第3章 从不完整日志中挖掘非自由选择结构

4. 1. 流程挖掘算法概述

本文设计了并实现了一个可也从不完整日志中挖掘非自由选择结构的流程挖掘算法，并将此挖掘算法命名为。本节从研究动机、基础知识和算法基本流程等三个方面对进行概述。

* + 1. 研究动机

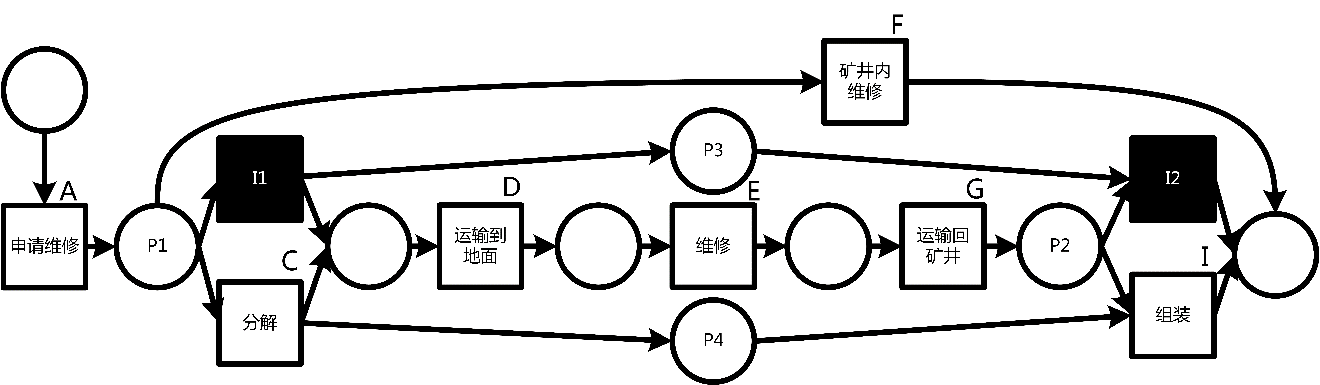


图3.1三一公司一个的业务流程模型

图3.1是展示了三一重工公司的以一个真实事件日志。三一公司是中国最大的装备制造业公司。这个模型描述的是矿井中掘进机维修流程。对于维修，有三个可选的方案。一个方案是掘进机在矿井内维修，通常，如果只是对掘进机进行简单的维修，可是适用于这种解决方案。另外两个解决方案中需要将掘进机从矿井中运输出来进行维修。这两种方案的区别是是否需要将掘进机在运输之前分解，并在运回矿井之后组装。通常，如果矿道较小，不是可整台掘进机运输的时候，需要将掘进机分解以后运输，反之，如果矿道宽度足够掘进机运输，则不需要对其在运输前后分解与组装，节省维修的时间与成本。

图3.1中的流程模型是通过工作流网建模的。在这个流程模型中共有两个不可见任务（I1和I2）,这两个不可见任务的作用是略过“分解”任务和“组装”任务。任务I2与任务“组装”和库所P3,P2和P4构成了一个非自由选择结构。在任务“运输回矿井”被执行以后，执行“组装”任务还是执行不可见任务I2是取决于库所P3和库所P4的状态，而这取决于之前是否执行了 “分解”任务。因此，这个非自由选择结构保证了只有在任务“分解”执行以后才会执行任务“组装”，即不会出现掘进机在维修前没有“分解”但是维修后会“组装”的情况。

为了行文的方便，我们使用每个任务右上角的英文字母来代表每个任务。例如，用A来表示任务“申请维修”。那么对于这个模型,{AF,ADEG,ACDEGI}是其一组日志。尽管这个日志看起来比较简单，但是它对于大部分流程挖掘算法来说都很难正确挖掘的。

* + 1. 基础知识

算法作为系列算法的开拓者，对事件日志中事件间的关系进行了分类与定义，并利用这些事件间的关系来挖掘流程模型。而算法和分别在算法定义的事件关系的基础上定义了描述非自由选择结构和不可见任务的事件关系。而这些定义的关系也是算法的基础，因此本节介绍上述算法中定义的事件关系。

在算法中定义了六种事件间的关系，分别是和。表达了两个任务可以顺序的依次执行。指任务间有直接的因果关系。表示一个可能的循环结构。 表示两个任务之间的并发行为，即两个任务之间可以按照任意顺序来执行。表示两个任务彼此之间没有顺接关系。表示两个任务之间彼此互为关系。例如对于图3.1的流程模型来说，，,,这几条规则是满足的。

**定义3.1** (*算法中定义的关系，虚假依赖，隐式依赖*)令T是一组任务的集合，W是T上的一组事件日志，a和b是T中的两个任务，那么算法中定义的关系，虚假依赖，隐式依赖的定义如下：

* + ,

在算法中，虚假依赖用来描述不可见任务的关系。正如定义3.1中所示，虚假依赖基于六个前提条件。在图3.1所示的流程模型中，是满足的，这是因为有,,,,.表明任务A和D之间应该有一个不可见任务，即图中的不可见任务I1

可达关系是用来描述任务之间的间接地依赖关系。在算法中，可达关系是发现不可见任务的一个必备的条件。例如在图3.1所示的流程模型中，,和是满足的。

* + 1. 算法基本流程

**定义3.2**（*算法流程*）令T是一个任务组成的集合，W是T上的一组事件日志。算法流程的定义如下：

1. 1. 发现改进的虚假依赖

这一步骤的目标是从事件日志中发现不可见任务。大多数不可见任务可以通过应用【alpha$的6】提出的虚假依赖来发现。然而当不可见任务出现在并行结构中的一个分支上时，虚假依赖则无法正确的发现不可见任务。因此在本节，我们提出改进的虚假依赖，可以在不可见任务与并行相结合的情况。

图3.2中的两个模型展示了虚假依赖的缺陷。N1和N2是正常的流程模型，N1’和N2’是算法在利用虚假依赖下挖掘的结果。由于发现虚假依赖不满足，N1’中算法并没有发现流程模型中不可见任务I。不满足的原因是来自另一并行分支任务C的干扰，C的存在使得虚假依赖的必要条件之一不满足。同理，对于N2’,由于任务B的干扰，使得流程模型另一并行分支上错误的发现了一个不可见任务I。这个不可见任务I导致N2’的行为没有N2“丰富”：如事件轨迹ABC在流程模型N2上是合理的事件轨迹，但是对N2’来说就不是合理的。

在定义3.3中定义的*Between-Set*是用来说明描述两个任务见执行的任务集合。当两个任务是平行结构的两个端点时，这两个任务的Between-Set是指并行结构的并行分支上的任务。例如在图3.2中，Between-Set(W1,A,D)={B,C}, Between-Set(W2,A,C)={B,D,E}.

