·先进计算与数据处理· 文章编号：1000-3428(2021)02-0118-08 文献标志码：A 中图分类号：TP391

一种基于MDL的日志序列模式挖掘算法

杜诗晴[1]，王 鹏[2]，汪 卫[2]
（1.复旦大学 软件学院，上海 201203；2.复旦大学 计算机科学技术学院，上海 201203）

摘 要：日志数据是互联网系统产生的过程性事件记录数据，从日志数据中挖掘出高质量序列模式可帮助工程师高效开展系统运维工作。针对传统模式挖掘算法结果冗余的问题，提出一种从时序日志序列中挖掘序列模式(DTS)的算法。DTS采用启发式思路挖掘能充分代表原序列中事件关系和时序规律的模式集合，并将最小描述长度准则应用于模式挖掘，设计一种考虑事件关系和时序关系的编码方案，以解决模式规模爆炸问题。在真实日志数据集上的实验结果表明，与SQS、CSC与ISM等序列模式挖掘算法相比，该算法能高效挖掘出含义丰富且冗余度低的序列模式。

关键词：数据挖掘；日志分析；事件关系；最小描述长度准则；序列模式

开放科学(资源服务)标志码(OSID)：

中文引用格式：杜诗晴，王鹏，汪卫.一种基于MDL的日志序列模式挖掘算法[J].计算机工程，2021，47(2)：118-125.
英文引用格式：DU Shiqing，WANG Peng，WANG Wei.A MDL-based pattern mining algorithm for log sequences[J].Computer Engineering，2021，47(2)：118-125.

A MDL-based Pattern Mining Algorithm for Log Sequences

DU Shiqing[1]，WANG Peng[2]，WANG Wei[2]
（1.Software School，Fudan University，Shanghai 201203，China；
2.School of Computer Science，Fudan University，Shanghai 201203，China）

【Abstract】Logs contain rich information about procedural events generated in Internet systems，and the mining of high-quality sequence modes from log data can improve the efficiency of system operation and maintenance. To address the problem of redundant results of traditional pattern mining algorithms，this paper proposes a Discovering sequential patterns from Temporal log Sequences（DTS）algorithm. DTS heuristically discovers the set of patterns that can best represent the event relationships and temporal regularities in the original sequence. At the same time，DTS applies the Minimum Description Length（MDL）principle to pattern mining，and proposes an encoding scheme that considers event relationships as well as temporal relationships to solve pattern explosion. Experimental results on real log datasets show that compared with SQS，CSC，ISM and other sequential pattern mining algorithms，the proposed algorithm is capable of efficiently mining meaningful sequential patterns with low redundancy.

【Key words】data mining；log analysis；event relationships ；Minimum Description Length（MDL）principle；sequential patterns
DOI：10. 19678/j. issn. 1000-3428. 0057181

0 概述

日志数据记录了互联网系统运行时的状态以及任务的开始与结束等重要事件，其易于获取且含有丰富的信息，已经成为系统运维领域的重要数据源。系统运行产生的日志数据可被视作时序事件序列，利用序列模式挖掘技术可从时序事件序列中发现有意义的序列模式。挖掘得到的模式结果能够有效帮助工程师理解系统行为，还可进一步用于异常检测、故障诊断与原因分析等任务。

序列模式挖掘是数据挖掘领域中的重要研究课题之一，由AGRAWAL和SRIKANT[1]于1995年提出。传统的序列模式挖掘算法致力于发现序列中频繁出现的序列模式，但是该类算法生成的模式结果集数目众多且信息冗余，不利于人工查看分析，因此从序列中挖掘出高质量低冗余的序列模式具有实际意义。为了减少模式结果的数目并降低模式结果冗余度，常用的方法是筛选出原始结果集中的部分模

基金项目：国家自然科学基金(61672163)。
作者简介：杜诗晴(1995—)，女，硕士研究生，主研方向为数据挖掘；王 鹏(通信作者)，副教授、博士生导师；汪 卫，教授、博士生导师。
收稿日期：2020-01-13 **修回日期：**2020-02-17 **E-mail：**17212010006@fudan.edu.cn

式作为最终结果返回,使得最终结果中的模式在充分代表原序列中典型行为模式的同时有效降低结果集数目。文本压缩中广泛使用的最小描述长度(Minimum Description Length,MDL)准则[2]是一种有效的筛选准则,该准则使得模式结果集中的复杂度和其代表序列的能力之间达到良好平衡。

目前,文献[3-5]将MDL准则应用于序列模式挖掘,该类算法的基本思路是通过设计编码方案,采用一组模式集合作为字典对原始序列进行编码。由于编码后所需的描述长度更少,因此该过程可视作对原始序列信息的压缩。通过寻找具有最佳压缩能力的集合作为结果返回,即可得到信息紧凑的模式结果。利用MDL准则筛选出的模式集合能够最大程度地压缩原始序列中的信息,因此该集合可以有效代表整个序列。然而,目前尚未有研究充分考虑时序序列中存在的时间规律性。日志数据记录的是系统运行过程中重复产生的行为,系统日志中事件与事件之间通常存在显著的时间规律性。相对于普通的序列模式而言,对相邻事件之间的时间关系进行总结的模式能够有效指示系统的运行状态,这说明发现具有时间规律的模式具有一定的实际意义和应用价值。

本文提出一种从时序日志序列中挖掘序列模式(Discovering sequential patterns from Temporal log Sequences,DTS)的算法,通过挖掘序列中能充分代表事件关系和时序关系的模式,有效增强模式结果集的表达能力。基于MDL准则设计一种编码方案,该方案以序列模式结果为字典集,在信息无损的前提下对原始时序序列进行编码,并通过计算编码前后的描述长度来衡量不同序列模式的有效性,从而发现高质量的序列模式结果集。

