一种新的工作流频繁闭合模式挖掘算法研究

王丹丹 蒋文娟
(南通大学计算机科学与技术学院 南通 226019)

摘 要 为了提高工作流环境下频繁模式挖掘的准确性,提出了一种新的频繁闭合模式挖掘算法。首先扩展了依赖矩阵的定义,即利用工作流日志建立包含直接依赖关系和交叠关系的依赖支持度矩阵。然后扩展了 CHARM 算法,以在支持度矩阵的基础上自动挖掘频繁闭合活动集。最后对频繁闭合项集进行处理,以形成最终的工作流频繁闭合模式。该算法对于并行和选择关系的处理能力优于同类算法。

关键词 工作流,频繁闭合模式,依赖矩阵
中图法分类号 TP311 **文献标识码** A

New Algorithm for Mining Workflow Frequent Closet Pattern

WANG Dan-dan JIANG Wen-juan
(Computer Science & Technology Institute, Nantong University, Nantong 226019, China)

Abstract To improve the accuracy of workflow frequent patterns, we proposed a new algorithm for mining frequent closed pattern. First, we extended the definition of the dependence Matrix, which includes direct dependency and overlapping relationship among activities from the work flow logs. Second, we extended the CHARM algorithm to mine all closed frequent patterns over the dependence matrix. Finally, the workflow frequent patterns were generated by discarding non-frequent patterns. The algorithm has advantages in dealing with the interleaving relations among activities and workflow models with both serial or parallel relations.

Keywords Workflow, Closed frequent patterns, Dependence matrix

1 引言

自 Agrawal 等人最早提出利用工作流系统日志挖掘工作流模型以来,工作流挖掘研究得到越来越多学者的关注[1]。工作流挖掘研究中主要有两个重要问题:工作流模型重构和工作流频繁模式挖掘。工作流模型重构主要通过研究企业已有的日志记录来重构工作流模型。在一系列的工作流模型重构算法中,Alast 和他的研究小组提出的 a 算法[2]最为著名。工作流频繁模式挖掘则是工作流挖掘领域中的另一个重要问题,是展现活动间逻辑关系强弱的重要工具。工作流频繁模式挖掘的意义在于:(1)通过发现工作流中经常出现的活动以及活动之间的依赖关系,帮助管理者及时了解工作流的变化情况,并对企业流程进行监控;(2)通过识别流程中的关键活动,定位决策点;(3)在关键决策点提供对后续活动的商业预测,包括走势预测与风险预测。

目前,学者们在工作流频繁模式挖掘领域开展的研究甚少。文献[3,4]提出了序列模式挖掘方法,该方法利用 Apriori 算法挖掘工作流实例中大于预定义阈值的最大序列作为工作流频繁模式。文献[5]利用窗口原理找到序列集合。文献[6]首次提出了工作流频繁模式挖掘的概念,并使用序列挖掘技术分析日志记录中蕴含的频繁活动执行序列。该工作将每个活动作为原子事件,而没有考虑活动的执行时间间隔,所以只能处理顺序的工作流模式。文献[7]提出了一种新的工作流频繁模式挖掘算法。该算法将活动间的依赖关系作为"项";然后构建依赖矩阵;最后通过依赖矩阵挖掘频繁项集,得到最终的工作流频繁模式。然而该算法未考虑活动之间存在的重叠关系,故不能完全处理具有并行关系的工作流模型。另一方面,该算法以 Apriori 算法为原型进行扩展,因此执行效率低,很难处理大规模数据。

本文在前人工作的基础上提出一种新的用于工作流环境下的频繁闭合模式挖掘方法,主要工作有:首先,扩展了 Mohammed J 等人的基于传统事务型数据库的频繁闭合模式挖掘算法 CHARM[8],使之适合工作流环境下的基于有向图表示的频繁活动模式挖掘。其次,使用活动间的直接依赖关系和交叠关系建立活动关系矩阵。通过对活动间各种依赖关系的挖掘得到最终的频繁闭合活动模式。最后,本文在同一实例上与文献[7]的工作进行了对比。实验表明,本文的算法在处理活动间的并行关系时更具优越性,且效率更高。

