基于增强日志的过程挖掘算法

邵叱风,方贤文,盛梦君

(安徽理工大学数学与大数据学院,安徽　淮南　232001)

摘　要: 过程挖掘的目的是通过分析系统中的日志以发现、构建和优化系统业务流程。现有的过程挖掘算法大多采用从控制流角度记录和观察进程工作流的系统日志,且日志在使用前需进行大量预处理工作将其转换为算法可识别的事件日志,不仅仅增加了挖掘难度,最终挖掘所获过程模型所含属性单一,很难准确描述实际流程的运作。为减少预处理工作,增强过程挖掘算法挖掘能力,基于系统事件执行详情,通过可利用属性的提取,提出了增强日志的概念,并基于增强日志开发出一种新的过程挖掘算法。此方法利用增强日志中的附加属性,识别事件结构,通过有色 Petri 网的加入,挖掘出具有场景信息的过程模型。最后通过一个具体的挖掘实例进一步说明了该方法的可行性及结果的可扩展性。

关键词: 过程挖掘; 事件日志; 过程模型; 增强日志; 有色 Petri 网

中图分类号: TP391.9　　**文献标志码**: A　　**文章编号**: 1672-1098(2020) 04-0025-08

Process Mining Algorithm Based on Enhanced Log

SHAO Chifeng, FANG Xianwen, SHENG Mengjun

(School of Mathematics and Big Data, Anhui University of Science and Technology, Huainan Anhui 232001, China)

Abstract: The purpose of process mining is to discover, build and optimize system business processes by analyzing the logs in the system. Most of the existing process mining algorithms use system logs that record and observe process workflows from a control flow perspective, in which way, a lot of preprocessing work needs to be done to convert it into an event log able to be recognized by the process mining algorithms before the log is used, not only increasing the difficulty of mining, but also making it difficult to accurately represent the operation of the actual process because of the single attribute of the process model. In order to reduce the preprocessing work and enhance the mining ability of process mining algorithm, based on the system event execution details, the concept of enhanced log is proposed by extracting the available attributes, and a new process mining algorithm developed based on the enhanced log. This method uses additional properties in the enhanced log to identify event structures, and with the colored Petri net added, the process model with scene information is mined. Finally, a mining example is given to further illustrate the feasibility of the method and the scalability of the results.

Key words: process mining; event logs; process models; enhanced logs; colored Petri nets

---

收稿日期: 2020-03-04

基金项目: 国家自然科学基金资助项目(61402011, 61572035) ; 安徽省自然科学基金资助项目(1508085MF111, 1608085QF149) ; 安徽理工大学研究生创新基金资助项目(2019CX2068)

作者简介: 邵叱风(1995-) ,男,安徽合肥人,在读硕士,研究方向: Petri 网、过程挖掘及模型修复。

数据挖掘与过程挖掘之间既有区别又有共性。就目的而言,数据挖掘是从大量的数据中提取或挖掘出有用信息[1],而过程挖掘是从表示流程执行工作流的数据中挖掘流程模型。故数据是数据挖掘以及过程挖掘的基础。从存储方式来看二者又是不同的,前者使用的数据常用数据库、数据仓库、万维网或其他信息存储库[2]存储信息,而后者使用的数据通常存储在捕获系统工作流的日志中。

在传统的过程挖掘研究中使用的日志称为事件日志,其用于从控制流角度记录和观察进程工作流。事件表示任务的执行。不同的挖掘算法使用不同的事件日志,文献[2]提出 α 算法,使用迹矩阵定义活动间的顺序操作符,从基本事件日志中发现结构化事件模型;文献[3]提出 β 算法为每个事件标记一个类型;文献[4]提出 λ 算法将后继任务添加到事件中。以上方法提出了不同的日志处理方案,生成不同算法所需的日志形式。

过程挖掘形式化定义用于从事件日志的一组实际执行中提取结构化流程的方法。过程挖掘至少在两个方面是有用的。首先,它可以被用作一个用来查明程序真实运作的工具;其次可用于增量分析,即将实际过程与一些预定义的过程进行比较。基于①多核和并行技术的发展导致数字世界的惊人增长[5]②随着数字世界的发展和组织进程的紧密结合,使得记录和分析更多的事件[6]成为可能,过程挖掘成为工作流技术的热点之一。文献[7]提出了遗传挖掘算法,该算法提出一种有效的因果矩阵结构来提高空间搜索的效率;文献[8]使用Rule-induction过程挖掘技术,提出RIPPER算法产生活动间的规则;文献[9]使用分治策略递归地构建过程模型,直到所有的迹都被处理为止,该方法称为归纳式挖掘算法;文献[10]提出一种由 α-算法产生的启发式挖掘算法,该方法仅考虑了事件的顺序;文献[11]阐述了灵活启发式挖掘算法,它是一个灵活的控制流挖掘算法,提出的两个算法都能处理噪音和低频行为。针对过程挖掘方法有时不能获得完备事件日志的问题,文献[12]提出一种从不完备日志中发现块结构过程模型的方法,该方法利用对完备性不敏感的概率行为关系,给出了一个适用性更强的过程发现算法;文献[13]提出一种挖掘局部过程模型的方法,该方法通过生成过程树并依据5种评估标准选择局部过程模型,扩展生成新的过程树,以此迭代直到完成任务。但以上挖掘方法多以基本事件日志作为输入,且均不能挖掘出系统中的场景信息。

