数.



(a) 合成数据集



(b) 真实数据集

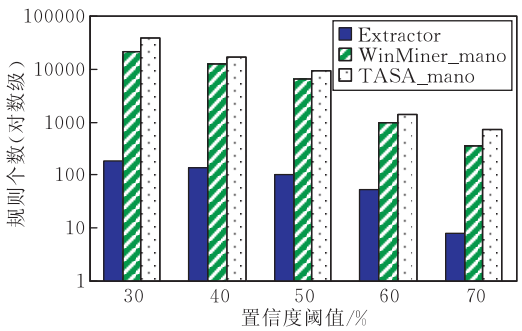
图 7 规则个数与支持度阈值的关系

可以看出,随着支持度阈值的减小,3 个算法均发现了更多的情节规则,这是因为支持度阈值越小,基于越多的频繁闭情节和情节生成子产生的情节规则也就更多. 我们同时还观察到:Extractor 发现的规则个数远少于 TASA_mano 和 WinMiner_mano,这是因为 Extractor 产生的是无冗余情节规则,而 TASA_mano 和 WinMiner_mano 产生的是所有情节规则.

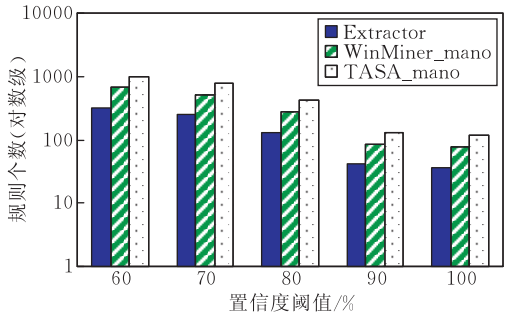
实验 8. 规则个数与置信度阈值的关系. 选择 300K 合成数据集和 30 天真实数据集,在设定这两个数据集的支持度阈值分别为 800 和 7 的前提下,通过改变置信度阈值,得到了如图 8 所示的算法在两个数据集上发现的情节规则个数.

可以看出,随着置信度阈值的减小,3 个算法均发现了更多的情节规则,但 Extractor 的规则个数远少于 TASA_mano 和 WinMiner_mano,原因与实验 7 的解释相同.

实验 9. 规则个数与序列长度的关系. 通过设定合成数据集和真实数据集的支持度阈值分别为 800 和 7,置信度阈值均为 60%,得到了如图 9 所示的序列长度对算法所发现情节规则个数的影响. 可以看出,随着序列长度的增加,3 个算法均发现了更多的情节规则,但 Extractor 的规则个数远少于 TASA_mano 和 WinMiner_mano,原因与实验 7 的解释相同.

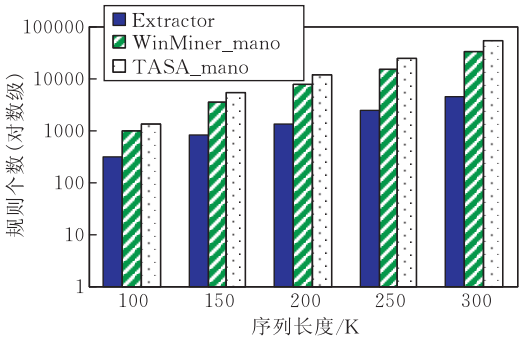


(a) 合成数据集

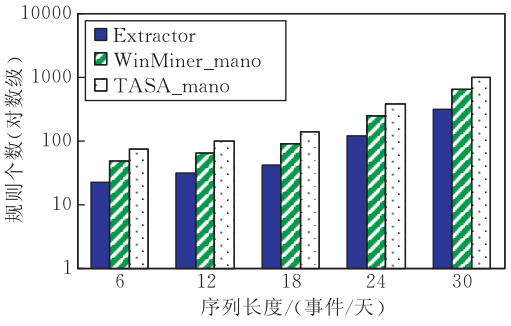


(b) 真实数据集

图 8 规则个数与置信度阈值的关系



(a) 合成数据集



(b) 真实数据集

图 9 规则个数与序列长度的关系

6 结 论

现有的情节规则挖掘算法存在挖掘质量不高、挖掘效率较低等不足. 本文提出的情节规则挖掘算法 Extractor 基于最小且非重叠发生来计算一个情

节的支持度，避免了情节发生的“过计数”问题，保证了频繁闭情节及其生成子的挖掘质量；采用深度优先的搜索策略并结合非生成子情节的 Apriori 性质，加快了频繁闭情节及其生成子的挖掘过程；直接由频繁闭情节及其生成子产生无冗余情节规则，提高了情节规则的生成质量和生成效率. 理论分析和实验评估证明算法 Extractor 能够有效地抽取给定事件序列上的所有无冗余情节规则.

