基于增广 Petri 网生成受控日志

邵叱风[1,2]，方贤文[1+]，王吴松[1]

（1. 安徽理工大学 数学与大数据学院，安徽 淮南 232001；
2. 安徽科技学院 信息与网络工程学院，安徽 蚌埠 233030）

摘 要：为减少冗余日志，降低事件约束不可控对算法评估及验证的影响，提出基于可达状态的随机选择生成受控日志的方法。利用增广 Petri 网为系统建立模型，依据模型中库所与变迁的结构关系及标识分布构建输入矩阵；基于 Petri 网可达状态分析方法，随机选择触发可发生变迁，记录变迁序列；对记录进行受控分析，拼装生成多重集事件日志和 XES 标准日志。实现相关日志生成工具，利用 BPIC2020 数据进行实验，与已有工具进行比较，实验结果表明了工具的效率和有效性。

关键词：日志冗余；事件约束；佩特里网；可达状态；受控分析

中图法分类号：TP391.9 **文献标识号**：A **文章编号**：1000-7024（2022）03-0876-10

doi：10.16208/j.issn1000-7024.2022.03.038

Generation of controlled logs based on extended Petri Net

SHAO Chi-feng[1,2], FANG Xian-wen[1+], WANG Wu-song[1]

(1. School of Mathematics and Big Data, Anhui University of Science and Technology, Huainan 232001, China;
2. College of Information and Network Engineering, Anhui Science and Technology University, Bengbu 233030, China)

Abstract: To reduce the redundant logs and the impact of uncontrollable event constraints on algorithm evaluation and verification, a method for generating controlled logs based on random selection of the available states was proposed, and the extended Petri Net was used to build a model for the system, and an input matrix was constructed based on the structural relationship between the library and the transformations and the token distribution in the model. Based on the available states analysis method of the Petri Net, random selection triggered the occurrence of transitions and the transition sequence was recorded. Controlled analysis of the records and collocation was carried out to generate multiple set event logs and XES standard logs. The relevant log generation tool was implemented, and experiments were conducted using BPIC2020 data, in which existing tools were adopted for comparison. Experimental results show the efficiency and effectiveness of the proposed approach.

Key words: log redundancy; event constraints; Petri Nets; reachable states; controlled analysis

0 引 言

Petri Net[1]是常用系统建模语言之一，广泛应用于过程挖掘[1]、模型修复[3]、建模优化[4]及算术计算[5]等诸多领域。Prom 是一个过程挖掘相关插件平台，可以利用 XES[6]标准日志及挖掘算法生成结果，并利用 Petri 网等建模语言对挖掘结果进行可视化处理[7]。为了更好地测试和评估过程挖掘算法，需要人工生成的日志。文献［8］提出了一种日志生成方法，可用于随机生成多角度流程模型并进行仿真，以合成多角度事件日志。文献［9］可以根据用户定义的控制流规范随机自动地生成过程树和事件日志。它采用了一种通用的两步方法：生成一个流程树，然后将该树仿真为事件日志，在文献［10］中进行了描述。但通常需要生成日志解决的问题如：通过模拟所选过程模型来生成事件日志，然后对此日志应用了新颖的过程发现算法，最后将发现的模型与初始模型进行比较发现了特殊结构。但是在现实事件日志中，经常缺少这样的“特殊结构”。因此，能够以受控方式生成模型和日志很重要。文献［11］

收稿日期：2020-10-30；**修订日期**：2021-01-25

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

作者简介：邵叱风（1995-），男，安徽合肥人，硕士研究生，CCF 学生会员，研究方向为 Petri 网和过程挖掘；＋通讯作者：方贤文（1975-），男，河南信阳人，博士，教授，CCF 会员，研究方向为 Petri 网和可信软件；王吴松（1995-），男，安徽合肥人，硕士研究生，研究方向为 Petri 网和变化挖掘。E-mail: fc_shao@126.com

提出了一种基于 BPMN 模型生成日志的方法，但目前 Petri 网与 BPMN 之前的转换仍有诸多不便。在少量日志挖掘模型时，为提升挖掘能力，文献［12］提出一种基于增强日志的过程挖掘方法，加强了算法对多属性日志的依赖并提升对并发结构的识别能力，但在重构日志时较为复杂。在日志生成后可能存在一些与预期或模型有差距需修改的内容，文献［13］提出了一种基于神经网络的日志修复方法，在识别大量不同活动标签效率表现一般。

本文利用增广 Petri 网直接模拟系统运行并实现一种用于生成 XES 标准事件日志的具体方法，且通过直接修改多重集日志来降低生成预期日志的难度。

1　准备知识

本节介绍了部分定义用以辅助理解日志生成工具中关于网的输入、运行以及日志的导出。限于篇幅原因 Petri 网相关定义见文献［1］。在此选用增广 Petri 网作为建模语言，可在建模过程中以更简洁的结构便捷模拟现实条件。

定义 1[1]　带抑制弧 Petri 网：一个带抑制弧的 Petri 网是一个五元组 $\sum=(S,T,F,I,M)$，其中，(S,T,F) 是一个网，M 是网的一个标识，$I\subset S\times T$ 称为抑制弧集，$I\cap F=\Phi$。

即 $\forall s\in S\wedge\forall t\in T:(s,t)\in F\rightarrow(s,t)\notin I$。

对于 $t\in T$，如果

(1) $\forall s\in S:(s,t)\in F\rightarrow M(s)\geqslant 1$

(2) $\forall s\in S:(s,t)\in I\rightarrow M(s)=0$

则 t 在标识 M 有发生权，记为 $M[t>$。

若 $M[t>$，则变迁 t 在 M 可以发生，t 在 M 发生产生新的标识 M'

$$M'(s)=\begin{cases}M(s)-1, \text{若}(s,t)\in F\wedge(t,s)\notin F\\ M(s)+1, \text{若}(t,s)\in F\wedge(s,t)\notin F\\ M(s), \text{其它}\end{cases}$$