本文主要工作有:首先提出增强日志的相关定义,继而提出基于增强日志的挖掘算法(Process Mining-Enhanced Log,PM-EL),通过颜色Petri网的加入用以表达具有场景信息的挖掘结果;使用UML转换结果模型说明其可扩展性;对比实验展示PM-EL算法在结构识别上的能力。以上工作就基于事件内部属性对模型做出优化[14]的工作提供了可切入点。

本文第1节介绍准备知识;第2节提出增强日志的概念以及基于增强日志的过程挖掘方法;第3节通过仿真实验验证所提方法的可行性及可扩展性;最后总结全文并展望未来。

1 准备知识

在本节中,将给出一些定义以及基本概念,用以辅助解释提出的方法。限于篇幅,有关Petri网的概念及结构的定义在此不做赘述,具体内容可以参考文献[15]。

定义1[16](颜色petri网)一个简单的(带有 k 种颜色的)颜色Petri网是一个五元组 $\Sigma=(S, T; F, W, M)$ 其中$(S, T; F)$是一个网,$W: F\rightarrow\{0,1,2,\cdots\}^k$, $M: S\rightarrow\{0,1,2,\cdots\}^k$ 对 $t\in T$,如果 $s\in {}^\bullet t\rightarrow M(S)\geqslant W(s, t)$,那么变迁 t 在标识 M 有发生权($M[t>$),在标识 M 下发生变迁 t,产生一个新的标识 M'($M[t>M'$),

$$M'(s)=\begin{cases}M(s)-W(s,t), & \text{若 } s\in {}^\bullet t-t^\bullet\\ M(s)+W(t,s), & s\in t^\bullet-{}^\bullet t\\ M(s)-W(s,t)+W(t,s), & \text{若 } s\in {}^\bullet t\cap t^\bullet\\ M(s), & \text{其它}\end{cases}$$

定义2[17](UML2.0序列图)序列图是常用的且偏向于捕获对象间行为的图。它通常可以与正在开发的系统的逻辑视图中的用例相关联。高级序列图(High-level Sequence Diagrams,HLSD)是一个序列图,它引用一组基本序列图(Basic Sequence Diagram,BSD),并使用一组交互操作符组合它们。主要的操作符是:弱顺序(SEQ)、选择(ALT)、循环(LOOP)和并行(PAR)。

在Petri网的许多现有变种中,CPN被用于以

序列图的形式表示的组合和集成场景。库所表示 BSD,变迁表示操作符,如 ALT、LOOP、SEQ 和 PAR。颜色用于区分库所。图 1 显示了 HLSD 如何映射到 CPN。T3 表示条件为 C1 的操作符 LOOP。操作符 PAR 和 SEQ 也可以映射,如图 2 所示。可见对于 HLSD,可以相互转换生成一个可表示主要的 UML 序列图操作符的 CPN。

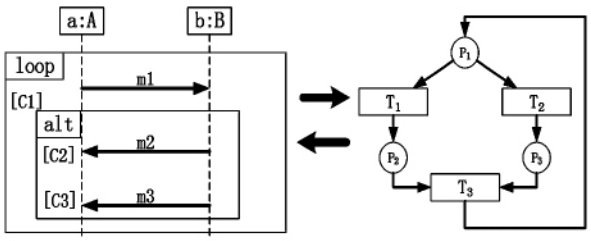

图 1　序列图将操作符 LOOP 和 ALT 映射到 CPN

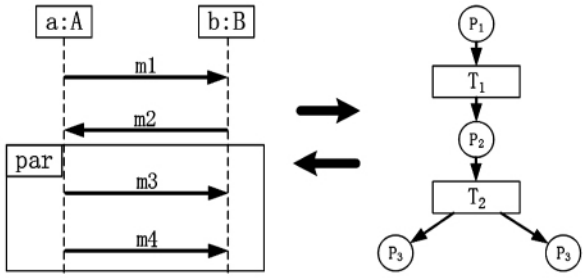

图 2　序列图将操作符 PAR 和 SEQ 映射到 CPN

2　基于增强日志的过程挖掘方法

过程挖掘起源于系统日志的使用,定义和统一系统日志是过程挖掘的关键步骤。过程挖掘 1995 年由 J.E.Cook 提出至今,技术方面取得了长足的发展,且取得了诸多较完善的研究成果,但多以行为间控制流结构挖掘过程模型; 不同类型的事件日志(其中可能包含略微不同的信息) 被不同的挖掘算法所使用,但大多数仅从控制流角度记录和观察工作流。