参　考　文　献

[1] Mannila H, Toivonen H, Verkamo A I. Discovering frequent episodes in sequences//Proceedings of the 1st ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Montreal, Canada, 1995: 210-215

[2] Hatonen K, Klemettinen M, Mannila H, Ronkainen P, Toivonen H. Knowledge discovery from telecommunication network alarm databases//Proceedings of the 12th IEEE International Conference on Data Engineering. New Orleans, Louisiana, 1996: 115-122

[3] Meger N, Rigotti C. Constraint-based mining of episode rules and optimal window sizes//Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases. Pisa, Italy, 2004: 313-324

[4] Patnaik D, Marwah M, Sharma R, Ramakrishnan N. Sustainable operation and management of data center chillers using temporal data mining//Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Paris, France, 2009: 1305-1313

[5] Hwang K, Cai M, Chen Y, Qin M. Hybrid intrusion detection with weighted signature generation over anomalous internet episodes. IEEE Transactions on Dependable and Secure Computing, 2007, 4(1): 41-55

[6] Ng A, Fu A. Mining frequent episodes for relating financial events and stock trends//Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Seoul, Korea, 2003: 27-39

[7] Lo D, Khoo S, Liu C. Efficient mining of recurrent rules from a sequence database//Proceedings of the 13th International Conference on Database Systems for Advanced Applications. New Delhi, India, 2008: 67-83

[8] Wang P, Wang H, Liu M, Wang W. An algorithmic approach to event summarization//Proceedings of the ACM SIGMOD International Conference on Management of Data. Indianapolis, Indiana, USA, 2010: 183-194

[9] Pasquier N, Bastide Y, Taouil R, Lakhal L. Discoving frequent closed itemsets for association rules//Proceedings of the 7th International Conference on Database Theory. Jerusalem, Israel, 1999: 398-416

[10] Bastide Y, Taouil R, Pasquier N, Stumme G, Lakhal L. Mining frequent patterns with counting inference. SIGKDD Explorations, 2000, 2(2): 66-75

[11] Li J, Li H, Wong L, Pei J, Dong G. Minimum description length principle: Generators are preferable to closed patterns//Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference. Boston, Massachusetts, USA, 2006: 409-414

[12] Bastide Y, Pasquier N, Taouil R, Stumme G, Lakhal L. Mining minimal non- redundant association rules using frequent closed itemsets//Proceedings of the 1st International Conference on Computational Logic. London, UK, 2000: 972-986

[13] Li J, Liu G, Wong L. Mining statistically important equivalence classes and delta-discriminative emerging patterns//Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. San Jose, California, USA, 2007: 430-439

[14] Lo D, Khoo SC, Li J. Mining and ranking generators of sequential patterns//Proceedings of the SIAM International Conference on Data Mining. Atlanta, Georgia, USA, 2008: 553-564

[15] Zhu H, Wang P, He X, Li Y, Wang W, Shi B. Efficient episode mining with minimal and non-overlapping occurrences//Proceedings of the 10th IEEE International Conference on Data Mining. Sydney, Australia, 2010: 1211-1216

ZHU Hui-Sheng, born in 1968, Ph. D., associate professor. His research interests include data stream and data mining.

WANG Wei, born in 1970, professor, Ph. D. supervisor. His research interests include data mining, privacy protection, and complex structure data management.

SHI Bai-Le, born in 1936, professor, Ph. D. supervisor. His research interests include database and knowledge base.

Background

This work was supported by the National Basic Research Program (973 Program) of China under grant No. 2005CB321905, the National Natural Science Foundation of China under grant Nos. 90818023, 61003001, 61103009. The projects are involved with the key technology of extracting representative episode rules from an event sequence. The research group has been working on many aspects of the projects since 2006, and many good papers have been published in worldwide conferences and transactions, such as SIGMOD, SIGKDD, ICDM, TKDE, JIIS, KAIS, JCST et al.

Extracting episode rules has received intensive research in data mining due to its broad applications such as sensor data processing, network security monitoring, finance & securities managing, transaction log analyzing, and so on. Different from existing related methods, the algorithm Extractor proposed in this paper can discover all frequent closed episodes and their generators by employing the support definition of both minimal and non-overlapping occurrences and the depth-first search strategy, which assures the quality and efficiency of mining frequent closed episodes and their generators. Moreover, Extractor avoids redundant generator checking by utilizing the Apriori Property of non-generators. In addition, Extractor generates non-redundant episode rules directly from frequent closed episodes and their generators, which improves the quality and efficiency of generating episode rules. Theoretical analysis and experimental evaluation demonstrate algorithm Extractor can effectively extract all non-redundant episode rules from the given event sequence.