**定义3.3**（*Between-Set*）T是一个任务的集合，W是T上的一组事件日志，a与b是T中的两个任务，是W中的一个事件轨迹，长度为n, .a,b的between-set(即Betweent(W,a,b))定义如下：

定义3.4中定义了改进的虚假依赖关系。我们将【alpha$ 6】中定义和的重新定义为和。相比于之前的定义，和能够消除并行分支对虚假依赖的影响。例如，在图3.2中的N1中，不满足，但是是满足的。在N2中，没有成立，然而是成立的。

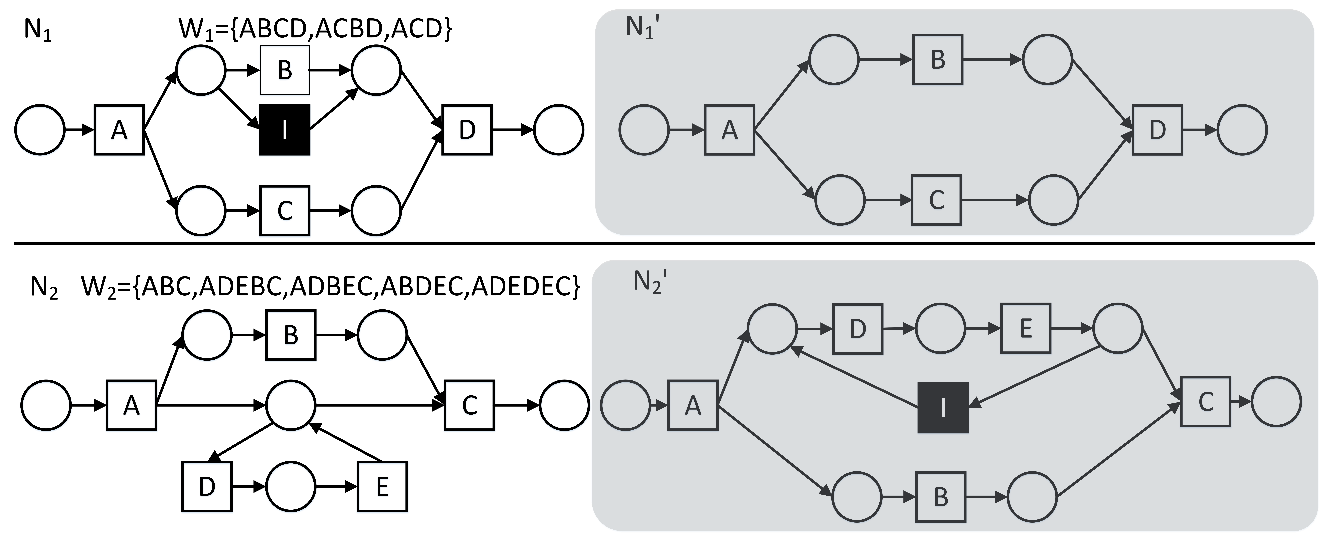
**定义3.4**（*改进的虚假依赖*）T是一个任务的集合，W是T上的一组事件日志，a,x,y,b是T中的四个任务，改进的虚假依赖定义如下：

图3.2两个关于虚假依赖缺陷的流程模型样例

* 1. 补充可达关系

发现非自由选择结构的两个必要条件是发现任务间的长程依赖关系（）和可达关系。这两个关系需要扫描日志才能获得，然而因为不可见任务不出现在事件日志中，因此，需要补充针对不可见任务的可达关系。例如图3.3是一个没有补充可达关系的流程模型的样例，如果没有补充不可见任务的可达关系，那么A到D与B到I的可达关系就没有发现，即由P2组成的非自由选择结构就没有发现。

为了补充与不可见任务关联的可达关系，我们首先定义了条件可达关系(Conditional reachable dependency,简写为CRD)。符号表示在事件轨迹中任务b间接地发生在任务a后。对于事件日志中的每个轨迹，我们人工的添加一个开始任务()和一个结束任务()，即对于一个长度为n的事件日志，有和。

正如定义3.5中定义的，共有三种条件可达关系：pre-CRD(i.e.), post-CRD(i.e.)和both-CRD(i.e.).对于both-CRD中，表示W中存在一个事件轨迹，满足，且a紧接着x发生， y紧接着b发生。例如在图3.3中有。pre-CRD 和post-CRD均是both-CRD的特殊情况。

**定义3.5**(*条件可达关系*) T是一个任务的集合，W是T上的一组事件日志，a,b是T中的两个任务，x,y是中的两个任务，条件可达关系的定义如下：

基于条件可达关系，定义3.6给出了不可见任务相关的可达关系。对于两个不可见任务x,y,成立当且仅当存在四个任务a1,a2,b1和b2满足,,,和.对于一个不可见任务x和一个可见任务m,成立当且仅当存在两个任务a,b,满足,且。例如在图3.3所示的流程模型中，成立，因为与有,和成立。与相类似.

**定义3.6**(*不可见任务相关的可达关系*) T是一个任务的集合，W是T上的一组事件日志，a,b是T中的两个任务，x,y是中的两个任务，条件可达关系的定义如下：

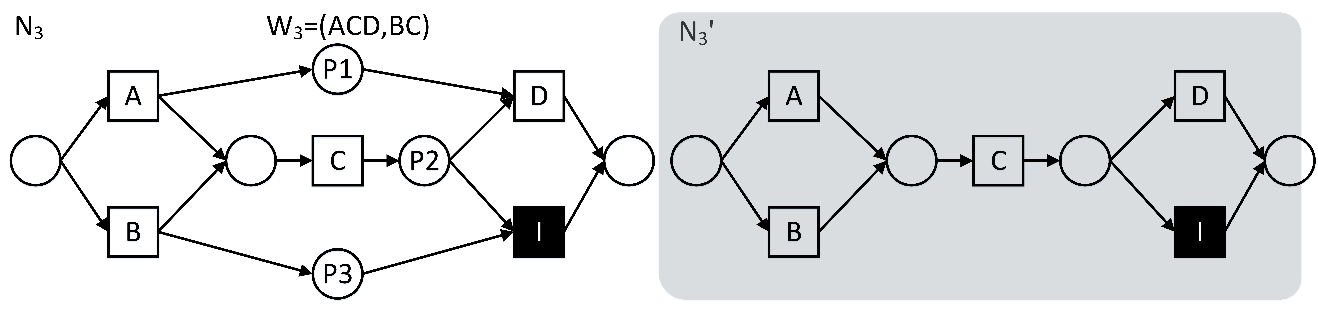


图3.3一个没有补充可达关系的流程模型样例

* 1. 发现非自由选择结构

在上一节完成补充不可见任务的可达关系后，这一节介绍发现不可见任务。算法可以在大多数情况下发现非自由选择结构。然而算法无法挖掘非自由选择结构与短循环（Length-1-Loop,简称L1L）互相嵌套的情况。L1L集合，正如定义3.7中所示，是指一组在事件日志中可能连续出现至少两次的任务集合。算法在发现不可见任务的时候会不考虑所有L1L集合中的任务。例如在图3.4中展示了算法发现L1L任务与非自由选择结构相结合时候的缺陷。N3和N4是两个包含L1L与非自由选择结构结合的流程模型。N3’和N4’是算法挖掘出来的模型。N3’中没有挖掘出非自由选择结构，使得N3’比N3的行为更多，如事件轨迹<ACGFDBE>在N3’中是合理的事件轨迹，但是在N3上是不合理的。另外，在N4’中，算法没有D相关的边，导致N4’不是一个合理的工作流网【alpha$,11】。