1 相关工作

国内外有许多关于日志分析领域的研究工作,如文献[6]提出DeepLog算法,该算法将原始日志文本转换为事件序列,并利用日志序列对系统进行异常检测和原因分析。文献[7]提出CloudSeer算法,以日志序列为输入数据,并对系统运行时的工作流进行建模。文献[8]提出一种基于归一化特征判别的日志模板挖掘算法,将原始日志文本解析成不同的事件类型。文献[9]利用层次聚类算法挖掘频繁日志事件序列,并以此为基础对不同类型的故障进行预测。现有工作主要集中在通过日志分析进行自动化系统维护和故障分析[10]。本文旨在从原始日志序列中发现有意义的序列模式,且发现的模式可以用作在线监控和异常检测等任务。

序列模式挖掘问题作为数据挖掘领域广泛研究的重要课题之一,其具有很高的研究热度。传统的序列模式挖掘算法例如PrefixSpan算法[11]的生成模式结果集数目众多且信息冗余。为解决该问题,用于衡量模式结果质量的不同模式语义被提出,例如闭频繁模式集[12]、最大频繁模式集[13]、高效用模式集[14]以及top-k模式集[15]等。然而,上述算法挖掘得到的模式结果集依旧存在高度冗余。

为进一步降低结果集冗余度,Krimp算法[16]将MDL准则应用于模式挖掘,并以此为评价标准筛选出最能代表原始数据的模式结果集合。GoKrimp算法[4]进一步将MDL准则应用拓展到序列模式挖掘领域,后续的SQS算法[3]和CSC算法[5]均沿用了该思路。现有基于MDL准则的序列模式挖掘算法可以分为两类。GoKrimp算法[4]和SQS算法[3]仅考虑了事件与事件之间的关系,通过对长时间间隔惩罚的方式,选择事件间隔尽可能小的模式,得到的模式依旧存在一定的冗余且不能正确反映序列信息。CSC算法[5]将一条序列模式中相邻事件之间的事件间隔限制为固定长度,并且不允许同一条模式中出现两个同一类型的事件,极大地限制了模式的表达能力,且对于同一条序列需要更多的序列模式来表示。上述算法均未充分考虑处理时序序列中存在的时间规律性,相比之下,DTS利用模式中的事件关系以及相邻事件之间的时间规律性来获得更好的模式,并且能够发现冗余度较低的模式集合。

2 问题描述

2.1 相关定义

定义1(时序日志序列) 一条时序日志序列是由n条具有时间戳的日志构成的有序序列$S_{\log}=\{(\log_1,t_1),(\log_2,t_2),\cdots,(\log_n,t_n)\}$,其中$t_i\leqslant t_{i+1}(1\leqslant i\leqslant n)$。序列的时间跨度为$\Delta(S)=t_n-t_1$。时间戳的粒度可以按需设置为不同级别。

由于原始日志信息的形式为无结构文本,日志分析工作的一个常用预处理步骤是将无结构的日志文本解析为结构化的表示形式[17]。由同一条日志输出语句生成的日志信息具有高度相似的结构,反之亦然。基于此观察,一条原始日志文本可以被解析为事件类型和参数两个部分信息。其中,事件类型对应日志输出语句的文本常量部分,参数对应日志输出语句中变量部分的取值。由于Drain算法[18]与现有同类算法相比具有优异的性能,因此本文使用该算法解析原始日志序列。一条日志信息$\log_i$可以被解析为$[e_i,\text{para}_1,\text{para}_2,\cdots,\text{para}_p]$的结构化形式,其中,$e_i$表示事件类型,$\text{para}_j$表示$\log_i$中的第$j$个参数值。由于本文的后续工作不涉及参数部分的信息,后续中仅用相应的事件类型e_i表示一条日志。

定义2(事件类型集合) 事件类型集合$\Sigma=\{E_1,E_2,\cdots,E_l\}$为日志序列$S_{\log}$中出现的所有事件类型的集合。

定义 3（解析后时序日志序列） 通过对原始时序日志序列中的日志文本进行解析，得到解析后的时序日志序列$S=\{(e_1,t_1),(e_2,t_2),\cdots,(e_n,t_n)\}$。

本文后续的模式挖掘工作以解析后的时序日志序列为输入数据。序列模式的相关定义如下：

定义 4（序列模式） 一条序列模式P_i是由事件类型构成的偏序序列$<E_1,E_2,\cdots,E_m>,E_j\in\Sigma(1\leqslant j\leqslant m)$。其中，$E_i$出现在$E_j$之前$(1\leqslant i<j\leqslant m)$，模式长度为$m$。

定义 5（实例） 模式P_i的一个实例O_{P_i}对应于其在序列中的一次出现，形式化表示为$O_{P_i}=<(e_1,t_1),(e_2,t_2),\cdots,(e_m,t_m)>$，其中，$(e_j,t_j)\in O_{P_i}(1\leqslant j\leqslant m)$，$(e_j,t_j)\in S$。$e_i$代表事件类型$E_i$的一个实例，$O_{P_i}$为$P_i$的所有实例构成的集合。

定义 6（支持度） 模式P_i的支持度supp_i为所有非重叠的实例数目。如果2个实例之间不共享同一事件实例，则它们被认为是非重叠的。本文要求不同模式的实例之间也是非重叠的。

定义 7（时间间隔分布） 给定模式P_i的实例集合O_{P_i}，相邻两个事件e_k和e_{k+1}之间的时间间隔构成了一个时间间隔分布。本文用直方图h_i^k描述该分布，其中桶$h_i^k[b]$对取值为b的时间间隔进行计数。为了保证时间信息无损，桶宽设定为当前序列的时间粒度。

定义 8（单事件序列模式） 单事件序列模式P_i为只包含一个事件类型的特殊序列模式，$P_i=<E_i>$，且长度为1。单事件序列模式不包含直方图信息。

2.2 问题定义

给定一条解析后时序日志序列S，本文目标是发现一组序列模式，这些模式能以最佳的形式无损压缩原序列中的事件关系以及时间规律性。本文使用MDL准则作为衡量数据压缩质量和结果复杂性之间平衡的指标。