为增加 PSLG（Petri simulation log generation）的实用性，利用输入矩阵对网模型进行输入。对如图 1 所示的 6 种关系和库所中的 *token* 数量均在输入矩阵中进行了定义。

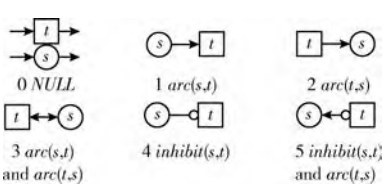


图 1　输入矩阵中库所与变迁的 6 种关系

定义 2　输入矩阵：一个输入矩阵 M 是基于矩阵元素扩展生成的，元素 M_{ij} 定义了库所 s_i 与变迁 t_j 的关系。最后一位数字代表两个元素之间的流弧关系，其中 0-5 分别表示 *NULL*、$arc(s,t)$、$arc(t,s)$、$arc(s,t)$ 和 $arc(t,s)$、$inhibit(s,t)$、$inhibit(s,t)$ 和 $arc(t,s)$，元素 M_{ij} 剔除最后一位数字即为当前关系库所 s_i 中 *token* 的数量，一行元素 *token* 之和即当前系统 s_i 中 *token* 的数量，输入矩阵及元素拓展如图 2 所示。

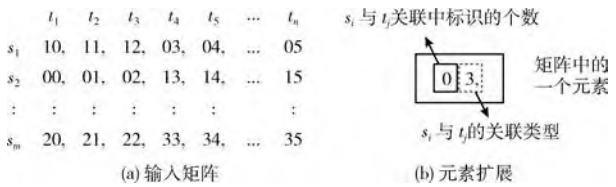


图 2　输入矩阵及元素扩展

定义 3　标签网：一个标签 Petri 网是一个四元集 $N=(P,T,F,\lambda)$，其中 (P,T,F) 是一个 Petri 网，$\lambda:T\rightarrow\mathcal{A}\cup\{\tau\}$ 是一个指定变迁标签的函数，其中 $\mathcal{A}$ 表示领域标签，τ 表示特殊标签且 $\tau\notin\mathcal{A}$。

如果 $\lambda(t)\neq\tau$，$t\in T$，那么 t 是可观测的，反之 t 是静默变迁。标签的执行语义是根据状态和状态转换定义的。标签网的状态由标识的概念俘获。

定义 4　网的标识：一个标签网 $N=(P,T,F,\lambda)$ 的标识是一个多重集 $M\in\mathcal{B}(P)$。

标签网 $N=(P,T,F,\lambda)$ 的标识 M 一般表示 *Token* 到库所的分配。如 $M(p)$ 分配 *Token* 在库所 p，$p\in P$。

定义 5　网系统：一个网系统，或者一个系统是一个三元集 $S=(N,M_{ini},M_{fin})$，其中 $N=(P,T,F,\lambda)$ 是一个标签网，M_{ini} 属于 $\mathcal{B}(P)$ 是网 N 的一个初始标识，M_{fin} 属于 $\mathcal{B}(P)$ 是网 N 的一个最终标识。

定义 6　可达标识图：设 $\sum=(S,T,F,M_0)$ 为一个有界 Petri 网。$\sum$ 的可达标识图定义为一个三元组 $RG(\sum)=(R(M_0),E,P)$，其中 $E=\{(M_i,M_j)\mid M_i,M_j\in R(M_0),\exists t_k\in T:M_i[t_k>M_j\}$，$P:E\rightarrow T,P(M_i,M_j)=t_k$，当且仅当 $M_i[t_k>M_j$ 称 $R(M_0)$ 为 $RG(\sum)$ 的顶点集，E 为 $RG(\sum)$ 的弧集；若 $P(M_i,M_j)=t_k$，则称 t_k 为弧 (M_i,M_j) 的旁标。

PSLG 含丰富的日志导出功能，支持定义 7 系统记录、事件日志以及 XES 格式日志文件的导出，分别用作系统执行过程的展示、多重集日志的展示以及 Prom 等工具的直接可用日志。

定义 7　系统记录、事件日志：迹是一个有限的标签序列。系统记录为迹的可重复记录集，一个事件日志是一组迹上的多重集。假定迹上的每个标签都代表一个事件，即一个活动的发生。此外假定标签在迹中出现的顺序表出了事件发生的顺序，即 $j>i$，那么位于迹 i 处的标签表示的事件发生在位于迹 j 处的标签表示的事件之前。

系统记录形如：$R_1=[<a,c,d,h>,<a,c,d,h>,<$

$a,b,c,d,f,g>,<a,c,d,h>]$；

事件日志形如：$L_1=[<a,c,d,h>^{10},<a,b,c,d,f,g>^{7},<a,c,d,e,b,c,d,f,g>^{5}]$。

为统一软件的日志输入格式，文献［6］提出了 XES 格式日志文件，是 XML 语言的一种扩展。主要由分层组件（Hierarchicalcomponents）、属性组件（Attributecomponent）、全局属性（Globalattributes）、分类器（Classifiers）和扩展（Extensions）共 5 个部分构成。

2 日志生成方法及实现

现有的日志生成方法大多是随机生成的模型日志，大量的冗余数据导致事件结构之间约束不可控；或者生成日志的方式较为繁琐，需要学习新的编程语言。以上方法在日志生成时难以生成实验需要的特定的满足模型结构的日志，并凭此去检验算法在某些行为模式上的识别能力。