**定义3.7**（*短循环集合,L1L*）T是一个任务的集合，W是T上的一组事件日志，短循环集合L1L定义如下：



因此，在发现非自由选择结构之前，我们需要从短循环L1L集合中找到一组任务，这组任务因为涉及到非自由选择结构，因此这组任务不能够存放在L1L中。定义3.8给出这组集合的定义，我们称其为L1L-Free集合。即对于L1L-Free集合中的任何一个任务x,存在两个任务a,b，其中a,b平行且满足a,xb.例如对于N3,L1L-Free集合为{D,H}.对于任务D,有F，D,且F满足。

**定义3.8**(*L1L-Free集合*) T是一个任务的集合，W是T上的一组事件日志，L1L是W上的短循环任务集合。L1L-Free集合的定义如下：

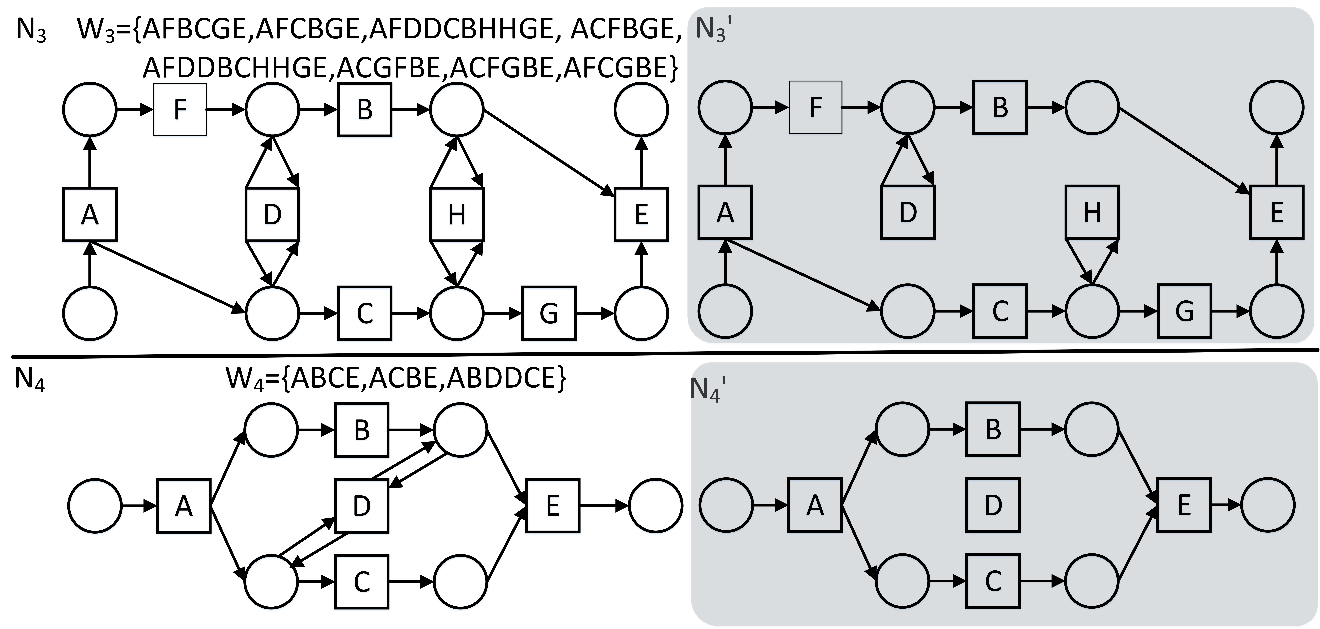
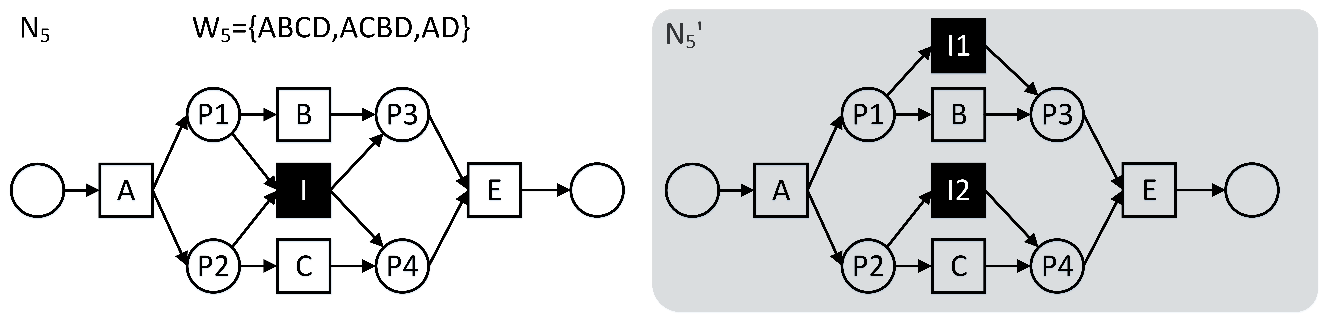


图3.4两个无法发现L1L与非自由结构嵌套的流程模型样例

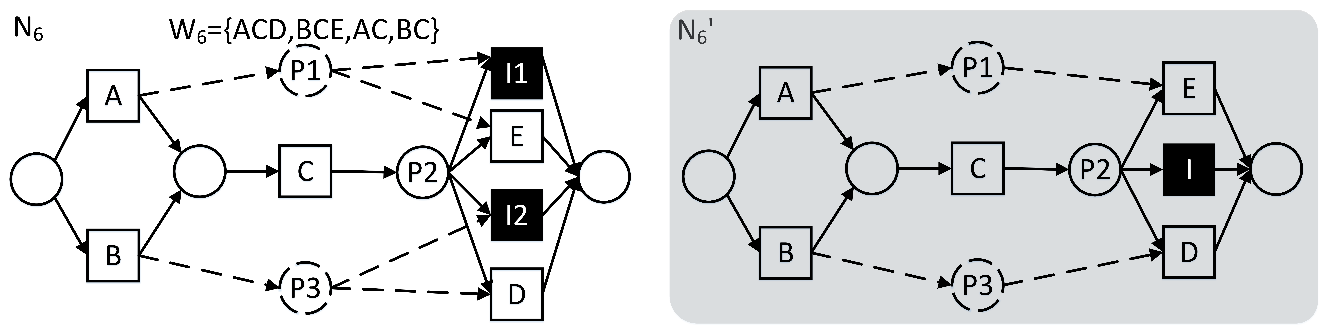
* 1. 调整不可见任务

为了保证构建一个合理与精确的流程模型，在本节需要对一些特定的不可见任务进行分裂和合并。在介绍具体的算法之前，首先及介绍一些辅助函数。

* MD表示改进的虚假依赖的集合;
* ID表示隐式依赖的集合;
* MD(t)表示与任务t相关的改进的虚假依赖的集合;
* 是事件轨迹在任务集合X上的投影。



(a)没有合并处理的挖掘获得的模型



(b)没有分裂处理的挖掘获得的模型

图3.5两个没有合并或分裂处理的挖掘获得的模型

4. 5. 1. 合并不可见任务

当一个流程模型中不同的并行分支上都含有不可见任务时，这些不可见任务有可能会合并成为一个不可见任务。然而算法并不会考虑这个情况。例如图3.5中N­5是一个并行结构（B,C,P1,P2,P3,P4）中含有一个不可见任务I. N­5’是相同日志在算法中挖掘的结果，其中不可见任务I1和I2分别在两个并行分支中。N­5’比N­5所能表达的行为更多，比如说事件轨迹ACE和ABE在N­5’中是合理的事件轨迹，但是对于N­5就不是合理的事件轨迹。我们利用定义3.8中指出的方法来确定哪些不可见任务可以合并。