MDL准则的基本思想为：给定模型集合$\mathcal{M}$，对于序列S而言，模型$M\in\mathcal{M}$能使描述长度$L(S,M)=L(M)+L(S|M)$最小，其中$L(M)$为M所需的描述长度，$L(S|M)$为编码后的S所需的描述长度，描述长度按位级别计算。

对于本文的目标问题而言，模型M为序列模式P_i构成的集合$\mathcal{P}$。确定好如何利用$\mathcal{P}$中的模式对序列S进行编码的方案后，可以通过计算编码前后描述长度的变化来衡量模式质量。本文的问题可以形式化定义为：给定时序日志序列S，寻找模式集合$\mathcal{P}$，使得描述长度$L(S,\mathcal{P})=L(\mathcal{P})+L(S|\mathcal{P})$最小。

3 编码方案

本节给出具体的编码方案和描述长度的计算方法。由于对序列编码是本文评价模式质量的手段而非目标，因此DTS更关注于编码后的描述长度而非具体的编码结果。

3.1 模式集合编码

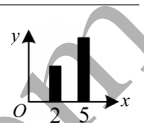
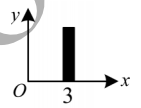
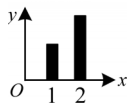
根据MDL准则，需要计算模型编码后的描述长度$L(\mathcal{P})$，计算方法如下所示：

$$L(\mathcal{P})=\sum_{P_i\in\mathcal{P}}L(P_i) \tag{1}$$

下文介绍单条模式的描述长度$L(P_i)$的计算过程。如表1所示，模式集合$\mathcal{P}$中存储了用于编码的序列模式的内容，包括模式、支持度、模式编码、描述相邻事件之间时间间隔分布的直方图和对应的编码。

表1 模式集合的描述

Table 1 Description of the set of patterns

模式集合	支持度	Code(P)	Code($h_i^k[b]$)	直方图
P_1=<ABC>	20	$C(P_1)$	$C(h_1^1[2])$，$C(h_1^1[5])$	
			$C(h_1^2[3])$	
P_2=<DD>	60	$C(P_2)$	$C(h_2^1[1])$ $C(h_1^1[2])$	

以表1中的模式$P_1=<ABC>$为例，该模式的支持度为20，模式编码为$C(P_1)$。P_1使用了两个直方图分布描述事件A和事件B以及事件B和事件C之间的时间间隔分布。其中，事件A和事件B的时间间隔有两种取值，分别为2和5，两个非空桶的编码分别为$C(h_1^1[2])$和$C(h_1^1[5])$。

对于每个模式P_i，需要对其内容序列和描述相邻事件之间时间间隔分布的直方图分别编码。使用编码（Code）替换模式在序列S中的实例，即可得到编码后的序列。尽管S中的大部分事件会被一条模式所覆盖，最终编码后的序列中可能依旧保留一些孤立事件，这些事件不属于任何模式的实例。本文用长度为1的单事件序列模式$<E_i>$对这些孤立事件进行编码，E_i为孤立事件的事件类型。

对单条模式P_i而言，描述长度$L(P_i)$的计算如下：描述一个事件类型需要$\text{lb}|\Sigma|$位比特，因此长度为m的模式需要$m\times\text{lb}|\Sigma|$位比特描述序列内容。本文使用全局唯一码来描述每个模式。考虑到出现更频繁的模式应当消耗更短的描述长度，因此二进制形式的模式编码的长度由模式频率f_i决定，其中$f_i=\dfrac{\text{supp}_i}{F}$，$F$为所有模式支持度之和，$F=\sum_{P_i\in\mathcal{P}}\text{supp}_i$。本文使用最佳前缀码，因此编码长度$|C(P_i)|=-\text{lb}f_i$。

根据前文所述，直方图的桶宽设置为序列 S 的时间粒度。对于 P_i 中的时间间隔分布而言，时间间隔越规律，直方图 h_i^k 中的非空桶数目越少。将 h_i^k 中的非空桶数目记作 $\mathrm{cnt}_{\mathrm{bins}}$，对于比值 $r_{h_k}=\dfrac{\mathrm{cnt}_{\mathrm{bins}}}{\mathrm{supp}_i}$ 而言，r_{h_k} 越小，相邻事件 e_k 和 e_{k+1} 之间的时间间隔越有规律性。基于此属性可以设计一种编码方案来压缩 P_i 实例集合中的时间规律。

考虑相邻事件 e_k 和 e_{k+1} 之间的时间间隔分布，非空桶 $h_i^k[b_j]$ 记录了取值为 b_j 的时间间隔数目 cnt_{b_j}。b_j 为时间戳之差 $t_{k+1}-t_k$，需要 $\lg\Delta(S)$ 位比特描述，cnt_{b_j} 需要 $\mathrm{lb}\,\mathrm{supp}_i$ 位比特描述。对于直方图中的非空桶 $h_i^k[b_j]$，同样需要一个编码来唯一识别。取值为 b_j 的时间间隔数目越多，编码 $C(h_i^k[b_j])$ 所需描述长度越短，计算方式与模式编码相同，$|C(h_i^k[b_j])|=-\mathrm{lb}\dfrac{\mathrm{cnt}_{b_j}}{\mathrm{supp}_i}$。因此，描述一个直方图 h_i^k 所需比特长度 $L(h_k^i)$ 计算方法如下：

$$L(h_k^i)=\sum_{j=0}^{\mathrm{cnt}_{\mathrm{bins}}}\left(\mathrm{lb}\,\Delta(S)+\mathrm{lb}\,\mathrm{supp}_i+|C(h_i^k[b_j])|\right) \tag{2}$$