在此提出对系统直接仿真生成指定条数的执行迹，后期处理生成多重集事件日志及 XES 标准日志的方法。主要过程为：①系统建模；②推导输入矩阵；③初始化输入模型；④手动或随机执行模型过程；⑤统计执行记录生成事件日志；⑥XES 标准日志的转换。其中系统建模部分即利用增广 Petri 网对现有系统进行模拟，在此不做赘述。

对于大规模负载程序的开发，程序的处理流程并非单一的一条主线，而是错综负载的网状结构。面向对象编程比起面向过程编程更能够应对复杂类型的程序开发；面向对象编程拥有更加丰富的特性（封装、继承、抽象和多态），利用这些特性可以使得代码更加易扩展，易维护，易复用。下面给出具体 Java 应用实现日志生成方法的过程。

2.1 输入矩阵

为了便捷化输入 Petri 网模型，Java 应用采用定义 2 形式的输入矩阵进行模型输入。如图 3 所示利用 split（“\r\n”）分割矩阵得 m 条字符串（即网包含 m 个库所 $s_1,s_2,\cdots s_m$），每条字符串 split(“,”) 可获得 n 个元素（即网包含 n 个变迁 $t_1,t_2,\cdots,t_n$），第 i 条字符串第 j 个元素为库所 s_i 与变迁 t_j 的关系描述，结合 *for* 循环利用定义 1 计算模型中弧的个数，初始化变迁、库所及弧的数组，并依据矩阵元素关联库所和变迁，得到初始化 Petri 网模型。抑制弧、库所流向变迁及变迁流向库所的关联代码如 Code_1 所示。

aArray[*z*][*total*] = *pnlist*[*z*].inhibitor("arc" + *total*, *pArray*[*i*],*tArray*[*l*]);// 添加库所到变迁的抑制弧

aArray[*z*][*total*] = *pnlist*[*z*].arc("arc"+*total*,*pArray*[*i*], *tArray*[*l*]);// 添加库所到变迁的流弧

aArray[*z*][*total*] = *pnlist*[*z*].arc("arc" + *total*,*tArray*[*l*], *pArray*[*i*]);// 添加变迁到库所的流弧

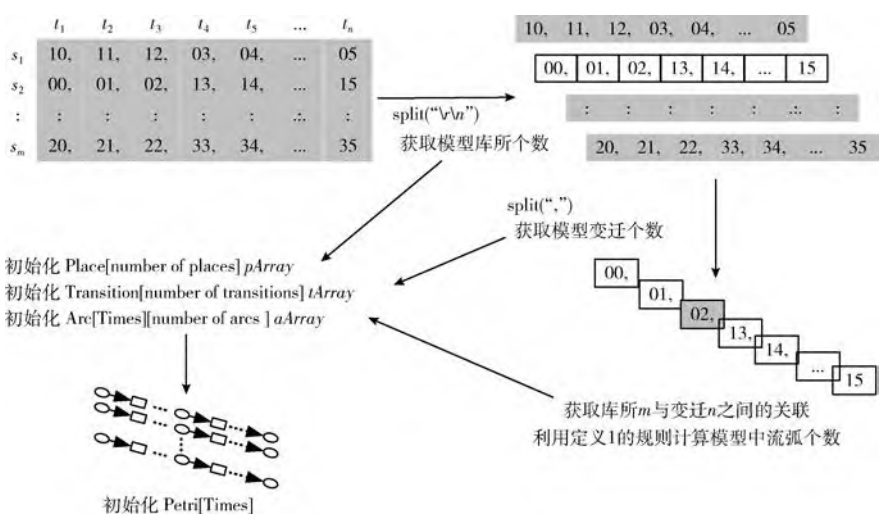


图 3 利用输入矩阵初始化 Petri 网的原理

2.2 网的运行

为保证流程执行的随机性，定义 AutoRun() 方法，使用 ArrayList<Transition>记录下 Petri 网当前状态所有具备触发条件的变迁，*Math*.random() 在数组大小范围内生成随机数 k，触发数组中下标为 k 的元素包含的变迁后进入下一个可达状态，并对触发的变迁进行记录，如此循环直到当前状态中无可触发变迁。具体实现的部分代码如 Code_2 所示：

```
(1)for (int j=1; j<pnlist.length; j++) {
(2)   int fireState=1;
(3)   int logLength=0;
(4)   while (fireState>0) {
(5)        ArrayList<Transition> waitRun=new
           ArrayList<Transition>();
(6)        fireState=0;
(7)        for (int i=0; i<pnlist[j].getTransitions().
```

```
        size(); i++) {
(8)     if (pnlist[j].getTransitions().get(i).
canFire()) {
(9)     waitRun.add(pnlist[j].getTransitions().get
(i));
(10)            fireState++;
(11)        }
(12)    }
(13)    if (waitRun.size()>0) {
(14)    int max=waitRun.size();
(15)    int ranNum=(int)(Math.random() * max);
(16)    waitRun.get(ranNum).fire();
(17)    logArea.append(waitRun.get(ranNum).
get Name() + ",");
(18)    logLength++;
(19)    }
(20) }
(21)}
```

fireState 为标识可触发变迁的个数，*logLength* 为统计每条执行记录的长度。定义 *canFirestate*=1，进入 while 循环，初始化可触发变迁缓存数组 *waitRun*，标记当前可触发变迁个数为 0。Code_2(7-12) 对网中变迁进行遍历，方法 canFire() 判断变迁能否触发，如果可以数组 *waitRun* 记录下可触发变迁且 *fireState*=*fireState*+1，Code_2(13-19) 如果 *waitRun* 含有可触发变迁，利用 *Math*.random() * *max* 在数组大小范围内生成随机数，触数组中对应下标的元素所含变迁，到达下一可达状态，循环至终态即遍历网中变迁无一可触发，*fireState*=0 结束循环。