**定义3.8**（*可以合并的不可见任务集合*）T是一个任务的集合，W是T上的一组事件日志，TI是W中获得的不可见任务的集合，是W的一个事件轨迹。可以合并的不可见任务集合C\_Set定义如下：

R(t)是指与发现不可见任务t相关的任务的集合。是事件轨迹在任务集合上的投影。成立的充要条件是和中的任务在中要么交叠出现，要么都不出现。的含义表示在事件轨迹上不可见任务a和b要么一起出现，要么一个都不出现。如果对于事件日志W中的任何一个事件轨迹均满足，（即），那么说明不可见任务在事件日志W上均在一起出现或者在一起不出现，那么不可见任务a和b应当合并成为一个不可见任务。

* + 1. 分离不可见任务

非自由选择结构是在发现不可见任务以后发现的，然而由于非自由选择结构会给任务间带来更多的依赖关系。这些非自由选择结构有可能在一些情况下给不可见任务带来死锁。在图3.5中，N6和N6’给出了一个这种死锁的样例。N6是原始的合理模型，N6’是没有经过分裂操作的流程模型。在N6’中，不可见任务I比非自由选择结构（P1,P2,P3,E,D）的发现更早一些，库所P1,P3和虚线的边是为了构造非自由选择结构而添加的。新添加的依赖关系(A,E)和（B,D）使得这个网不合理。例如在执行完事件ACI或者BCI以后，会在库所P1或者P2上出现残存的令牌。因此，在本节，对于每一个不可见任务检查，判断这个不可见任务是否需要分裂。

**定义3.9**（*可以分裂的不可见任务集合*）T是一个任务的集合，W是T上的一组事件日志, TI是W中获得的不可见任务的集合。可以分裂的不可见任务集合定义S\_Set如下：

对任何一个不可见任务t,IMD(t)表示与t的虚假依赖相关的隐式依赖（简称其为混合依赖）。根据对流程模型的分析，我们发现每个混合依赖都应该对应一个不可见任务。如果挖掘出来的模型中存在一个不可见任务表达了两个混合依赖的情况，那么这个流程模型将会像图3.5中的N6’一样是不合理的，因此当一个不可见任务表达了多余一个混合依赖时，这个不可见任务应该被分裂。

* 1. 算法的实现

我们将算法实现于软件BeehiveZ。BeehiveZ是由清华大学软件学院信息系统与工程研究所实现开源业务流程与数据分析软件原型系统。BeehiveZ既是一个流程模型数据管理工具，也是一个流程模型数据的应用开发框架。作为流程模型数据管理工具，BeehiveZ中可以导入、导出、存储、可视化展现以及批量生成流程模型。同样的，作为应为开发框架，BeehiveZ提供了进行流程模型相似度度量、流程模型索引、流程挖掘的开发接口。

图3.6 是BeehiveZ的流程挖掘算法模块的用户界面。其中界面中主要功能是集中在“流程挖掘”的标签页中（需要将界面中的“过程挖掘”修改为“流程挖掘”）。在“流程挖掘”标签页中,主要由三个部分组成，分别是“选择日志文件夹”、  
“选择挖掘算法”和“选择输出文件夹”。其中“选择日志文件夹”和“选择输出文件夹”是由两个文件树构成，可以分别用来选择流程挖掘用来输入的事件日志的文件夹和用于输出流程模型的文件夹。中间的“选择流程挖掘算法”标签下列举了一些常用的业务流程挖掘算法，如系列、Genetic、Heuristic等流程挖掘算法。用户分别在“选择日志文件夹”中选择流程挖掘中存放流程挖掘的输入的事件日志的文件夹，然后在“选择输出文件夹”中选择输出的流程模型的文件夹位置，在选择挖掘算法中选择希望使用的流程挖掘算法。最后点击面板左下方的“挖掘”按钮，即可开始流程挖掘。同时在挖掘按钮右侧的进度条会显示挖掘的进度。图3.7是挖掘完成的界面，其中界面中弹出提示框“挖掘完成”，同时可以看到界面中的进度条也展示出了流程挖掘已经完成。

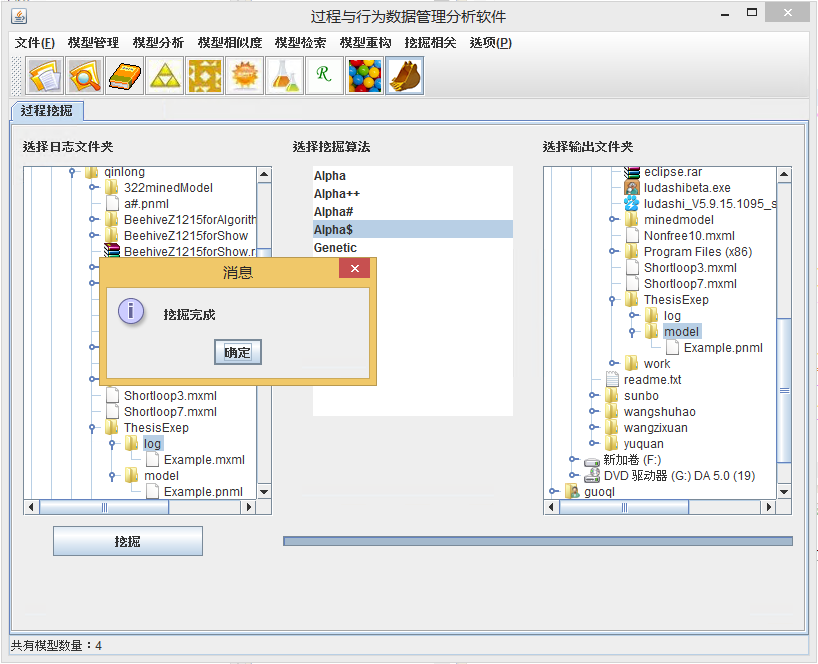


图3.7 流程挖掘结束的界面

我们以事件日志{AF,ACDEGI,ADEG}为例，利用在BeehiveZ中实现的算法挖掘，图3.9展示了流程挖掘的结果。可以看到挖掘了流程模型中一个非自由选择结构和两个不可见任务(图中黑色框所示)，成功的将图3.1中的业务流程模型挖掘出来。