2.1　增强日志

在软件系统运行时会产生大量的日志,常用来作为挖掘算法输入的事件日志往往只记录了执行的事件。在此对文献[18]中的日志数据格式进行扩展,针对事件执行信息中一些可利用的属性,提出增强日志的概念。

定义 3(增强日志) 若 Traces 是任务集,且满足如下两个条件,则称 $\omega \in \text{Traces}^*$ 是一条事件轨迹,$W \subseteq \text{Traces}^*$ 为一个增强日志。

(1) Trace 是一个四元组 Trace = (线程号; 发送方,消息,接收方) ,其中: 发送方是调用对象,形如 threadNumber: package: class: object,消息是接收方对象的调用方法,形如 mathodName (param1, param2,…) ; 接收方是被调用对象,形如 package: class: object。

(2) 方法调用之间的等价性: 方法调用如 A1 = (sender1; message1; receiver1) 和 A2 = (sender2; message2; receiver2) 是等价调用,当且仅当: 两个对象隶属同一实体类(可创建相同属性值对象) ,并在同一线程中以相同的参数值创建,那么等价。信息对象 message1 与 message2 涉及相同的方法且具有相同的签名。

其中有 4 个函数操作 T_{LineNumb}(Trace) = 线程号,T_{Send}(Trace) = 发送方,T_{Msg}(Trace) = 消息,T_{Receive}(Trace) = 接收方。其中的线程号可用来分析事件执行的内部联系,其他三个属性可用来生成场景化信息。

2.2　过程挖掘算法 PM-EL

现有的过程挖掘算法从事件日志中挖掘任务之间的因果依赖关系和并行关系,发现过程模型中所有库所的前后任务,然后重建整个过程模型。文章在此提出了一种新的算法 PM-EL。在这一部分,我们首先介绍了算法的定义和数据结构。然后详细介绍了该算法的伪代码及运行过程。

在增强日志中,因果依赖、并发及选择关系定义如下。

定义 4(因果依赖) : 设 Traces = $\{t_1, t_2, \cdots, t_n\}$ 是 WF-netPN = (P, T, F) 的变迁集. $\forall a, b \in T$,变迁之间的因果关系记为$'\rightarrow w'$,$a \rightarrow_w b$,当且仅当:

(1) $T_{\text{LineNumb}}(a) = T_{\text{LineNumb}}(b)$,

(2) $T_{\text{Receive}}(a) = T_{\text{Send}}(b)$ 。

定义 5(并发关系) : 设 Traces = $\{t_1, t_2, \cdots, t_n\}$ 是 WF-net PN = (P, T, F) 的变迁集. $\forall a, b \in T$,变迁之间并发关系记为 '$||_w$' , $a||_w b$ 当且仅当: $\exists c \in$ Traces

(1) $(c\rightarrow a, c\rightarrow b) \vee (a\rightarrow c, b\rightarrow c)$

(2) $T_{\text{LineNumb}}(a) \neq T_{\text{LineNumb}}(b)$,

(3) $(T_{\text{Send}}(a) = T_{\text{Send}}(b) = T_{\text{Receive}}(c)) \vee$

$(T_{\text{Receive}}(a) = T_{\text{Receive}}(b) = T_{\text{Send}}(c))$。

定义 6(选择关系):设 Traces = $\{t1, t2, \cdots, tn\}$ 是 WF-net $PN = (P, T, F)$ 的变迁集. $\forall a, b \in T$,变迁之间选择关系记为'+',$a+b$ 当且仅当: $\exists c \in$ Traces

(1) $(c\rightarrow a, c\rightarrow b) \vee (a\rightarrow c, b\rightarrow c)$

(2) $T_{\text{LineNumb}}(a) = T_{\text{LineNumb}}(b)$,

(3) $(T_{\text{Send}}(a) = T_{\text{Send}}(b) = T_{\text{Receive}}(c)) \vee$

$(T_{\text{Receive}}(a) = T_{\text{Receive}}(b) = T_{\text{Send}}(c))$。

对于相同用例(场景)的 CPN 被迭代合并,以获得用例的集 CPNs,其颜色用于区分不同输入场景。算法 PM-EL 描述了这一步操作。它可以输入多个执行序列,并且增量的生成一个可表示系统行为的 CPNs。首先,算法逐行扫描执行序列,如果是新执行序列,则分析每一个执行迹,创建一个新库所。其中表示执行迹的所有库所都具有相同的颜色,这些颜色帮助我们区分场景,变迁表示场景的执行。算法还可以创建变迁,它通过分析前面的执行迹和当前的执行迹,为每个库所关联相应的变迁(UML 操作符)。该算法主要通过执行迹内部属性来进行结构化模型的提取。如用线程的 ID 来区分线程,如果这些执行迹属于同一线程,则可以通过在不同的序列中对以前的迹和当前的迹进行简单的比较,然后识别操作符"SEQ"和"ALT"。通过分析线程号信息识别主线程及其子线程来检测操作符"PAR"。如果执行迹已经存在并且属于相同的执行序列(相同的场景),那么与当前线对应的变迁将被标记为临时循环。如果一个变迁已经被标记为一个临时循环,就会更改变迁的关联操作符为"LOOP"。PM-EL 算法如下所示。