2.3　多重集日志

为统计网运行记录库中每条记录出现频次，增加了普通的事件记录转为多重集日志的功能。编写工具类 StringSameCount，利用 HashMap＜String,Integer＞对事件记录进行统计，利用＜*key*,*value*＞判断是否包含此条记录，如果有则计数增加，如果没有增加此条记录（如图 4 所示）。具体实现代码如 Code_3 所示。

```
(1)public class StringSameCount {
(2)  private HashMap map;
(3)  private int counter;
(4)  public StringSameCount() {
(5)      map = new HashMap<String, Integer>();
(6)  }
(7)  public void hashInsert(String string) {
(8)      if (map.containsKey(string)) {
(9)          counter = (Integer) map.get(string);
(10)         map.put(string, ++counter);
(11)     } else {
(12)         map.put(string, 1);
(13)     }
(14) }
(15) public HashMap getHashMap() {
(16)     return map;
(17)  }
(18)}
```

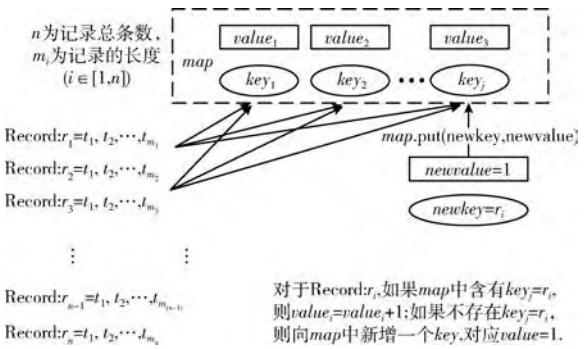


图 4　网运行记录统计方法

Code_3 定义了一个日志迹计数的功能，通过统计网运行记录，*map* 记录每条执行迹的执行次数，生成一个迹上的多重集事件日志。Code_3(8-13) 判断当前 *map* 中是否含有当前字符串 r_i（执行记录），如果有则对此 *key*=r_i 对应的 *value* 值进行加 1（即增加一次执行次数），如果没有则新增 *key*=r_i，*value*=1（即新增一种执行迹，并记执行次数为 1）。

2.4　XES 标准日志

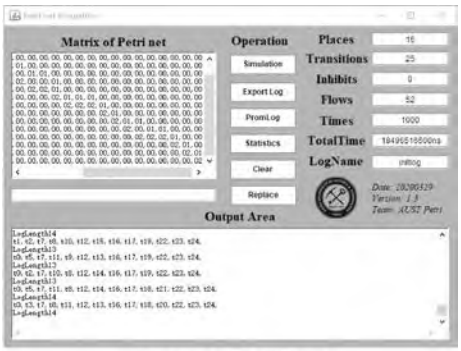
Prom 是一个过程挖掘插件平台，XES 为该平台日志支持标准。为减少日志后期处理，增加导出日志的可用性，增加了 XES 标准日志导出功能。主要是遍历系统执行记录，将每条执行记录记为一个 Trace，记录中的事件记为一个 Event，并对 *key* ="concept:name"、*key* ="time:timestamp" 进行赋值，再与文件的头文件进行拼接，生成 XES 格式日志。具体实现代码如 Code_4 所示：

```
(1)public static String pLog(String txtLog, String lo-
gName) {
(2)  String file = headerOfFile;// XES 格式日志头
文件
(3)  String[] Ilist=txtLog.split("\r\n");
(4)  for (int i=0; i<Ilist.length; i=i+2) {
(5)      file=file+"<trace>\r\n<string key="
+ "\"concept:name\" value=\"Case"+caseID+".0"/>";
(6)      caseID++;
(7)      String[] splitLog=Ilist[i].split(",");
(8)      for (int j=0; j<splitLog.length; j++) {
(9)          SimpleDateFormat Pdf=new Simple-
```

```
DateFormat("yyyy-MM-dd");// 设置日期格式
(10) SimpleDateFormat Ldf = new SimpleDateFormat("HH:mm:ss");// 设置日期格式
(11) int x=1+(int)(Math.random() * 1000);
(12) file=file+"<event>\r\n<string key="
+"\"org:resource\" value=\"UNDEFINED\"/>\r\n"
+" <date key=\"time:timestamp\" value=\""
+Pdf.format(new Date())+"T"
+Ldf.format(new Date())+"."+x+"+01:00\"/>
\r\n"
+ " <string key=\"concept:name\" value=\""+
splitLog[j]+"\"/>\r\n"
+ " <string key="lifecycle:transition" value"
+"="complete"/>\r\n"
+ " </event>";
(13) }
(14) file=file+" \r\n</trace>\r\n";
(15) }
(16) file=file+"</log>";// XES格式日志结尾
(17) return file;
(18)}
```

Code_4 中，*txtLog* 为系统运行记录，*logName* 为日志导出时文件的名称，*headOfFile* 为 XES 标准日志拼装时的头部文件（可自定义）。Code_4(3) 按行分割系统运行记录，获得记录列表；Code_4(4-15) 为日志中层信息的拼装，其中 Code_4 (12) 完成中层信息内部 event 的拼装，通过遍历运行记录中的活动，完成事件名称及执行时间的赋值，Code_4(14) 补全 < trace >< /trace > 标签；Code_4(16) 补全 < log >< /log > 标签。

为调节网中部分变迁的发生频率、屏蔽部分运行路径、增加日志噪音等，通过编辑定义 7 形式多重集日志的方式进行实现。对于记录（t_0，t_1，t_3，t_2，t_5，t_6，t_8，t_9，t_{11}，t_{13}，t_{14}；47；），t_0，t_1，t_3，t_2，t_5，t_6，t_8，t_9，t_{11}，t_{13}，t_{14}；用作 XES 日志 *event* 组装源数据，数字 47 代表此条迹的执行次数即在日志中的条数。依据输入矩阵创建日志界面如图 5（a）所示，多重集日志转 XES 日志界面如图 5（b）所示。