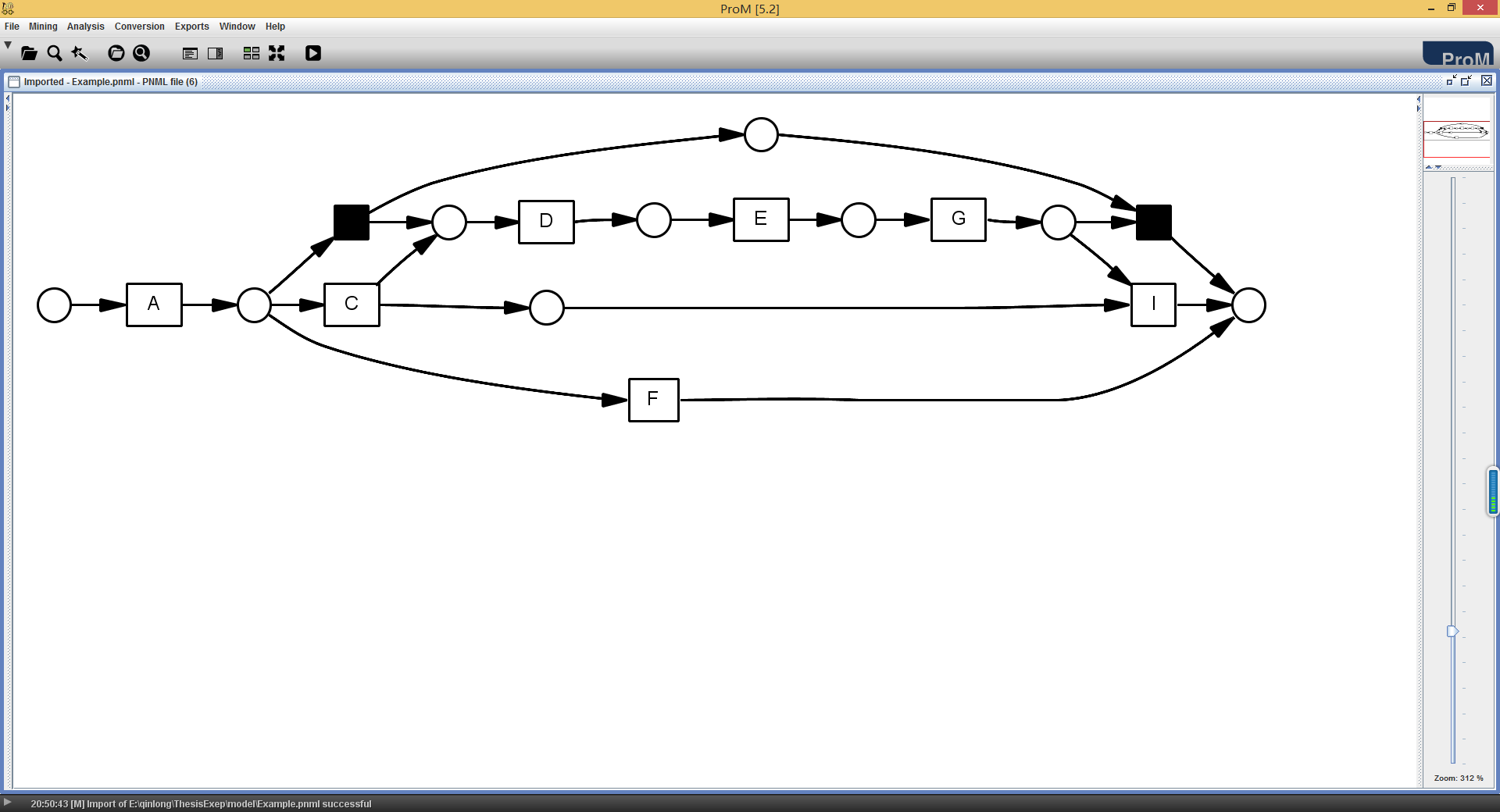


图3.9 流程挖掘的结果

* 1. 实验设计与分析

本节介绍对算法的实验评价。我们将算法与主流的流程挖掘算法，，，Genetic, Heuristic, ILP和Region分别在人工数据和真实数据上做了对比。需要注意的是，Genetic和Heuristic算法的默认输出是Heuristic网，我们使用ProM中的转换插件将Heuristic网转换为Petri网。所有的算法执行的参数都是默认的。

3. 7. 1. 人工数据对比试验

首先，为了说明流程挖掘算法在非自由选择结构和不可见任务上的挖掘能力，首先本文介绍一组流程挖掘算法的对比样例。

表3.1(a) 是一个不可见任务与并行结构嵌套的例子。在正确的参考模型中，两个并行结构共享一个不可见任务。和Genetic成功的挖掘出了正确的流程模型。和Heuristic没有能够挖掘出不可见任务，在每个并行分支上各挖掘出了一个不可见任务，但是没有能够将这两个并行分支合并成为一个不可加任务。ILP算法没有能够挖掘出一个合理的工作流网。而且在表3.1的其他例子中，ILP算法也不能够挖掘出正确的模型。表3.1（b）中的参考正确流程模型中含有一个非自由选择结构与两个不可见任务嵌套的结构，成功的挖掘出了正确的模型，，Genetic和Heuristic算法都没有能够将所有非自由选择结构挖掘出来，这三个流程挖掘算法挖掘出来的模型相比正确的模型就会有更多的行为。算法没有能够挖掘出来不可见任务。表3.1（c）也是一个不可见任务与非自由选择结构嵌套的例子。，Genetic, Heuristic都没有挖掘出不可见任务，而虽然发现了不可见任务，但是没有挖掘出非自由选择结构。表3.1（d）中的正确模型含有一个不可见任务，，和Heuristic没有能够发现不可见任务。而Genetic算法挖掘出来的模型是不合理的。

在人工数据模型集合中，一共有40个流程模型。其中30个流程模型是来自典型流程模型参考库，会在下一章中详细介绍。另外，有5个流程模型(Artif-1)是手工构造的含有非自由选择结构与不可见任务嵌套的流程模型，用来说明的挖掘能力。另外5个人工模型(Artif-2)是利用流程模型与事件日志生成工具Process Log Generator(PLG)【alpha$的21】生成的。

在评估中，我们使用了两个评价标准：

一个评价标准是Fitness,指用来衡量挖掘出来的模型与输入流程挖掘算法的事件日志之间的匹配程度。Fitness的计算是通过将事件日志在流程模型上重现的方式进行的，如果事件日志中的所有事件轨迹都可以在流程模型上重现，那么说明流程模型与事件日志之前的匹配程度很好，即有Fitness = 1。

另一个评价标准是模型重现性（Model Rediscoverability）。在已知原始模型的情况下，模型重现性通过计算原始模型与挖掘所得的模型之间的相似程度来反应流程挖掘算法的挖掘能力。如果挖掘所得的流程模型与原始模型完全一样，那么说明流程挖掘算法可以根据事件日志完全的重构出原始模型，即Model Rediscoverability = 1.