对一条序列进行编码所需描述长度 $L(P_i)$ 计算方法如下：

$$L(P_i)=m\times\mathrm{lb}|\Sigma|+|C(P_i)|+\sum_{k=1}^{m-1}L(h_i^k) \tag{3}$$

假设表1中模式集合描述的序列包含事件类型数目为 $|\Sigma|=100$ 的序列，则表1中模式 $P_1=<ABC>$ 的描述长度如下所示：

$$\begin{aligned}L(P_1)=3\mathrm{lb}\,100+&\left|C(P_1)\right|+\left|C(h_1^1[2]\right|+\\&\mathrm{lb}\,\Delta S+\mathrm{lb}\,20+\left|C(h_1^1[5]\right|+\mathrm{lb}\,\Delta S+\mathrm{lb}\,20+\\&\left|C(h_1^2[3]\right|+\mathrm{lb}\,\Delta S+\mathrm{lb}\,20\end{aligned} \tag{4}$$

3.2　序列编码

给定模式集合 $\mathcal{P}$，使用模式编码和直方图编码替换序列 S 中模式对应的实例，得到编码后序列结果。由于本文针对的是时序序列数据，模式实例中覆盖的事件和时间戳均应被编码。本文的编码方案如图1所示，模式 P_i 的实例 O_{P_i} 由编码 $C(P_i)$ 和首个事件实例的时间戳替换，相邻事件之间的时间间隔由对应直方图的桶编码 $C(h_i^k[b_j])$ 替换。实例 O_{P_i} 编码后的结果为 $(C(P_i),t_i),C(h_i^1[b_{j_1}]),\cdots,C(h_i^{m-1}[b_{j_{m-1}}])$。对于不属于任何模式实例的孤立事件，用对应事件类型构成的单事件序列模式对其编码，编码后结果为 $(C(E_i),t_i)$。使用表1中的模式集合对原序列编码后的结果如图1所示。例如 $(a,0)(b,2)(c,5)$ 为模式 P_1 的一个实例，编码后的结果为 $(C(P_1),0)C(h_1^1[2])C(h_1^2[3])$。而事件 $(e,4)$ 没有被任何模式覆盖，在编码后序列中表示为 $(C(E),4)$。

原序列：{(a,0)(d,1)(a,1)(d,2)(b,2)(b,3)(e,4)(d,5)(c,5)(c,6)(d,7)⋯}
编码后：{$(C(P_1),0)C(h_1^1[2])C(h_1^2[3]),(C(P_2),1)C(h_2^1[1]),$
$(C(P_1),1)C(h_1^1[2])C(h_1^2[3]),(C(E),4),(C(P_2),5)C(h_2^1[2]),\cdots$}
时间轴 ⟶

图1　时序事件序列编码结果

Fig.1　Encoding result of a temporal event sequence

使用模式集合 $\mathcal{P}$ 对序列 S 编码所需的描述长度 $L(S|P)$ 计算方法如下：

$$L(S|P)=\sum_{P_i\in\mathcal{P}}\Big[|C(P_i)|+\mathrm{lb}\,\Delta(S)+\sum_{k=1}^{m-1}|C(h_i^k[b])|\Big]\times\mathrm{supp}_i \tag{5}$$

4　序列模式挖掘算法

文献[4]证明了基于MDL准则从序列 S 中挖掘最优模式集合 $\mathcal{P}$ 为NP问题。因此，本文设计一种启发式算法DTS迭代地从序列中发现模式。

假定序列 S 最初仅用单事件模式编码，模式集合 $\mathcal{P}$ 初始状态下为空，DTS迭代地更新 $\mathcal{P}$。本文涉及到的更新操作有两种：添加操作和改进操作，且分别定义如下：

定义9（添加操作）　将从序列中新发现的候选模式 P 添加到集合 $\mathcal{P}$ 中，更新后的模式集合为 $\mathcal{P}\cup P$。

定义10（改进操作）　给定集合 $\mathcal{P}$ 中的某条序列模式 $P=<E_1,E_2,\cdots,E_m>$，通过在位置 j 添加事件类型为 E' 的事件实例，对集合 O_P 中部分实例改进，生成长度为 $m+1$ 的新模式 P_r，事件 E' 称为改进事件。新模式 $P_r=P\oplus E'$ 有如下3种情况：

1）$j=0$，$P_r=<E',E_1,E_2,\cdots,E_m>$。

2）$j\in[1,2,\cdots,m-1]$，$P_r=<E_1,E_2,\cdots,E_i,E',E_{i+1},E_{i+2},\cdots,E_m>$。

3）$j=m$，$P_r=<E_1,E_2,\cdots,E_m,E'>$。

若 O_P 中所有实例均被改进，称作完全改进 $P\xrightarrow{R_f}P_r$，更新后的集合为 $\{\mathcal{P}\setminus P\}\cup\{P_r\}$，否则，称作部分改进 $P\xrightarrow{R_p}\{P_m,P_r\}$，$P_m$ 为原模式，更新后的模式集合为 $\{\mathcal{P}\setminus P\}\cup\{P_m,P_r\}$。

一种常见的情况是候选模式 P 覆盖的某些事件，亦是集合 $\mathcal{P}$ 中某条模式 P' 的改进事件，即出现了事件冲突。每轮迭代中决策更新操作时，若不存在事件冲突，可以直接采用添加操作将当前最佳候选模式更新到集合 $\mathcal{P}$ 中。在出现事件冲突的情况下，考虑到本文定义的编码方案要求不同模式之间是非重叠的，即同一条事件实例不能被多个模式所覆盖，DTS需要选择更优的更新操作，即能够使得描述长度 $L(\mathcal{P})+L(S|\mathcal{P})$ 减少更多的更新操作。每次对集合 $\mathcal{P}$ 更新后，DTS对 S 中为新模式所覆盖的事件实例进行编码。DTS对集合 $\mathcal{P}$ 迭代更新，直到没有更新操作能够使得描述长度进一步减少为止。DTS维护了两个数据结构来分别支持添加和改进这两个更新操