(a) 依据输入矩阵创建日志

(b) 普通日志转XES格式

图 5 方法实现的界面截图

3 实验评估

PSLG 实现日志生成方法，主要是对系统进行建模，直接仿真模型运行，记录网的运行情况，并通过工具类 OperationOfLog.java、StringSameCount.java 对之进行处理生成多重集事件日志及 XES 标准日志，并支持多重集日志直接转换为 XES 标准日志。相较于 PIPE、CPNTools 和 Tina 等工具，PSLG 更注重于模型信息及执行结果的展示，表 1 给出了输入方式、模型数据、记录生成和结果导出共 4 个方面对比结果。

表 1 工具的多方面对比

插件名	比较形式			
	输入方式	模型数据	记录生成	结果导出
PSLG	矩阵输入	数据展示	多次模拟，随机生成	普通日志，XES 格式
PIPE	绘制输入	图形展示	单次模拟，随机生成	无
CPNTools	绘制输入（编码条件）	图形展示	编程选择输出数据	结果结构复杂，需后期处理
Tina	绘制输入	图形展示	单次模拟，随机生成	无

为检验方法的效率及有效性，基于 BPIC2020 数据中的 Domestic Declarations.xes 和 Request For Payment.xes 两个日志文件，使用 Prom 平台中的 Mine with Inductive visual Miner S. J. J. Leemans 插件挖掘得到如图 6 所示 Petri 模型，

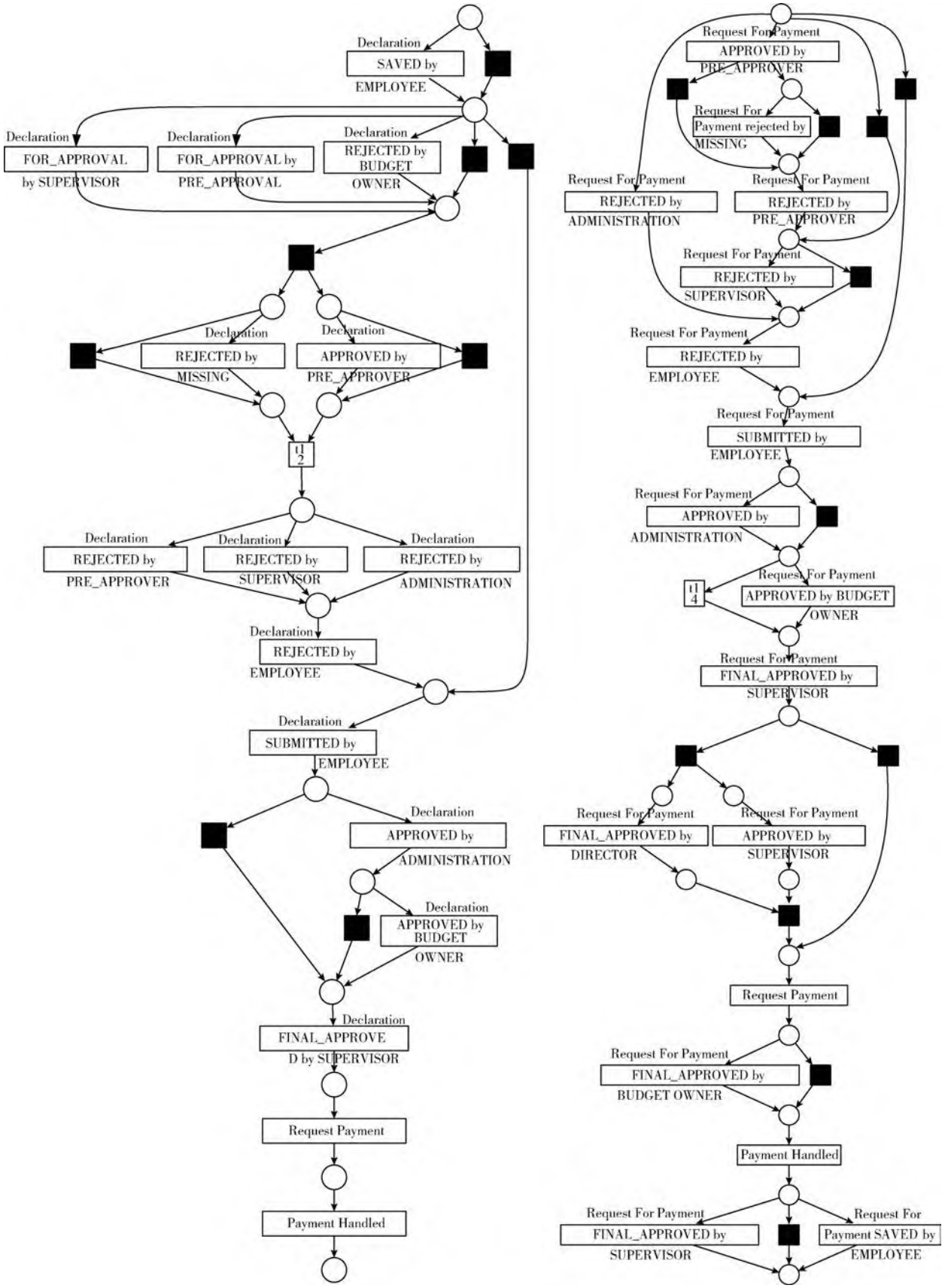


图 6　Domestic Declarations 流程模型（左）与 Request For Payment 流程模型（右）

转换为输入矩阵如下所示。

input matrix of Domestic Declarations

11,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
02,02,01,01,01,01,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,02,02,02,02,00,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,02,01,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,02,00,00,01,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,02,02,00,00,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,02,02,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,02,01,01,01,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,02,02,02,01,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,02,00,00,00,00,00,00,00,00,00,02,01,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,01,01,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,00,01,01,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,02,02,01,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,01,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,01
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02

input matrix of Request For Payment

11,01,00,00,00,01,00,00,01,00,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
02,00,01,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,02,02,02,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,02,02,01,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,02,02,02,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,02,02,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,02,01,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,02,02,01,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,02,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,01,00,00,00,01,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,01,00,00,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,00,01,00,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,00,01,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,01,00,00,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,02,01,00,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,01,01,00,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,02,01,00,00,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,01,01,01,00,00,00,00
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,02,02,02,01,00,00,00