表3.1四个流程模型的例子

（a）

|  |  |  |
| --- | --- | --- |
| 事件日志 | {ABCD7, ACBD9, AB1CD4, ABC1D3, AC1BD6,ACB1D5, AC1B1D3, AB1C1D4, AD11} | |
| 参考模型& |  |  |
| 1a | 1b | C:\Users\GUO Qinlong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1a.png |
| Genetic | Heuristic | ILP |
| 1d | C:\Users\GUO Qinlong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1c.png | C:\Users\GUO Qinlong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1e.png |

（b）

|  |  |  |
| --- | --- | --- |
| 事件日志 | {BCDFG22, ACEFH17, BCFG32, ACFH18} | |
| 参考模型& |  |  |
| 2g | 2b | 2c |
| Genetic | Heuristic | ILP |
| 2c | C:\Users\GUO Qinlong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2d.png | C:\Users\GUO Qinlong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2e.png |

（c）

|  |  |  |
| --- | --- | --- |
| 事件日志 | {ACD22, C23} | |
| 参考模型& |  |  |
| 3g | 3a | 3b |
| Genetic | Heuristic | ILP |
| 3c | 3d | 3e |

(d)

|  |  |  |
| --- | --- | --- |
| 事件日志 | {AEFG, AEBCFG6, AECBFG, AECFBG10,ACEFBG8, ACFEBG6, ACEBFG4} | |
| 参考模型& |  |  |
| 4g | 4a | 4b |
| Genetic | Heuristic | ILP |
| 4c | 4d | 4e |

表3.2实验中的时间开销

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 挖掘算法 |  |  |  | Genetic | Heuristic | ILP | Region |
| 人工日志 | 14.63 | 10.33 | 12.06 | 122.04 | 14.95 | 23.95 | 47.13 |

表3.2展示了各个流程挖掘算法在人工数据集合上的时间开销。的时间开销比和均大一些，小于他们两个的时间开销的总和。但是相比于Genetic算法，的时间开销则要小很多

* + 1. 真实数据对比试验

本节介绍针对真实数据的实验。首先本节举例给出两个真实实例的流程模型，分析与其他主流流程挖掘算法在真实实例上的表现，其次针对一组流程模型分析。

**模型1**

模型1来自2013年度的业务流程智能挑战（Business Process Intelligence Challenge，BPIC 2013）中提供的真实事件日志。这个大型事件日志是来自比利时沃尔沃信息系统部门的事件与问题管理系统。在这个事件日志中，共有7554个事件轨迹和80641个事件。

因为在这个事件日志中不存在已有的流程模型，因此模型重现性(model rediscoverability)并不能作为这个事件日志的评价标准。在这里例子中，我们额外的使用了两个评价标准:准确度(precision):评价流程模型的行为与日志表达的行为之间的相似程度。简洁度(simplicity):通过统计模型的节点(库所、变迁)个数评价模型的简洁程度。另外我们判断每个流程模型是否是工作流网，并记录了每个挖掘算法的挖掘时间。表3.3是这个各个挖掘算法在流程日志的挖掘结果，其中，Heuristic，ILP和Region算法均没有挖掘出一个工作流网。图3.10是挖掘出来的流程模型，可以看到挖掘出来的流程模型中并不包含非自由选择结构，而具有一些不可见任务。因此，算法与算法挖掘出了相同的流程模型，然而因为算法中考虑的流程关系较多，因此时间开销较大。算法没法挖掘出来流程模型中的不可见任务，因此在Precision和Fitness中略逊于。Genetic算法与相比，在Precision、Simplicity和Fitness方面效果相似，然而时间上快了两个数量级。

表3.3模型1的实验结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | Genetic | Heuristic | ILP | Region |
| Fitness | 1.00 | 0.88 | 1.00 | 0.96 | 0.91 | 1.00 | 0.81 |
| Precision | 0.67 | 0.42 | 0.67 | 0.68 | 0.62 | 0.07 | 0.11 |
| Simplicity | 44 | 22 | 44 | 50 | 50 | 14 | 22 |
| Workflow Net | 是 | 否 | 是 | 是 | 否 | 否 | 否 |
| Time(ms) | 4172 | 639 | 456 | 2308595 | 563 | 506 | 22820 |

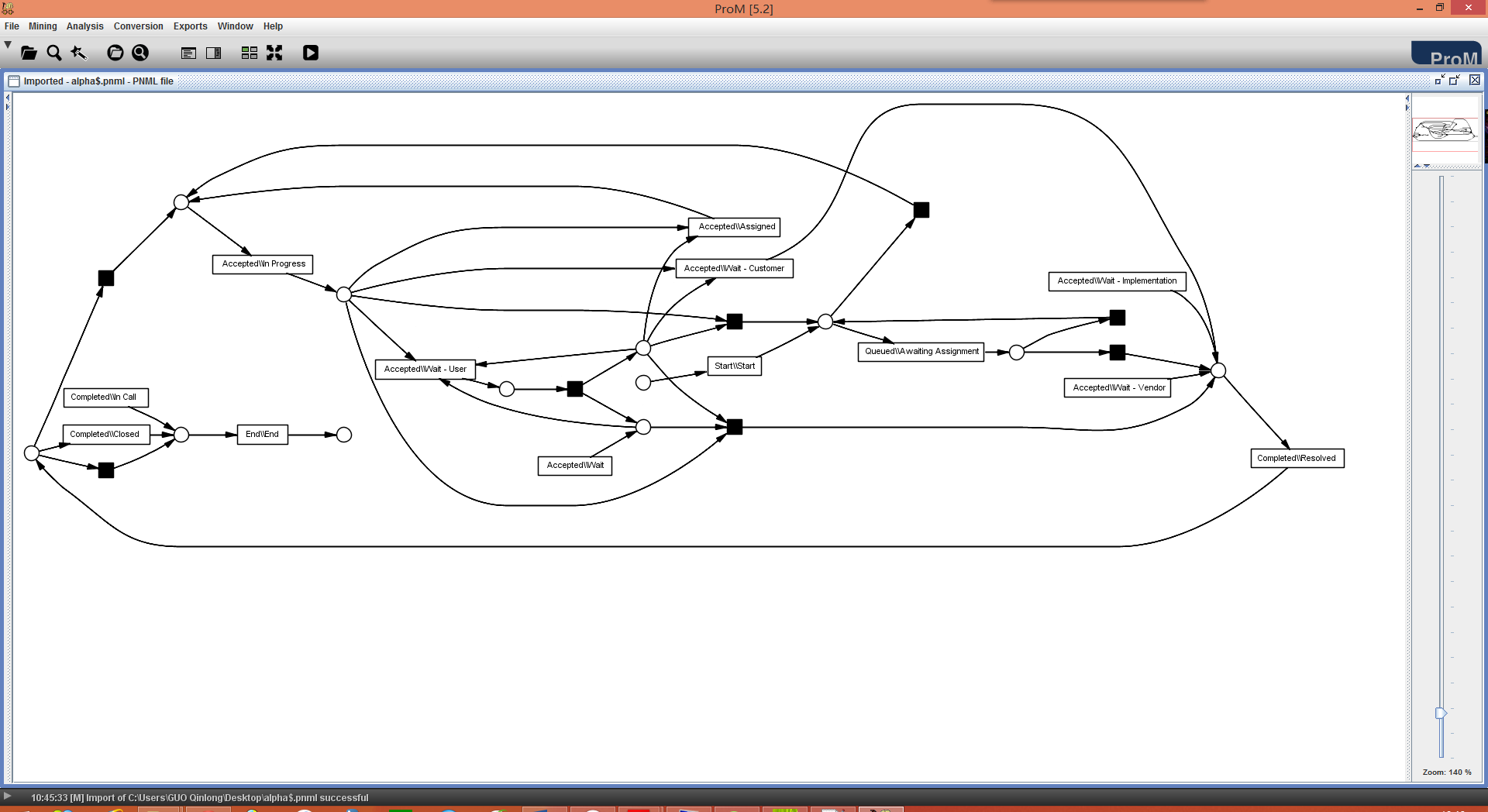


图3.10 在模型1 上的挖掘结果

**模型2**

模型2是一个参加驾校学习驾驶汽车或者摩托车的流程模型，来自学术论文【Genetic】。在这个流程模型中，同时存在不可见任务和非自由选择结构，且非自由选择结构与不可见任务是嵌套在一起的。