作,即候选模式集和候选改进表。下面详细介绍这两个数据结构和算法流程。

4.1 候选模式集

对于添加操作,被添加到$\mathcal{P}$中的模式应当能够最大限度地优化描述长度。一个直观的思路是通过穷举当前序列中所有模式来选择最佳的候选模式,然而在指数级空间中搜索的复杂度很高。因此,在算法迭代过程中,候选模式集$\mathbb{C}$P中仅存储当前有潜力被添加到集合$\mathcal{P}$的候选模式。候选模式P对描述长度优化能力的评价函数如式(6)所示,且gain(P)值越大,模式P的优化能力越强。

$$\text{gain}(P)=L(S,\mathcal{P})-L(S,\mathcal{P}\cup P) \tag{6}$$

候选模式集$\mathbb{C}$P的初始化计算如算法1所示,初始状态下$\mathbb{C}$P为不包含任何模式的空集。对于序列S中的每个事件类型$E\in\Sigma$,创建一个以E开头的模式P,对该序列模式以深度优先的方式调用程序BestGrowth,计算以当前事件类型为开头能生成的最佳序列模式。若最佳序列模式能进一步优化描述长度,将该模式加入候选模式集,即候选模式集$\mathbb{C}$P贪心地存储了以每种事件类型为开头,并将能生成具有最佳优化能力的模式作为候选模式。

算法1 候选模式集初始化Initialize $\mathbb{C}$P

输入 序列S,事件集合Σ

输出 候选模式集$\mathbb{C}$P

1.$\mathbb{C}$P ← ∅
2.for E ∈ Σ do
3.创建模式P ← E,新模式P′ ← P
4.while(P ←BestGrowth(P, S, Σ)≠null) do
5.P′ ← P
6.end while
7.若gain(P′) > 0,将P′存入$\mathbb{C}$P//当前模式P已经是最佳模式
8.end for
9.返回$\mathbb{C}$P

子程序BestGrowth尝试将模式P扩展到长度为$|P|+1$且具有更好优化能力的新模式中,具体过程为:尝试使用每个事件类型对当前模式扩展,得到新模式$P'\leftarrow P\circ E'$。依次计算序列S中P'的最左实例并存入$O_{P'}$,直到S中不存在P'的最左实例为止。其中,最左实例为序列中最早出现的实例,例如图1中$(a,0)(b,2)(c,5)$即为模式$<ABC>$的一个最左实例。根据编码方案计算gain(P'),选取其中描述长度降低最多的模式作为最佳扩展结果返回。若本次调用BestGrowth没有找到新扩展模式,程序返回null。

4.2 候选改进表

对于改进操作,DTS维护了一个候选改进表$\mathbb{C}$R并用于存储模式集合$\mathcal{P}$中所有模式P所有位置上的最佳改进。一个模式P在位置j处的改进$R(P)_j^E$可以格式化地存储为结构[改进位置j,改进事件E',原模式P_m,改进模式P_r]。例如[1,F,DD,DFD]存储的是模式DD在位置1处部分改进得到的模式DFD,原模式DD依旧保留部分实例。若DD被完全改进,结果为[1,F,null,DFD]。候选改进表中的改进结构以改进事件类型为索引,即同一改进事件类型关联的改进结构聚集为一组。

每次由添加操作或改进操作新生成的模式P存入模式集合$\mathcal{P}$后,需要对P计算其每个位置上可能的改进。对模式P计算改进操作的流程如算法2所示,对模式P的每个位置j调用程序Refine计算该位置上能生成的最佳改进,即最能减少编码后描述长度的改进。若位置j上存在最佳改进,将最佳改进$R^*(P)_j^E$存入候选改进表,改进操作对描述长度优化能力的评价函数如式(7)所示,且gain($R(P)_j^E$)值越大,则改进$R(P)_j^E$的优化能力越强。

$$\text{gain}(R(P)_j^E)=L(S,P)-L(S,\{PP\}\cup\{P_m,P_r\}) \tag{7}$$

算法2 对模式P计算改进ComputeRefining

输入 模式P,序列S,事件集合Σ,候选改进表$\mathbb{C}$R

输出 候选改进表

1.for j = 0 → |P| do
2.$R^*(P)_j^E$ ← null
3.for E ∈ Σ do
4.$R(P)_j^E$ ←Refine(P, S, j, E)
5.若gain($R(P)_j^E$) > gain($R^*(P)_j^E$),$R^*(P)_j^E$ ← $R(P)_j^E$
6.end for
7.若gain($R^*(P)_j^E$) > 0,将$R^*(P)_j^E$存入$\mathbb{C}$R.get(E)
8.end for
9.返回更新后的$\mathbb{C}$P

给定待改进模式P,改进位置j,改进事件E,子程序Refine计算是否能使用事件E在位置j处对模式P改进,具体过程为:首先根据改进操作的定义创建新模式P_r,对实例集合O_P中的每个实例,计算该实例能否被事件E的实例按照定义进行改进,此处依旧遵循最左原则。若存在改进,将改进后的实例结果存入模式P_r的实例集合。若O_{P_r}不为空,说明存在有效改进,O_P中未被改进的实例$O_P\setminus O_{\text{used}}$对应原模式剩余的实例,用$P_m$表示原模式,返回改进结果为[$j$,$E$,$P_m$,$P_r$]。图2展示了调用程序Refine($P$,$S$,2,$E$)对模式计算改进的过程,使用事件对$P$的前两个实例在位置2处改进得到模式$P_r$的实例集合,而第三个实例找不到满足时序关系的改进事件实例e,因此保留为原模式P_m的实例。

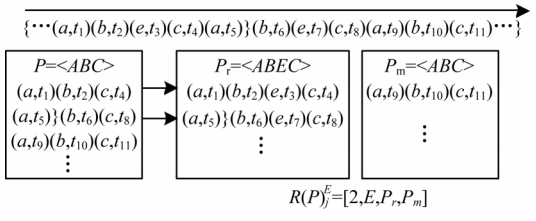


图2 Refine程序计算示例

Fig.2 Example of the computation of Refine program

4.3 序列模式挖掘

DTS基于启发式的思路迭代地从序列中挖掘模式,算法描述如算法3所示。首先对算法涉及的数据结构初始化(第1行~第2行),模式集合$\mathcal{P}$初始状态下为空,S仅用单事件序列模式编码。调用程序Initialize$\mathbb{C}$P初始化候选模式集$\mathbb{C}$P,由于集合$\mathcal{P}$中尚未存入新模式,则候选改进表$\mathbb{C}$R初始状态下为空。