利用 PSLG 工具为这两个模型各生成执行记录 10 000 条；同时构造一个变迁个数为 60 的直链网模型（保证每条执行迹长度相等），并为其生成执行记录 1000 条。利用以上数据对工具的效率进行分析，并对其稳定性、可移植性以及日志的有效性等进行实验分析。所有的测试均在配有 I5-7300HQ 2.5 Ghz 四核处理器和 16 GB 内存的机器上进行，使用 Java SE 1.7 开发包。

3.1 耗时分析

为检验工具运行稳定性，在执行直链模型时，记录下变迁触发时间，对大小为 60×1000 的时间数据绘制如图 7 所示。其中变迁触发时间大多在 100 ns～1000 ns 之间，保持较高的变迁触发稳定性。

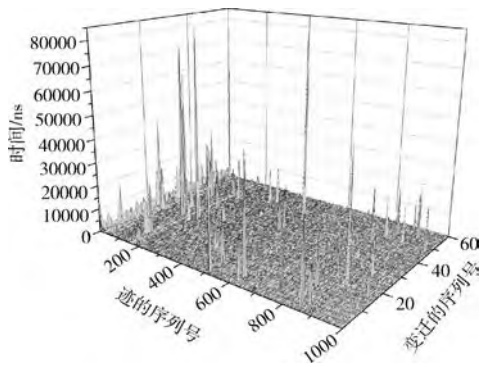


图 7　变迁触发时间展示（60×1000）

分别取迹长度为 10∶10∶60，迹编号为 100∶100∶1000，得到迹执行时间数据 6×10 绘制如图 8 所示，在系统运行不同时间段，相同长度迹生成时间基本不变；分别取迹长度为 10∶10∶60，日志大小为 100∶100∶1000，得到日志生成时间数据 6×10 绘制如图 9 所示，在系统日志生成过程中，对固定的每条迹长度，生成时间与日志大小（迹的条数）成正比；对固定的日志大小，时间与迹的长度成正比。

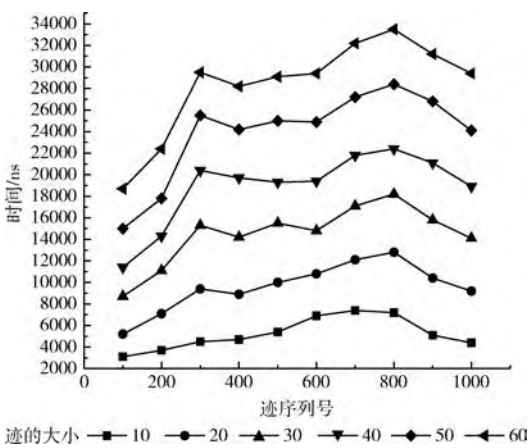


图 8　不同迹的触发时间

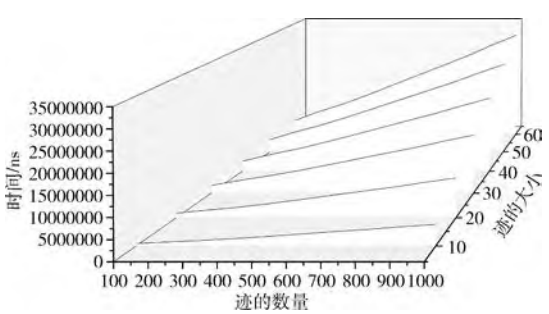


图 9　不同日志大小的生成时间

3.2　日志有效性

为验证日志有效性，首先利用日志挖掘出 Petri 网模型与原模型对比，对于 BPIC 模型生成的事件日志，再次使用 Prom 中 Mine with Inductive visual Miner S. J. J. Leemans 插件，以消除不同算法对挖掘结果的影响。将 BPIC 的两个生成日志导入 Prom 挖掘结果与原模型保持一致。

对两个生成日志进行处理，分别取前 10∶10∶100 条迹组成大小为 10∶10∶100 的共 20 个日志，利用 Compute projected fitness and precision（log and accepting Petri Net）插件分别计算日志与原模型之间的适应度以及精确度，结果如图 10 所示，Domestic Declarations 包含变迁 25 个，Request For Payment 包含变迁 28 个，随日志大小的增加，精确度值逐渐增加至 1；适应度值在一个较高的数值波动，且 Domestic Declarations 一直高于 Request For Payment。