表3.4展示了各个流程挖掘算法的在模型2上的挖掘结果，可以看出在fitness,precision和simplicity方面，均比其他挖掘算法效果好。而时间开销上约为和的和，远小于Genetic算法。图3.11展示了算法的挖掘结果。

表3.4模型2的实验结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | Genetic | Heuristic | ILP | Region |
| Fitness | 1 | 0.9 | 1 | 1 | 1 | 1 | 0.90 |
| Precision | 1 | 0.9 | 0.89 | 0.92 | 0.89 | 0.47 | 0.80 |
| Simplicity | 22 | 20 | 20 | 23 | 20 | 16 | 24 |
| Workflow Net | True | True | True | True | True | False | False |
| Time(ms) | 187 | 97 | 97 | 24783 | 25 | 247 | 1316 |

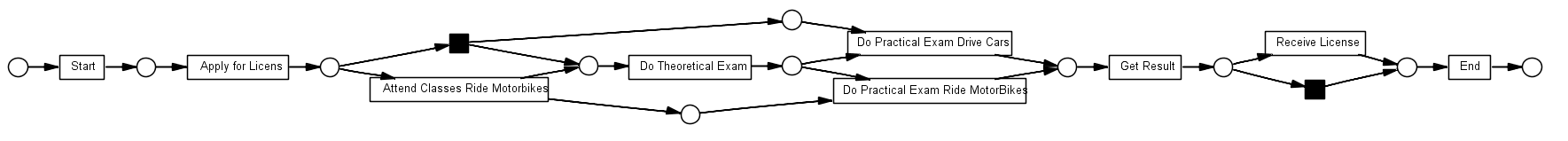
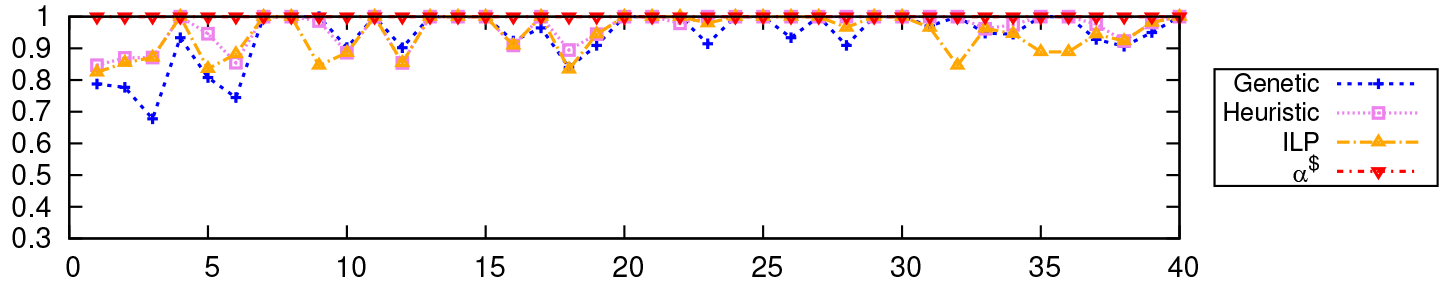


图3.11 在模型2上的挖掘结果

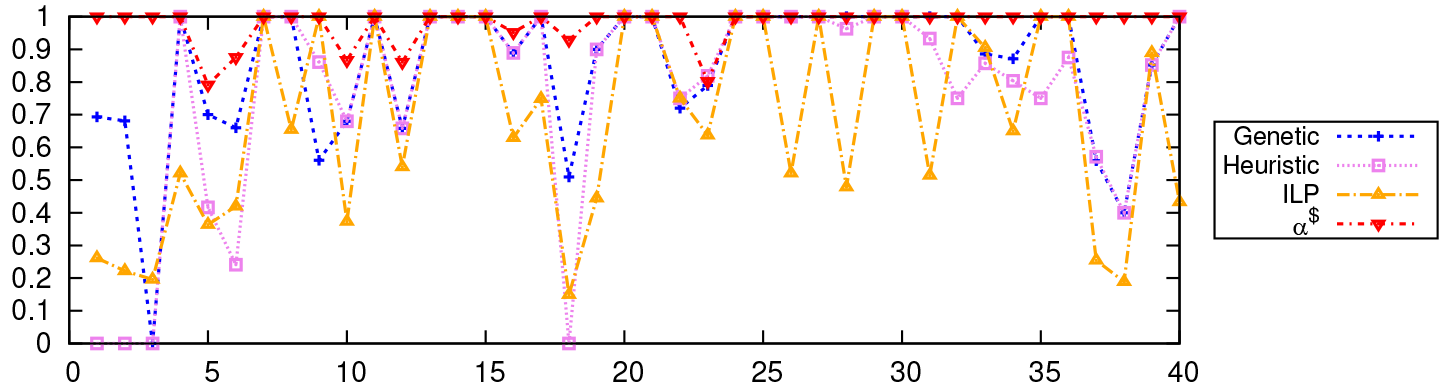
**一组真实数据实验**

真实数据是一组来自东方锅炉厂和唐山车辆制造厂的四十个真实业务流程模型,因为这组真实数据是含有参考模型，因为如同评价手工模型，在评价这组真实数据时依然采用了fitness和model rediscoverability两条评价标准。

图3.12(a)和(b)分别展示了在这组真实数据集上fitness和model rediscoverability的结果。图3.12(a)表明在真实数据中表现十分良好，所有的模型的fitness均为1，而其他算法的fitness均不为1。图3.12(b)表明在model rediscoverability上表现超过其他的所有算法。



(a)Fitness



(b)Rediscoverability

图3.12真实数据的实验结果

* 1. 一个真实的例子

在本节中，我们以图3.1的流程模型为例，分步骤给出挖掘出流程模型的流程。

表3.5给出了一个在真实场景中事件日志的样例。表中分为四列，其中“事件轨迹编号”表示每个事件所述的事件轨迹的编号，“事件标号”代表事件所对应的唯一的编号。“时间戳”和“活动”分别是事件的两个属性，其中“时间戳”表示事件发生的时间，“活动”表示事件所代表的具体活动中为了表述方便，我们对于活动使用大写字母作为简写。表中的每一行代表一个具体的事件。