算法3 序列模式挖掘算法DTS

输入 序列S

输出 模式集合P

1.Σ ← 序列S中所有事件类型,P ← ∅
2.$\mathbb{C}$P ←Initialize$\mathbb{C}$P(S, Σ),$\mathbb{C}$R ←new Map()
3.P ←$\mathbb{C}$P中最佳候选模式
4.将P存入P
5.ComputeRefining(P, S, Σ, $\mathbb{C}$R)
6.while true do
7.P^* ← $\mathbb{C}$P中最佳候选模式
8.if P^* =null,跳出循环
9.$R^*(P')_j^E$ ← $\mathbb{C}$R中存在事件冲突的最佳改进
10.if cmp(P^*, $R^*(P')_j^E$) > 0,
11.P ← P ∪ { P^*}
12.else P ← { P \ P } ∪ { P_m, P_t}
13.end if
14.更新$\mathbb{C}$P,$\mathbb{C}$R,对S中被覆盖的事件编码
15.end while
16.使用$\mathbb{C}$R中的改进更新P,直到$\mathbb{C}$R = ∅
17.返回P

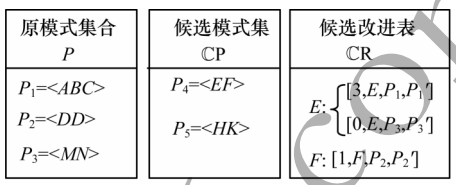
如前文所述,DTS对$\mathcal{P}$的迭代更新是由候选模式主导的。在出现事件冲突的情况下,DTS选择能够使得描述长度减少更多的操作,比较模式P和改进$R(P')_j^E$对描述长度优化能力的函数为:

$$\mathrm{cmp}(P, R(P')_j^E) = \frac{\mathrm{gain}(P)}{|P| \times |O_P|} - \frac{\mathrm{gain}(R(P')_j^E)}{|O_{P'_j}|} \tag{8}$$

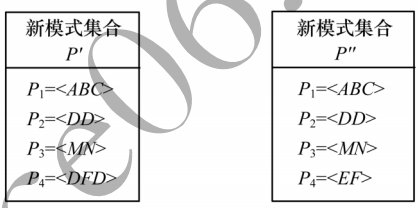
由于长模式或支持度较高的模式往往对描述长度有更多的优化,而改进操作的优化仅由单个事件贡献,为了保证比较的公平性,此处比较的是单个事件的优化能力。函数返回值为正则表明模式P有更好的优化能力。

在开始迭代过程之前,从候选模式集$\mathbb{C}$P中获取当前最佳候选模式P,由于候选改进表$\mathbb{C}$R为空,不存在事件冲突,直接将P存入集合$\mathcal{P}$。由于加入了新模式,调用程序ComputeRefining计算其相应的改进并存入$\mathbb{C}$R。在接下来的算法迭代过程中(第6行~15行),DTS首先从$\mathbb{C}$P中获取最佳候选模式P^*,若没有新的候选模式,迭代流程终止(第8行)。检测$\mathbb{C}$R中是否存在与P^*事件冲突的改进,即模式P^*中的事件是其他模式的改进事件,并获取优化能力最佳的改进$R^*(P')_j^E$。对集合$\mathcal{P}$的更新采用添加操作还是改进操作取决于cmp函数的比较结果。

以图3为例说明算法核心思想,图3(a)所示为当前待更新的模式集合$\mathcal{P}$以及当前迭代轮次的候选模式集$\mathbb{C}$P和候选改进表$\mathbb{C}$R。若当前$\mathbb{C}$P中的最佳候选模式P^*为P_1=<EF>,相应的$\mathbb{C}$R中两个改进对P_1存在事件冲突,其中$R^*(DD)_1^F$=[1, F, DD, DFD]具有最佳的优化能力。调用函数cmp对二者的优化能力进行比较,若$R^*(DD)_1^F$优化能力更强,模式集合$\mathcal{P}$被改进操作更新为P',更新后的模式集合如图3(b)所示。否则,将P_1存入$\mathcal{P}$,模式集合$\mathcal{P}$被添加操作更新为P'',更新后的模式集合如图3(c)所示。每轮更新$\mathcal{P}$后,需要及时对数据结构候选模式集$\mathbb{C}$P以及候选改进表$\mathbb{C}$R中的内容更新,删除已经失效的模式和改进结构,并重新计算新的候选模式和改进结构。DTS对模式集合$\mathcal{P}$的迭代更新在无法产生新候选模式后终止,随后仅用改进操作更新$\mathcal{P}$直到$\mathbb{C}$R = ∅。



(a)待更新模式集合P及当前候选模式集和候选改进表



(b)改进操作更新为模式集合P' (c)添加操作更新为模式集合P''

图3 模式集合的更新示例

Fig.3 Update example of pattern collection

5 实验与结果分析

5.1 实验环境

本文所有实验在单机环境下运行,处理器为Intel® Core™ i7-3770 CPU @ 3.40 GHz,内存为24 GB,操作系统为64位Windows 10。

5.2 实验数据

本文采用真实Hadoop分布式文件系统(Hadoop Distributed File System,HDFS)环境运行生成的日志序列作为输入数据,共生成8条不同长度的日志序列,其中4条日志序列是HDFS系统的NameNode生成的,包含200种事件类型,4条日志序列是HDFS系统的DataNode生成的,包含106种事件类型。这两种日志数据有不同的特点,其中NameNode日志包含更丰富的事件类型和更复杂的系统行为,因此模式更为复杂,而DataNode日志包含相对更少的事件类型,其中的模式也相对更为简单规律。