算法 PM-EL

1.Data: Trace trace set;

2.Result: CPN cpn;

3.Place NewPlace;

4.Color New_Color;

5.CPN New_CPN;

6.For each trace T in trace set New_Color = creatNewCPNColor( ); //为每一条执行序列创建一个 CPN 颜色

7.For each activity a in T New_Place = creatPlace WithColor(a, New Color);

8.If(isCPNImpty( )) /若是 CPN 为空,直接添加库所到 CPN 中

9.Add_Place_To_CPN(New Place);

10.else If(is Not New(a)) // New Placea 不是新的

11.If(is ln Same Trace(a)) /已经存在并且属于相同的执行序列即相同的场景

12.If(is Loop Marked(New CPN))

13. change _ pervious _ Transition (LOOP, New _ CPN); //修改变迁关联 'LOOP'

14.else

15.mark pervious_Transition(LOOP, New CPN);

16.EndIf

17.else

18.If(is The Same thread( )) //属于在同一线程中

19. If (previous place have Transition (New CPV)) //库所前已有变迁

20. Change _ Transition _ To _ CPN ('ALT, New CPN); //修改变迁类型为'ALT'

21. Add _ Place _ To _ CPN (New Place, New CPN);

22.else Add_Place_To_CPN(New Place, New CPN);

23. Add _ Transition _ To _ CPN ('SEQ, New _ CPN);

24.End If

25.else

26.If(is previous Main ThreadID( )) /如果前一个线程 ID 为主线程

27.Add Transition To CPN('PAR, New CPN);

28. Add Place To CPN (New Place, New CPN); //加入前置变迁 'par'

29.else

30.Add Place To CPN(New Place, New CPN);

31.End If

32.End If

33.End If

34.End If
35.End Foreach
36.End Foreach
37.return New CPN

3　仿真实验

在本节中,我们选择了一个信息查看程序示例。使用应用程序对多场景登录以及信息查看进行操作并产生日志。信息查看程序检测登录类型如果是普通用户登录,使用系统的登录线程(场景 1) ; 员工登录增加两个部分,校验工号以及工种,每部分都是由不同的线程来解决(场景 2) ; 管理员账号登录,需要对登录 IP 地址进行检测(场景 3) ; 普通用户登陆后查看信息(场景 4) ; 员工登录后查看信息(场景 5) ; 管理员登陆后查看信息并对信息进行过滤(场景 6) 。不同用户登陆后查看信息是不同的。经过以上操作获得系统日志文件,这个文件包含系统当前运行时被检测类的对象的所有创建和销毁事件。对象请求的方法调用和返回事件也同时被记录。依据执行的父 ID 对日志进行聚类并记录执行次数,表 1 中数据字典对日志中的方法名及对象名进行相应替换,生成的最终增强日志如表 2 所示。

表 1　事件坐标对应方法或对象

编号	方法或对象	编号	方法或对象	编号	方法或对象
1	pack1: LoginThread: Login	9	pack: compareInfo2: compareWorkNum	17	pack2: InfThread: News
2	checkUser( )	10	getResult1( )	18	queryLocalNews( )
3	pack1: ClientOject: ClientObj	11	getResult2( )	19	getNews( )
4	getUserInfo(User [ ] userinf)	12	pack1: compareInfo: compareJobType	20	querySpecialNews( )
5	checkUserType( )	13	pack1: compareInfo2: compareWorkNum	21	getSpecialNews( )
6	getResult( )	14	pack: compareIP: checkStaticIP	22	queryAllNews( )
7	order( )	15	getResult3( )	23	screenAllNews( )
8	pack: compareInfo: compareJobType	16	pack1: compareIP: checkStaticIP	24	getAllNews( )