表3.5事件日志的样例

|  |  |  |  |
| --- | --- | --- | --- |
| 事件轨迹编号 | 事件编号 | 属性 | |
| 时间戳 | 活动 |
| 1 | 35654422 | 30-12-2010:11.02 | A |
|  | 35654423 | 31-12-2010:13.32 | F |
| 2 | 35654481 | 01-01-2011:09.02 | A |
|  | 35654483 | 01-01-2011:13.24 | D |
|  | 35654484 | 02-01-2011:10.22 | E |
|  | 35654485 | 02-01-2011:21.22 | G |
| 3 | 35654579 | 03-02-2011:9.12 | A |
|  | 35654580 | 03-02-2011:10.23 | C |
|  | 35654581 | 03-02-2011:11.53 | D |
|  | 35654582 | 03-02-2011:12.23 | E |
|  | 35654583 | 03-02-2011:14.04 | G |
|  | 35654584 | 03-02-2011:17.04 | I |
| 4 | 35655442 | 30-12-2010:11.02 | A |
|  | 35655443 | 31-12-2010:13.32 | F |

为了方便表达，表3.5的事件日志可以简写为W={AF2,ADEG,ACDEGI},其中上标2表示事件轨迹AF在事件日志中出现了两次。所有的任务的集合为T={A,C,D,E,G,I,F}

第一步是计算事件日志W中的所有事件:

表示事件日志W中出现的所有事件，即事件ACDEGIF是事件日志W中出现的所有的事件日志。

第二步是计算所有处在短循环中的事件:

表示事件日志W中所有的处于短循环中的日志，但是在本例中，因为没有任务是处于短循环中的，因此是空集。

同理，第三步和第四步分别计算同时处于短循环与隐式依赖关系的事件集合和仅处于短循环中的事件集合，由于本例中没有处于短循环中的事件，因此集合和集合均为空集，即有和。

第五步是将事件日志中所有短循环事件从事件日志W中剔除，构造剔除以后的事件日志.

因为本例中不含有短循环的事件，因此出去了短循环任务后的事件日志与原始的事件日志是相同的，即有。

第六步是利用改进的算法挖掘事件日志，这里的改进是指3.2节针对虚假依赖与并行结构相互干扰的情形。

其中有

表示挖掘出的流程模型中的库所，挖掘出来的模型中的变迁，表示挖掘出来的流程模型中的不可见任务构成的变迁，表示挖掘出来的模型中的边。图3.13为在第六步中挖掘所得的流程模型，可以看到改进的算法已经挖掘出一个流程模型。但是这个模型中存在行为较原始日志W多的问题，除了原始日志W中含有的四条事件轨迹外，事件轨迹ACDEG和ADEGI均是这个模型的合理事件轨迹。

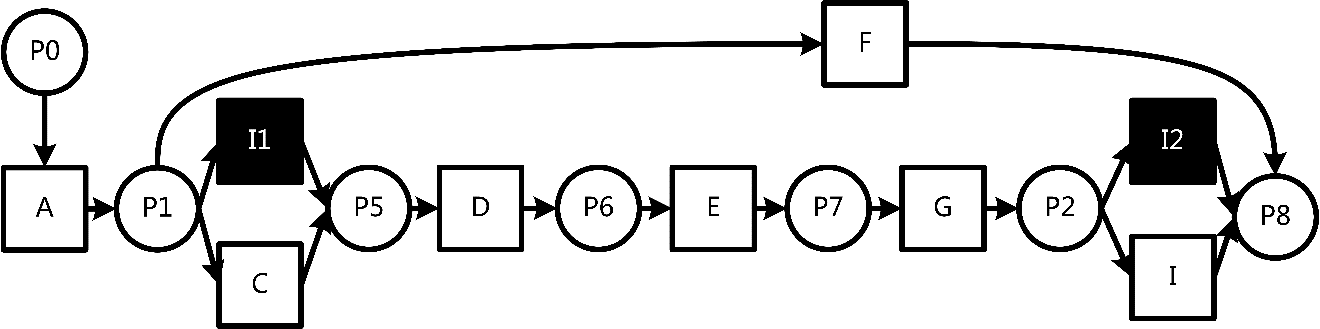


图3.13 第六步挖掘获得的流程模型

第七步是计算与不可见任务相关的可达关系。

表示不可见任务相关的可达关系。在本例中，有两个不可见任务I1和I2，通过第3.3节介绍的补充可达关系的方法，可以计算出与不可见任务I1和I2相关的可达关系。

第八步是计算隐式依赖

表示隐式依赖，即对应于流程模型中的非自由选择结构，利用提供的方法可以计算出流程模型中存在的隐式依赖.

第九步是将上一步发现的隐式依赖添加到现有的流程模型中，添加隐式依赖时需要添加对应库所之间的边，同时因为库所之间不能直接相连，必要时需要添加一些库所。

其中新添加的库所和边如下:

第十步是将L1L中的事件日志补充进入流程模型中，即

但是因为本例中不含有L1L循环，因此这个步骤对流程模型不产生影响。

最后，获得的流程模型即为实际获得的流程模型。

其中有

图3.14展示了最后获得的流程模型，其中包含一个不可见任务与非自由选择结构嵌套的结构，即图中的I1,I2为不可见任务，而P3,P4,P2,I2和I以及对应的边构成了一个非自由选择结构。

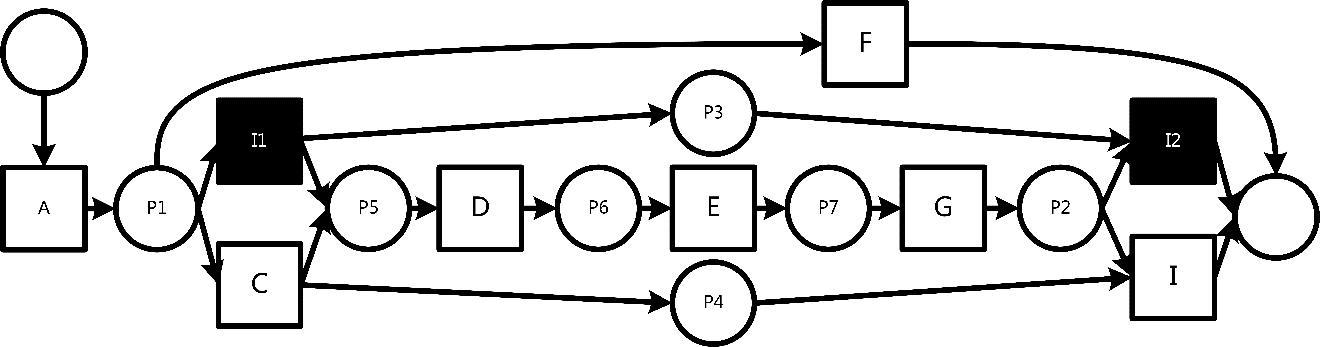


图3.14 最终挖掘获得的流程模型

* 1. 本章小结

本章介绍了面向复杂结构的流程挖掘算法设计与实现。为了方便表述，这个流程挖掘算法命名为。算法可以挖掘出非自由选择结构与不可见任务嵌套的流程模型。本章介绍了算法的设计背景，并且通过一个真实的例子详细介绍了算法的步骤。最后，分别在手工模型和真实数据模型上对和一些主流的流程挖掘算法做了对比，实验证明算法能够按预期挖掘出非自由选择结构与不可见任务嵌套的流程模型，并且性能较当今主流的流程挖掘算法有提升。