5.3 算法对比分析

实验对DTS与SQS[3]、GoKrimp[4]、CSC[5]、ISM[19]算法进行对比分析。其中,文献[3-5]都是基于MDL准则挖掘模式,ISM算法是一个基于概率的机器学习方法。对于CSC和DTS,输入数据即为整条序列,对于其他3种算法,将序列切分为长度为10的子序列作为输入数据。DTS由Java实现,其余算法的源代码由作者提供。

5.3.1 运行效率的评估

本节使用不同长度的日志数据对算法的效率和可扩展性进行评估,5种算法所需运行时间如图4所示。值得指出的是,尽管5种算法的结果呈现在一张图内,CSC、GoKrimp和DTS算法的时间单位为秒,SQS和ISM的时间单位为分钟,为了节省空间,本文将不同数量级的运行时间展示在同一张图片内。从图4可以看出,DTS略慢于CSC和GoKrimp,这是因为CSC和GoKrimp在计算过程中通过设定间隔阈值的方式减枝,极大地减少了搜索空间,但后续实验证明这也限制了模式的表达能力。尽管DTS相对来说耗时较长,但也只需400 s即可处理长度为600 000的复杂日志序列,这是一个可以接受的耗时时间,说明DTS可以高效发现模式且有很好的可扩展性。相比之下,SQS和ISM算法效率不高,处理日志所需耗时从几十分钟到几小时不等,难以满足日常应用需求。

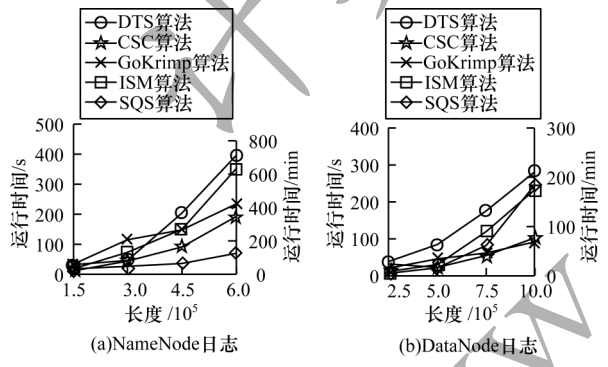


图4 不同数据集下的运行时间对比

Fig.4 Comparison of processing time under different datasets

5.3.2 模式表达能力的评估

本节使用不同长度的日志序列对模式结果的表达能力进行评估,使用到的评估指标为压缩率(Compression Ratio,CR)。基于MDL准则筛选模式挖掘出的模式结果可以用于对序列编码,与原序列所需描述长度相比,编码后的序列长度大幅减少。根据MDL准则描述长度越小,用于编码的模式集合表达能力越强。压缩率的计算为只使用单事件序列模式对序列编码所需描述长度和使用模式集合对序列编码所需描述长度的比值。

实验所涉及的基于MDL准则算法的压缩率如图5所示。从图5可以看出:对于NameNode日志,DTS在所有长度的日志序列上均取得了最高的压缩率。在对比算法中,SQS的压缩率最好,CSC和GoKrimp相对更差,如前所述,这两种算法的减枝和贪心策略使得它们不能很好地处理复杂的日志序列;对于DataNode日志,所有算法的压缩率都较低,这是因为该日志中的模式更为简短。DTS在部分序列上仍然拥有最高的压缩率,仅在两条序列上略逊于CSC,这与前文的分析一致,CSC更适合处理模式相对简短的序列数据,但是后续实验分析可以看出CSC结果的冗余度很高。DTS在大部分数据上取得了最好的压缩率,这表明DTS挖掘出的模式能更好地代表原序列。

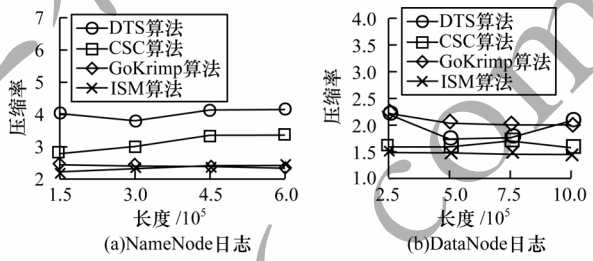


图5 不同数据集下的压缩率对比

Fig.5 Comparison of compression ratio under different datasets

5.3.3 模式冗余度的评估

DTS挖掘出的模式集合能更好地对序列进行压缩,这并不意味着DTS的模式集合有很高的冗余度,本节对该结论加以实验验证。本文采用的评估指标如下:

1)平均序列间编辑距离(Average inter-sequence Edit Distance,AED)[19]:用于评估模式与模式之间的文本相似度,计算方式为对模式集合$\mathbb{CP}$中所有模式P与$\mathbb{CP}$中其他模式P'之间编辑距离的最小值求平均值,且AED越高,模式集合冗余度越低。

2)平均序列间杰卡德距离(Average inter-sequence Jaccard Distance,AJD):用于评估模式与模式之间的杰卡德相似度,计算方式与AED类似。

3)事件覆盖率(Event Coverge,EC)[20]:用于评估模式集合对事件类型的覆盖率,且EC越高,说明模式集合涵盖的事件类型越丰富。

表2展示了5种算法在不同数据集下的3种评估指标结果。其中,最优结果加粗表示。由于空间限制,此处仅列出两个最长序列的结果。从表2可以看出:DTS具有最大的AED和AJD;与其他算法相比,DTS发现的模式具有最低冗余度;此外,DTS在事件覆盖率方面也优于其他算法,这表明DTS挖掘出的模式结果在高质量的同时且低冗余,这对于人工分析和处理是非常便利的。

表2　5种算法在不同数据集下的模式冗余度对比

Table 2　Comparison of pattern redundancy of five algorithms under different datasets