表 2　增强日志详情

编号	执行序列	执行次数
Trace1	L0.(0: 1,2,3) ,L1.(0: 3,4,1) ,L2.(0: 1,5,1) ,L3.(0: 1,6,3) ,L0.(0: 1,2,3)	437
Trace2	L0.(0: 1,2,3) ,L1.(0: 3,4,1) ,L2.(0: 1,5,1) ,L4.(0: 1,7,8) ,L5.(0: 1,7,9) ,L6.(1: 12,10,1) ,L7.(2: 13,11,1) ,L3.(0: 1,6,3) ,L0.(0: 1,2,3)	306
Trace3	L0.(0: 1,2,3) ,L1.(0: 3,4,1) ,L2.(0: 1,5,1) ,L8.(0: 1,7,14) ,L9.(0: 16,15,1) ,L3.(0: 1,6,3) ,L0.(0: 1,2,3)	419
Trace4	L0.(0: 1,2,3) ,L1.(0: 3,4,1) ,L2.(0: 1,5,1) ,L3.(0: 1,6,3) ,L10.(0: 3,18,17) ,L11.(0: 17,19,3) ,L0.(0: 1,2,3)	488
Trace5	L0.(0: 1,2,3) ,L1.(0: 3,4,1) ,L2.(0: 1,5,1) ,L4.(0: 1,7,8) ,L5.(0: 1,7,9) ,L6.(1: 12,10,1) ,L7.(2: 13,11,1) ,L3.(0: 1,6,3) ,L12.(0: 3,20,17) ,L13.(0: 17,21,3) ,L0.(0: 1,2,3)	520
Trace6	L0.(0: 1,2,3) ,L1.(0: 3,4,1) ,L2.(0: 1,5,1) ,L8.(0: 1,7,14) ,L9.(0: 16,15,1) ,L3.(0: 1,6,3) ,L14.(0: 3,22,17) ,L15.(0: 17,23,17) ,L16.(0: 17,24,3) ,L0.(0: 1,2,3) ,	510

文章在此利用 Java 编程实现了算法,其算法复杂度为 O(m^* n) ,图形化界面如图 3 所示,功能包括 txt、rtf 格式日志导入及算法结果保存,CPN 结果以节点形式输出。

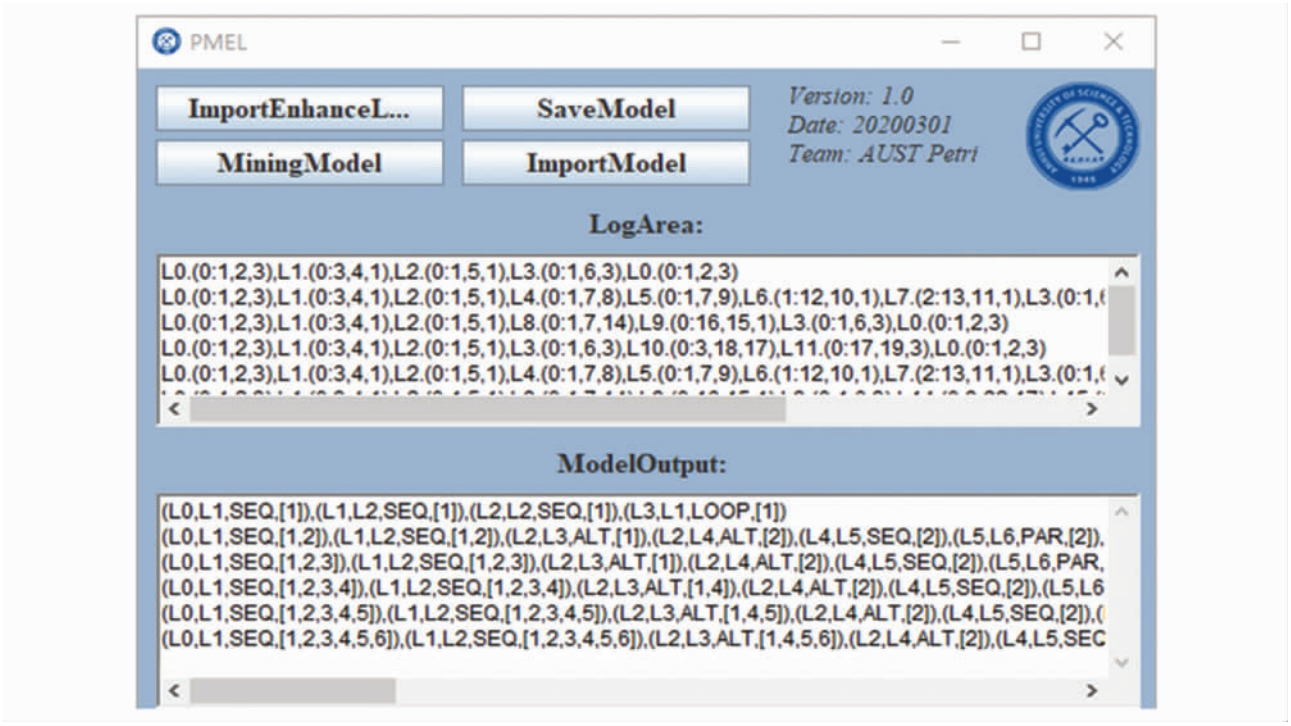

图 3 算法的实现插件(PMEL) 及挖掘结果展示

图 4 形象的表达了算法的执行步骤,且每条执行序列均对应不同场景进行了遍历迭代,故日志的每一条执行序列与模型中的可达路径都是一一对应的。

通过定义 4、5、6 的描述可知 L_3、L_8、L_4, $L_1$0、$L_1$2、$L_1$4 为选择关系,L_6、L_7 为并发关系。在 PM-EL 算法结果中这些结构均被识别且通过颜色区别不同场景;