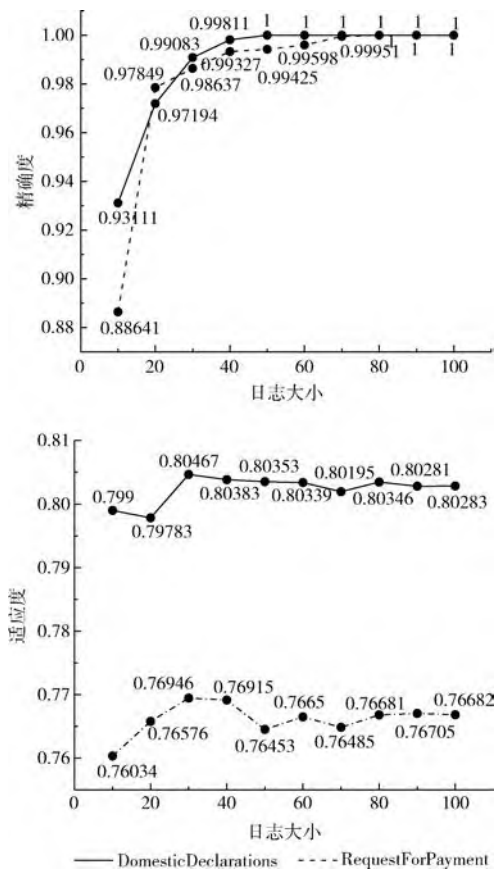


图 10　随日志大小变化的精度和适应度

鉴于日志大小在 100 时 precision 已达到 1，仅对前 100 条迹组成的日志进行时间分析如图 11 所示，前 100 条迹长度各不相同，生成时间在一定范围内对应变化（图 11（a）），日志生成耗时与日志大小成正比（图 11（b））。

3.3　编辑有效性

PSLG 是对现有模型进行日志生成，但日志中迹的执行次数、不同事件发生频次具有一定随机性，对于模型生成日志的修改有以下 3 种方案：

（1）可通过修改输入矩阵即可修改（直接修改了用于日志生成的模型，执行次数、频次仍不可控）；

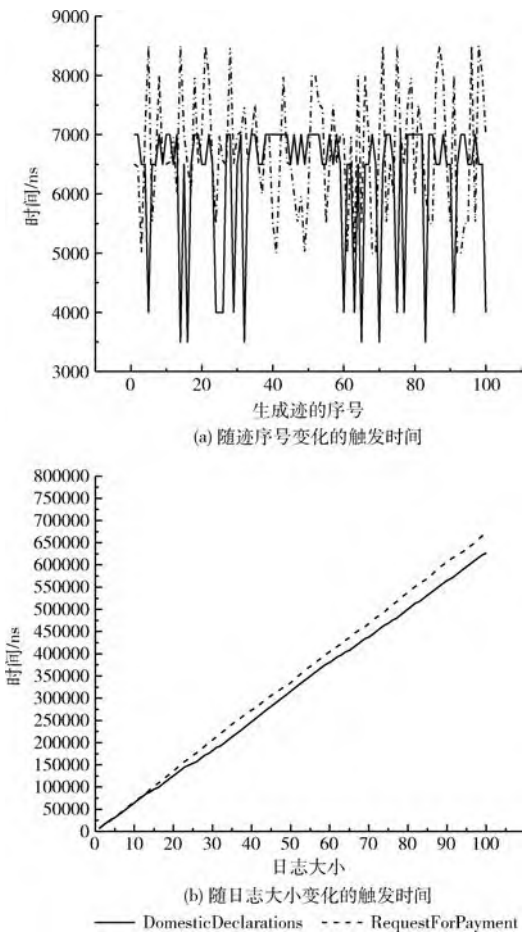


图 11 前 100 条迹组成日志的时间分析

(2) 对于已生成的 XES 日志可通过直接修改 XES 文件(文件结构复杂，且体积一般较大)；

(3) 对于已生成的多重集日志可通过改变序列及执行次数，再利用图 5 (b) 所示插件转为 XES 日志 (不同执行序列是有限的，变迁增删便捷，序列发生次数易修改)。

对于 Domestic Declarations 模型的运行记录 (100 条发生序列)，分别删除变迁 t_6，t_{10}，t_{15}，t_{20}，导入 Prom 利用 Alpha 插件挖掘结果如图 12 所示，利用方法 (3) 调节日志是可行的。

4 结束语

本文提出一种对现有系统生成受控日志的方法，使用增广 Petri 对现有系统进行建模，便于尽可能满足真实系统运行条件。PSLG 利用矩阵输入的方式输入 Petri 网模型，在主界面展示了网的一些属性值用以确认模型输入的正确性；通过随机触发可发生变迁达到下一可达状态并记录触发过程，重复模拟 n 次网的完整运行以生成大小为 n 条的系统运行记录，利用 Hash Map 对系统运行记录进行分类统计生成多重集日志；遍历系统运行记录，通过嵌套循环拼装的方式生成 XES 标准日志，并以此为模板增加了多重集日志转为 XES 标准日志的方法；最后的实验验证了实现插件 PSLG 的可行性、稳定性、日志的有效性及可编辑性。

通过 PSLG 生成日志，为过程挖掘、模型修复等诸多依赖于日志的算法验证及评估提供了有效助力。未来工作包括使用更多建模语言作为输入，丰富日志中的属性，增加日志导出格式，高频行为的可控触发以及基于受控日志的过程挖掘、一致性检验等算法的研究。

参考文献：

[1] Reisig W. Understanding petri nets: Modeling techniques, analysis methods, case studies [M]. Heidelberg: Springer, 2013: 18-27.

[2] VAN DER AALST W. Data science in action [M] //Process mining. Springer, 2016: 3-23.

[3] Teng Y, Du Y, Qi L, et al. A simple logic transition repair method for business process models via logic Petri Nets [J]. IEEE Access, 2019, 7: 76628-76644.

[4] SHAO Chifeng. Parallel optimization algorithm based on process mining [J]. Journal of Chifeng College (Natural Science Edition), 2019, 35 (10): 66-70 (in Chinese). [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.]

[5] SHAO Chifeng. Arithmetic computation of the Petri net model and its implementation [J]. Journal of Chifeng College (Nature Science Edition), 2020, 36 (8): 21-24 (in Chinese). [邵叱风. 算术计算 Petri 网模型及实现 [J]. 赤峰学院学报 (自然科学版), 2020, 36 (8): 21-24.]