2 基本定义

2.1 日志模型

系统日志记录了系统实际的工作流执行情况。工作流建

到稿日期:2011-12-09 返修日期:2012-06-05 本文受江苏省高校自然科学基础研究项目(10KJB510022),南通市应用研究计划项目(BK2012035)资助。

王丹丹(1979—),女,硕士,讲师,主要研究方向为数据挖掘与知识发现,E-mail:wang.dd@ntu.edu.cn;蒋文娟(1975—),女,博士,主要研究方向为网络控制系统、时滞系统。

模时定义了流程需要执行的多个任务。任务的一次具体执行叫做活动。活动是工作流模型的基本组成单元。

定义 1(活动,Activity) 设 $Task$ 是所有任务的有限非空集合,$Time$ 为时间域。活动集合 $A \subseteq Task \times Time \times Time$,$A=\{a \mid a=(task,t_s,t_e), task \in Task, t_s \in Time, t_e \in Time, t_s<t_e\}$。其中,$task$ 表示该活动对应的任务,t_s 表示活动开始时间,t_e 表示活动结束时间,且 $t_s<t_e$。元组 a 的 3 个分量分别用 $a.task, a.t_s, a.t_e$ 相应表示。

定义 2(工作流轨迹,Workflow Trace) 用 σ 表示一个工作流执行的轨迹。本文假设工作流中不存在循环,则任何工作流轨迹中不存在重复活动。σ 记录了工作流程一次执行中的各活动的执行时间。$\sigma=a_1,a_2,\cdots a_m \in A^+$,对于 $1 \leqslant i<j \leqslant m$,有 $a_i.t_s<a_j.t_s$。另外,$|\sigma|=m$ 表示该工作流轨迹中执行的活动个数,$AS(\sigma)$ 表示工作流执行轨迹中的活动集合。$W \subseteq A^+$ 表示工作流日志,$\sigma^{(i)}$ 表示日志 W 中的第 i 条工作流运行轨迹。

2.2 活动依赖关系

定义 3(优先) 活动 a_i 优先于活动 a_j,如果 $\exists \sigma \in W$,对于 $1 \leqslant i \leqslant j \leqslant |\sigma|$,$a_i.t_e<a_j.t_s$,且 $\neg \exists a_k \in \sigma, a_i.t_e<a_k.t_s<a_k.t_e<a_j.t_s$。

定义 4(直接依赖) 如果 $\forall \sigma \in W$,都有活动 a_i 优先于活动 a_j,则称活动 a_j 直接依赖于活动 a_i。

由定义 4 可知,只有当活动 a_i 对 a_j 的优先关系在整个日志集上都成立时,其直接依赖关系才是正确的。这时,在有向图表示的工作流模型中存在一条有向边从 a_i 指向 a_j,记为 $a_i \rightarrow a_j$。

定义 5(重叠) 活动 a_i 和 a_j 具有重叠关系,如果 $\exists \sigma \in W, a_i, a_j \in \sigma$,且 $a_i.t_s \leqslant a_j.t_s \leqslant a_i.t_e \vee a_j.t_s \leqslant a_i.t_s \leqslant a_j.t_e$。

定义 6(并行) 活动 a_i 和 a_j 之间存在并行模式,当且仅当 $\exists \sigma^{(i)} \in W, a_i \in \sigma^{(i)}, a_j \in \sigma^{(i)}$,活动 a_i 优先于活动 a_j,且 $\exists \sigma^{(j)} \in W, a_i \in \sigma^{(j)}, a_j \in \sigma^{(j)}$,活动 a_j 优先于活动 a_i,或者 $\exists \sigma^{(k)} \in W$,活动 a_i 和 a_j 具有重叠关系,记为 $a_i \| a_j$。