通过 CPNtool4.0.1 对 PM-EL 所获结构模型进行仿真实验(见图 5) ,此 CPN 模型为可运行的。另外通过定义 2 及图 1、图 2 表述的映射规则,可将此 CPN 模型转化为 UML 模型,可更加直观展示系统的运行,并验证此方法结果的可扩展性。

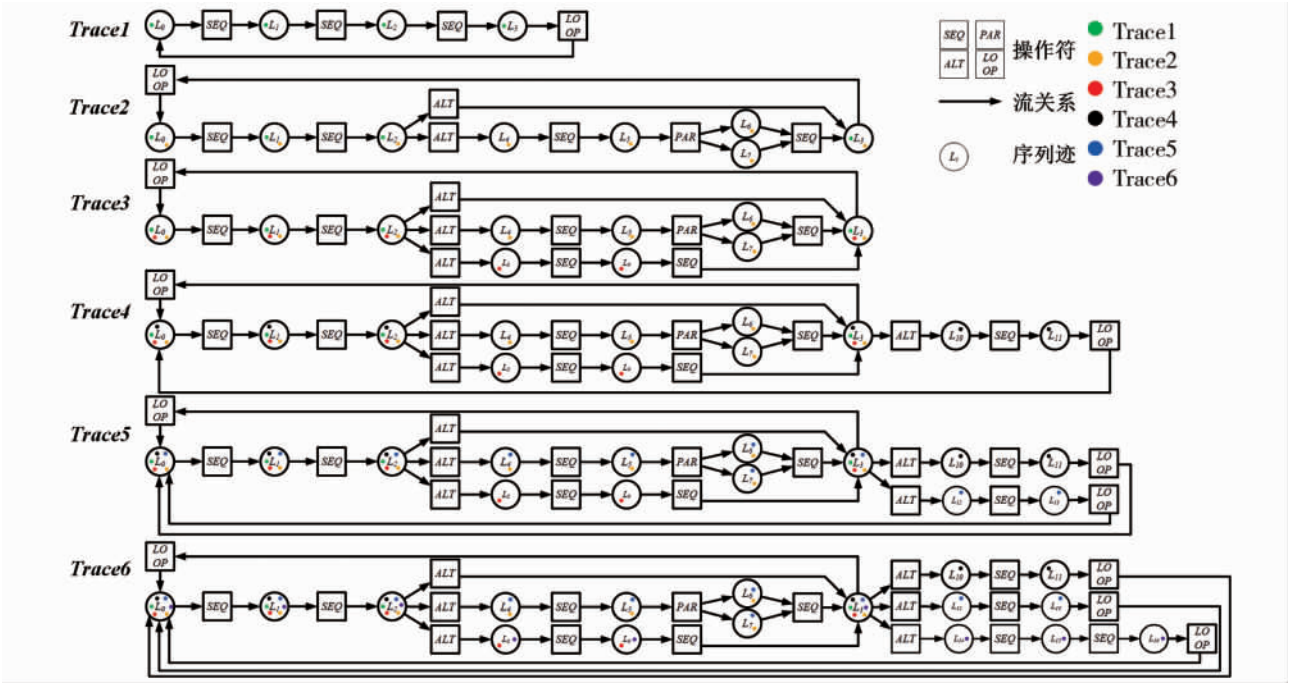

图 4 过程挖掘算法 PM-EL 执行示例图

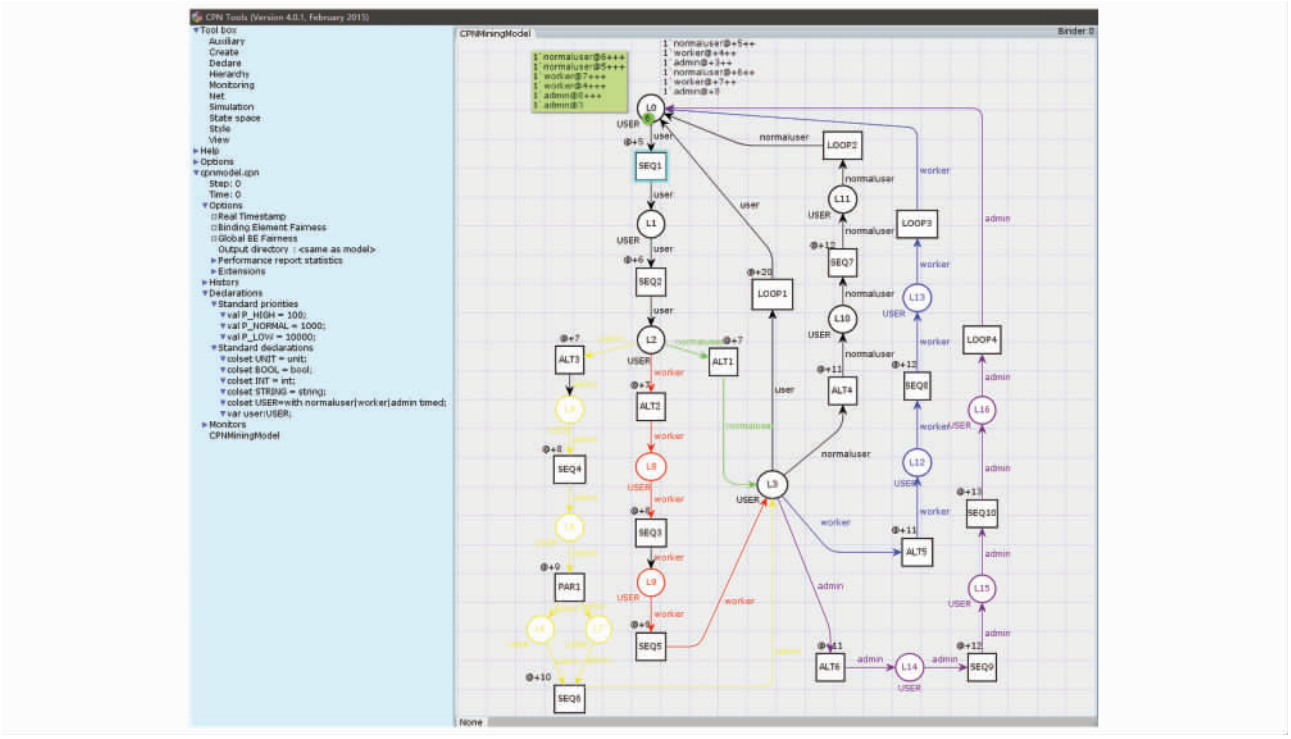

图 5　使用 cpn tools 4.0.1 对结果模型进行模拟可行