[6] XES Working Group. IEEE standard for extensible event stream (XES) for achieving interoperability in event logs and event streams [J]. IEEE Std, 2016, 1849: 1-50.

[7] TAX N, SIDOROVA N, VAN DER AALST WMP, et al. LocalProcessModelDiscovery: Bringing Petri nets to the pattern mining world [C] //International Conference on Applications and Theory of Petri Nets and Concurrency. Springer, Cham, 2018: 374-384.

[8] Burattin A. PLG2: Multiperspective process randomization with online and offline simulations [J]. BPM Demonstration Track, 2016, 169: 1-6.

[9] Jouck T, Depaire B. PTandLogGenerator: A generator for artificial event data [J]. BPM (Demos), 2016, 1789: 23-27.

[10] Jouck T, Depaire B. Generating artificial data for empirical analysis of process discovery algorithms: A process tree and log generator [J]. Business & Information Systems Engineering, 2018, 165: 24-35.

[11] Mitsyuk AA, Shugurov IS, Kalenkova AA, et al. Generating event logs for high-level process models [J]. Simulation Modelling Practice and Theory, 2017, 74 (9): 1-16.

[12] SHAO Chifeng, FANG Xianwen, SHENG Mengjun.

Process mmining algorithm based on enhanced log [J]. Journal of Anhui University of Science and Technology (Natural Science Edition), 2020, 40 (4): 25-32 (in Chinese). [邵叱风，方贤文，盛梦君. 基于增强日志的过程挖掘算法 [J]. 安徽理工大学学报（自然科学版），2020，40（4）：25-32.]

[13] Nguyen HTC, Lee S, Kim J, et al. Autoencoders for improving quality of process event logs [J]. Expert Systems with Applications, 2019, 131 (2): 132-147.

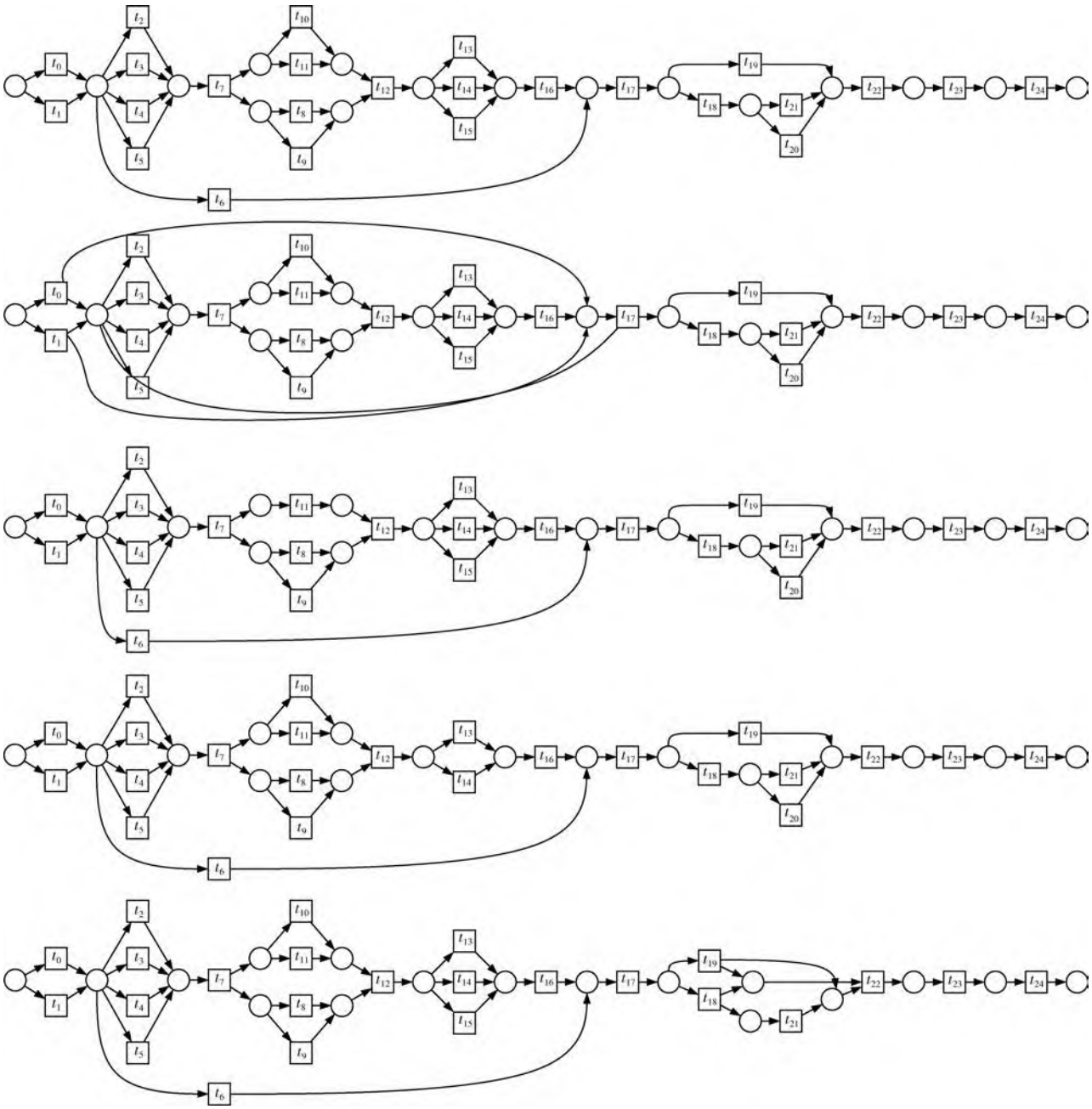


图 12　修改后日志的 Alpha 挖掘结果