不考虑活动间并行关系开始的时间,显然 $a_i \| a_j \Leftrightarrow a_j \| a_i$。

2.3 工作流频繁模式

定义 7(活动模式,Pattern) 模式 $P=(V,F)$ 为一个有向图,若 $\exists \sigma \in W$,使得 P 为 σ 所对应的有向图的子图,则称 σ 支持一个有向图表示的工作流模式 P,记为 $P \Uparrow \sigma$。

定义 7 用图的形式表达一个工作流模式。所以,在工作流环境下挖掘频繁活动模式的过程等价于在工作流图中寻找频繁子图的过程。下面给出频繁活动模式的定义。

定义 8(频繁活动模式,Frequent Pattern) W 为工作流日志中所有的工作流运行轨迹集合,min_sup 为区间[0,1]之间的一个实数。模式 P 是频繁的,当且仅当 $\sup(P) \geqslant$ min_sup,其中,模式 P 的支持度 $\sup(P)=|\{\sigma:\sigma \in W, P \Uparrow \sigma\}| / |W| \times 100\%$。

文献[9]提出了频繁闭合模式的概念,同时证明了挖掘频繁模式的完全集合与挖掘频繁闭合模式的等价性。频繁闭合模式可以提高挖掘效率,缩减挖掘时间,降低系统开销。本文挖掘工作流环境下的频繁闭合活动模式。

定义 9(频繁闭合活动模式,Closed Frequent Pattern) 若一个模式 P' 是频繁闭合活动模式,那么一定不存在这样的一个模式 P 使得(1)P' 是 P 的子图;(2)任何一个支持模式 P' 的执行轨迹也支持模式 P;并且 $\sup(P') \geqslant$ min_sup。

3 算法挖掘基本思想

工作流环境下各活动之间并不是简单的原子或顺序关系,而是呈现诸如顺序、选择、并行等多种逻辑关系。文献[7]中,依赖矩阵是整个算法的核心,包含了活动集中任意活动间的依赖关系。一方面,由依赖关系的传递性可知,部分活动间的依赖关系可通过其他活动间的依赖关系进行传递表达,故无需在矩阵模型中包含所有活动间的依赖关系。另一方面,文献[7]中的依赖矩阵无法显式地表达活动之间的交叠关系,导致算法无法完全处理具有并行结构的工作流模型。本文在此基础上进行了两方面的改进:(1)舍去所有可以从直接依赖关系导出的活动依赖关系。(2)将活动间的交叠关系合并入矩阵。修改之后包含直接依赖与重叠关系的矩阵简记为 DO 矩阵。(3)矩阵中元素的值表示活动 a_i 和 a_j 之间直接依赖或交叠关系的支持度计数。为了区别活动之间的这两种关系,直接依赖关系的支持度计数用正整数表示,交叠关系的支持度计数用负整数表示。下面首先给出 DO 矩阵的定义,然后证明该矩阵模型表示的依赖关系是正确的。

定义 10(直接依赖和重叠关系矩阵) $\forall \sigma^{(i)}, \sigma^{(j)} \in W_i$,$W_i \subseteq W$,$W_i$ 为具有相同活动集的工作流运行轨迹集合,即 $AS(\sigma^{(i)})=AS(\sigma^{(j)})$,则 W_i 对应的 n 阶直接依赖和重叠关系矩阵为 $DO=(d_{ij})$,$n=|\sigma^{(i)}|$。其中,if $a_i \rightarrow a_j$,则 $d_{ij}=|\sigma^{(k)} \in W_i(a_i \rightarrow a_j) \Uparrow \sigma^{(k)}|$;else if $a_i \| a_j$,则 $d_{ij}=-|\sigma^{(k)} \in W_i(a_i \| a_j) \Uparrow \sigma^{(k)}|$;else $d_{ij}=0$。