在此将日志导入 Prom 平台中,并利用 Alpha Miner 及 IDHM (interactive Data - aware Heuristic Miner) 进行挖掘,以突出 PM-EL 算法在不完备日志基础上对模型结构的识别能力。

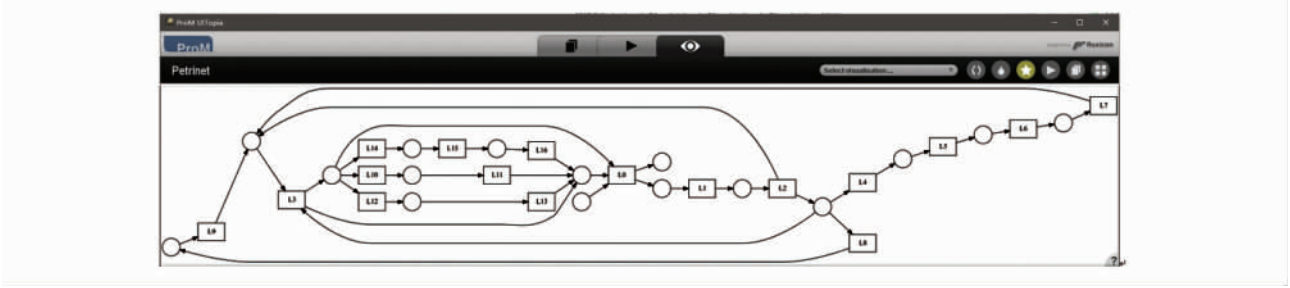

图 6　Alpha Miner 挖掘结果

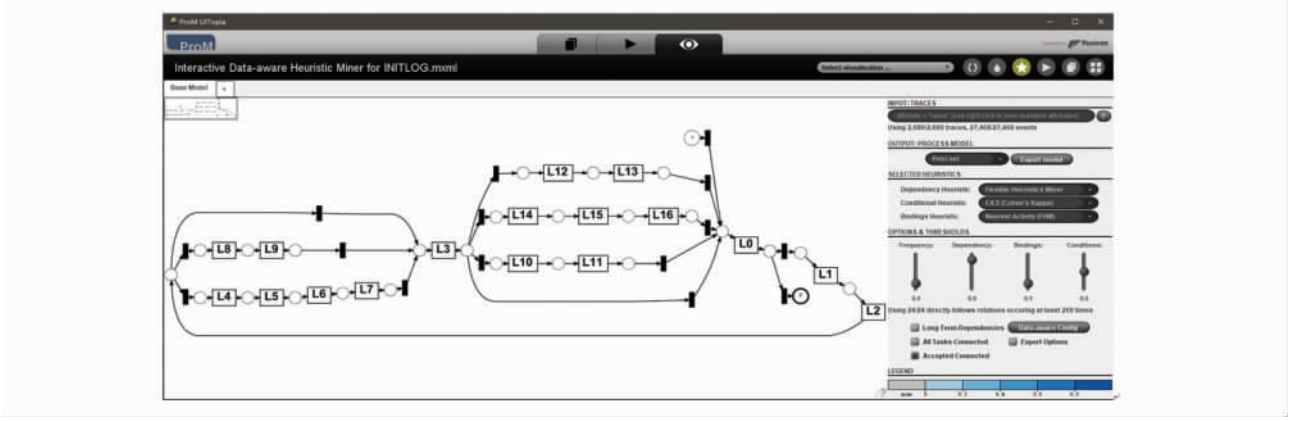

图 7　IDHM 挖掘结果(Frequency 为 0.1,Dependency 为 0.9,Binding 为 0.1,Conditions 为 0.5)