算法	NameNode(600 000)			DataNode(1 000 000)		
	AED	AJD	EC	AED	AJD	EC
DTS	**6.173**	**0.531**	**0.820**	**5.030**	**0.747**	0.877
ISM	2.085	0.271	0.800	2.117	0.311	**0.887**
CSC	2.522	0.137	0.740	2.000	0.092	0.764
SQS	1.620	0.157	0.815	1.302	0.146	0.840
GoKrimp	1.667	0.314	0.160	1.444	0.306	0.132

6　结束语

为从时序日志序列中发现高质量的序列模式，本文提出一种基于MDL的序列模式挖掘算法DTS。DTS能够以黑盒的方式处理日志数据，在不需要任何先验系统知识的情况下，从日志序列中发现有意义的行为序列用于日志分析。该算法结合系统行为在执行时间上存在一定规律性的特点，设计一种考虑事件关系以及时间规律性的编码方案，以编码前后描述长度的变化为衡量标准来评估序列模式质量，并从数据中迭代挖掘出可有效代表序列的模式。实验结果表明，DTS可以有效发现高质量且低冗余的模式结果集。下一步将对算法搜索策略进行改进，以有效减少计算开销，提高算法效率。

参考文献

[1] AGRAWAL R, SRIKANT R. Mining sequentialp patterns [C]//Proceedings of the 11th International Conference on Data Engineering. Washington D. C. , USA: IEEE Press, 1995: 3-14.

[2] NWALD P D. The minimum description length principle (adaptive computation and machine learning) [M]. Cambridge, USA: MIT Press, 2007.

[3] TATTI N, VREEKEN J. The long and the short of it: summarising event sequences with serial episodes [C]// Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM Press, 2012: 462-470.

[4] LAM H T, MÖRCHEN F, FRADKIN D, et al. Mining compressing sequential patterns [J]. Statistical Analysis and Data Mining: the ASA Data Science Journal, 2014, 7(1): 34-52.

[5] IBRAHIM A, SASTRY S, SASTRY P S. Discovering compressing serial episodes from event sequences [J]. Knowledge and Information Systems, 2016, 47 (2): 405-432.

[6] MIN D, LI F, ZHENG G, et al. DeepLog: anomaly detection and diagnosis from system logs through deep learning [C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York, USA: ACM Press, 2017: 1285-1298.

[7] XIAO Y, JOSHI P, XU J, et al. CloudSeer: workflow monitoring of cloud infrastructures via interleaved logs [C]//Proceedings of the 21st International Conference on Architectural Support for Programming Languages and Operating Systems. New York, USA: ACM Press, 2016: 489-502.

[8] SHUANG Kai, LI Yiwen, LÜ Zhiheng, et al. Log template extraction algorithm based on normalized feature discrimination [J]. Journal of Beijing University of Posts and Telecommunications, 2020, 43(1): 68-73. (in Chinese)
双锴，李怡雯，吕志恒，等. 基于归一化特征判别的日志模板挖掘算法[J]. 北京邮电大学学报，2020，43(1)：68-73.

[9] WANG Weihua, YING Shi, JIA Xiangyang, et al. A multi-type failure prediction method based on log clustering [J]. Computer Engineering, 2018, 44(7): 67-73(in Chinese)
王卫华，应时，贾向阳，等. 一种基于日志聚类的多类型故障预测方法[J]. 计算机工程. 2018，44(7)：67-73.

[10] HE S L, ZHU J M, HE P J, et al. Experience report: system log analysis for anomaly detection [C]// Proceedings of the 27th International Symposium on Software Reliability Engineering. Washington D. C. , USA: IEEE Press, 2016: 207-218.

[11] JIAN P, HAN J, MORTAZAVI-ASL B, et al. PrefixSpan: mining sequential patterns by prefix-projected growth [C]// Proceedings of the 17th International Conference on Data Engineering. New York, USA: ACM Press, 2001: 215-224.

[12] GOMARIZ A, CAMPOS M, MARIN R, et al. ClaSP: an efficient algorithm for mining frequent closed sequences [EB/OL]. [2019-12-10]. http://citeseerx. ist. psu. edu/viewdoc/download; jsessionid=30AC1EBBDC64DA97300F9BCB9A1606B1? doi=10. 1. 1. 377. 3720&rep=rep1&type=pdf.

[13] GUAN Enzheng, CHANG Xiaoyu, WANG Zhe, et al. Mining maximal sequential patterns [C]//Proceedings of International Conference on Neural Networks and Brain. Washington D. C. , USA: IEEE Press, 2005: 525-528.

[14] HU J Y, MOJSILOVIC A. High-utility pattern mining: a method for discovery of high-utility item sets [J]. Pattern Recognition, 2007, 40(11): 3317-3324.

[15] TZVETKOV P, YAN X, HAN J. TSP: mining top-k closed sequential patterns [J]. Knowledge and Information Systems, 2005, 7(4): 438-457.

[16] VREEKEN J, VAN LEEUWEN M , SIEBES A. Krimp: mining itemsets that compress [J]. Data Mining and Knowledge Discovery, 2011, 23(1): 169-214.

[17] FU Q, LOU J G, WANG Y, et al. Execution anomaly detection in distributed systems through unstructured log analysis [C]//Proceedings of the 9th IEEE International Conference on Data Mining. Washington D. C. , USA: IEEE Press, 2009: 149-158.

[18] HE P J, ZHU J M, ZHENG Z B, et al. Drain: an online log parsing approach with fixed depth tree [C]//Proceedings of 2017 IEEE International Conference on Web Services. Washington D. C. , USA: IEEE Press, 2017: 33-40.

[19] FOWKES J, SUTTON C. A subsequence interleaving model for sequential pattern mining [C]//Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM Press, 2016: 835-844.

[20] YAN Y, CAO L, MADDEN S, et al. SWIFT: mining representative patterns from large event streams [EB/OL]. [2019-12-10]. https://arxiv. org/pdf/1602. 05012. pdf.

编辑　刘继娟