证明:DO 矩阵模型是正确的,即证明该模型可以正确表示活动之间的顺序、并行和选择关系。由定义 10 易知,完备日志集 W 上的活动关系模型将由多个 DO 矩阵模型共同表示。由于具有不同活动集的工作流轨迹之间可能蕴涵逻辑或的关系,因此逻辑或关系必也蕴涵在多个不同的 DO 矩阵之间。定义 6 保证了活动间并行关系的发现。下面证明模型所表示的顺序依赖关系的完备性。

定理 1 $\forall a_i, a_j$,若 $a_i \pi_w a_j$[7],则一定存在一个活动序列 $a_{k_1}, a_{k_2} \cdots a_{k_m}$,使得,$a_i \rightarrow a_{k_1}, a_{k_1} \rightarrow a_{k_2}, \cdots a_{k_m} \rightarrow a_j$。

证明:$\because a_i \pi_w a_j$,$\therefore \forall \sigma^{(i)} \in W$,都有 $a_i.t_e<a_j.t_s$。故根据定义 5 易知,活动 a_i 和活动 a_j 不可能存在交叠关系,又 $\because a_i$ 和 a_j 出现在同一执行轨迹中,故 a_i 和 a_j 之间不可能存在选择关系。易知 a_i 和 a_j 必为顺序执行关系。$\therefore$ 对于有向图表示的工作流模型,活动 a_i 到活动 a_j 之间必存在一条经过若干活动序列的通路 $a_i, a_{k_1}, a_{k_2} \cdots a_{k_m}, a_j (m \geqslant 0)$。

又 $\because DO$ 矩阵中包含了所有活动间的直接依赖关系,故 $a_i \rightarrow a_{k_1}, a_{k_1} \rightarrow a_{k_2}, \cdots, a_{k_m} \rightarrow a_j$,定理 1 得证。

4 频繁模式挖掘过程

4.1 数据预处理

工作流日志中存在大量重复的工作流运行轨迹。为了减少后续挖掘过程中处理的数据量,提高挖掘的效率,数据预处理的第一步就是过滤掉这些重复出现的运行轨迹。由于算法关心的是执行轨迹中的每个活动的开始事件和结束事件发生的先后顺序而不是具体的开始和结束事件,因此,数据预处理的第二步是将过滤后的日志中的每一条执行轨迹表示为以下

形式：

$$\sigma^{(i)}=\{a_1.t_s,a_2.t_s,a_1.t_e,a_3.t_s\cdots,a_i.t_s,a_j.t_s,a_i.t_e,\cdots,a_m.t_s,\cdots,a_m.t_e\}$$

第三步需要将第二步得到的工作流轨迹集 W_0 划分为多个子集 $W_1,W_2,\cdots,W_m$，以满足 $\forall W_i$ 中如果有 $\sigma^{(i)}\in W_i$，则 $\forall\sigma^{(i)}\in W_i$，都有 $AS(\sigma^{(i)})=AS(\sigma^{(j)})$，且 $W_0=W_1\cup W_2\cup\cdots\cup W_m$。后续挖掘过程分别为每个 W_i 建立相应的 DO 矩阵，得 DO 矩阵列表 $DO=\{DO_1,DO_2,\cdots,DO_m\}$。

4.2 频繁闭合活动集挖掘

本文采用扩展的 CHARM 算法挖掘频繁活动集。与类 Apriori 算法不同的是，CHARM 算法采用的是纵向数据表示形式，也就是说，每一个项和一组标识集相关。在工作流环境下，标识集指的是包含该项的工作流运行轨迹编号的集合，算法在 IM-Tree(Itemset-Midset Search Tree)上挖掘所有的频繁闭合活动集，IM-Tree 上的每一个节点表示为二元组 $X\times M(X)$。其中，X 为项的集合，$M(X)$ 则为包含该项集 X 的工作流运行轨迹编号的集合。详细的 CHARM 算法原理可参考文献[8]。

扩展后的 CHARM 算法以 DO 矩阵为模型进行频繁活动模式挖掘，算法采用活动间的优先和重叠关系作为 1-项集。即

$$1\text{-}itemset=\{d_{ij}\mid d_{ij}>0,d_{ij}\in DO_{k=1\cdots t},i<j\}\cup\{p_{mn}\mid p_{mn}<0,p_{mn}\in DO_{k=1\cdots t},m<n\}$$