三个算法实验结果均有完整回路(即循环结构) ,现就结果中选择结构和并发结构数量进行对比,如表 3 所示。

表3 选择、并发结构数量对比

	PM-EL	Alpha	IDHM
选择结构	6	7	7
并发结构	1	0	0

4 结论

文章通过分析数据挖掘与过程挖掘的异同,说明日志的重要性,然后提出增强日志的概念。将系统日志中除事件执行外的一些可利用信息加入事件日志,基于增强日志提出过程挖掘算法PM-EL,依赖事件执行的附加属性识别结构关系。依据定义4、5、6,结果过程模型也是符合源增强日志的。通过UML序列图的转换说明了算法结果的可扩展性。即从增强日志中挖掘出附带场景信息的过程模型。

通过PM-EL算法所提取的过程模型可以为后续的模型修复及优化工作提供支持。在此文章的研究基础之上,未来可以利用增强日志提升模型修复精度,将模型优化细分为多场景有针对性的优化。

参考文献:

[1] F GORUNESCU. Data Mining: Concepts, Models and Techniques [M]. Blue Publishing House, ClujNapoca, 2011: 30-35.

[2] W VAN DER AALST, T WEIJTERS, L MARUSTER. Workflow mining: discovering process models from event logs [J]. IEEE Trans. Knowl. Data Eng. 16 (2004): 1 128-1 142.

[3] L WEN, J WANG, W VAN DER AALST, et al. A novel approach for process mining based on event types [C]//J. Intellig. Inform. Syst. 32 (2009): 163-190.

[4] D WANG, J GE, H HU, et al. Discovering process models from event multiset [J]. Exp. Syst. Appl. 39 (2012): 11 970-11 978.

[5] J COOK, A L WOLF. Discovering models of software processes from event-based data [M]. ACM Transac. Softw. Eng. Methodol. (TOSEM) 7 (1998) 215-249.

[6] K G COMAN, A M ODLYZKO. Internet growth: is there a Moore's law for data traffic? [J]. Handbook of Massive Data Sets, Kluwer, 2001: 47-93.

[7] BROUCKE S K L M V, VANTHIENEN J, BAESENS B. Declarative process discovery with evolutionary computing [C]// 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2014: 2 412-2 419.

[8] MARUSTER L, WEIJTERS A, VAN DER AALST W M P, et al. A rule-based approach for process discovery [J]. Data Mining & Knowledge Discovery, 2015, 13 (1): 67-87.

[9] LEEMANS S J J, FAHLAND D, VAN DER AALST W MP. Scalable process discovery and conformance checking [J]. Software & Systems Modeling, 2018, 17(2): 1-33.

[10] WEIJTERS A, VAN DER AALST W M P, DE MEDEIROS A K A. Process mining with the heuristics miner-algorithm [J]. Technische Universiteit Eindhoven, Tech. Rep. WP, 2006, 166: 1-34.

[11] WEIJTERS A T, RIBEIRO J J. Flexible heuristics miner(FHM) [C]//Proceedings of IEEE Conference on Computational Intelligence and Data Mining. Washington, D.C., USA: IEEE, 2011: 310-317.

[12] LEEMANS S J J, FAHLAND D, VAN DER AALST W MP. Discovering block-structured process models from incomplete event logs [C]//Proceedings of International Conference on Applications and Theory of Petri Nets and Concurrency. Berlin, Germany: Springer-Verlag, 2014: 91-110.

[13] TAX N, SIDOROVA N, HAAKMA R, et al. Mining local process models [J]. Journal of Innovation in Digital Ecosystems, 2016, 3(2): 183-196.

[14] 邵叱风.基于流程挖掘的并行优化算法[J].赤峰学院学报(自然科学版), 2019, 35(10): 66-70.

[15] 方贤文.Petri网行为轮廓理论及其应用[M].上海:上海交通大学出版社, 2017: 39-40.

[16] VAN DER AALST W MP. Process mining: discovery, conformance and enhancement of business processes [M]. Springer Publishing Company, Incorporated, 2011: 56-89.

[17] ZHANG L, ZEIGLER B P, LAILI Y. Introduction to Model Engineering for Simulation [M]. Model Engineering for Simulation, 2019: 43-56.

[18] CHUANYI LI, JIDONG GE, LIGUO HUANG, et al. Process mining with token carried data [J]. Information Sciences, 2016: 34-48.

(责任编辑:李 丽)