根据定义 8 计算每个项的支持度，大于最小支持度阈值的项即为频繁 1-项集。将频繁 1-项集按照包含该项的 2-项集支持度之和升序排列，构成 $f-list$，$f-list$ 中的每一项均表示成 $X\times M(X)$ 二元组的形式。函数 *CHARM-Extended* 用来发现所有的频繁闭合活动集。

函数：*CHARM-Extened*([P],C)

1)参数说明：

[P]={$X_i\times M(X_i):X_i\in 1\text{-}itemset\wedge count(X_i)\geqslant$min_sup}

C：用来存放挖掘出的频繁闭合活动集，初始情况下 $C=\phi$。

2)函数步骤：

```
for each Xi×M(Xi)in [P]
  [Pi]=ϕ and X=Xi
  for eachXj×M(Xj) in [P],with Xj≥fXi
    X=X∪Xjand Y=M(Xi)∩M[Xj]
    for each DOi in DO
      if (there exits element values ≠ 0)=elements in X then
      count(X)+=element value
    if count(X)≥min_sup then
      if M(Xi)=M(Xj) then
        remove Xj from [P]
        replace all Xi with X
      else if M(Xi)⊂M(Xj) then
        replace all Xi with X
      else if M(Xi)⊃M(Xj) then
        remove Xj from[P]
        add X×Y to [Pi]
      else if M(Xi)≠M(Xj) then
        add X×Y to [Pi]
    if [Pi]≠ϕ then CHARM-Extened([Pi],C)
    delete [Pi]
    C=C∪X
```

4.3 生成频繁闭合活动模式

函数 GetPattern 用来根据频繁闭合活动集生成有向图表示的频繁活动模式。对于其中的每一个频繁闭合活动集 C_i，函数首先计算 C_i 所对应的活动集 $AS(C_i)$。接着对于每个活动 $a_i\in AS(C_i)$，计算 a_i 的直接前趋活动集 *FormerSet*、直接后继活动集 *TrailingSet* 和 *ParallelSet*。最后根据这 3 个集合生成有向图表示的频繁模式。

函数 *GetPattern*(C)

```
for each Ci in C
  for each dij in Ci
    if dij>0 then
      add ai to aj. FormerSet and add aj to ai. TrailingSet
    else if dij<0 then
      add ai to aj. ParallelSet and add aj to ai. ParallelSet
  for each ai in AS(Ci)
    if ai. FormerSet=ϕ then
    for each aj in ai. TrailingSet
      add a directed edge from ai to aj in digraph
    else if ai. TrailingSet=ϕ
      for each aj in ai. FormerSet
      if there does not exits a directed edge from aj to ai then
        add a directed edge from aj to ai in digraph
    else if ai. FormerSet≠ϕ and ai. TrailingSet≠ϕ then
      for each aj in ai. FormerSet
        if there does not exits a directed edge from aj to ai then
          add a directed edge from aj to ai in digraph
      for each aj in ai. TrailingSet
        add a directed edge from ai to aj in digraph
    if ai. ParallelSet≠ϕ then
      for each aj in ai. TrailingSet
        if there exists a route from ai to aj then
          delete Ci from C
    if there has existed a same pattern then deletes the pattern
    end
```

4.4 一个实例

图 1 为某公司处理客户投诉的工作流模型，本文使用与文献[7]一致的日志集进行频繁模式挖掘测试。最小支持度阈值 min_sup=0.4。算法执行过程如下。

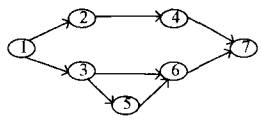

图 1 客户投诉工作流模型

第 1 步 按照 4.1 节的方法进行预处理后的运行轨迹集合为：

$$\sigma^{(1)}=\{a_1^s,a_1^e,a_2^s,a_3^s,a_2^e,a_4^s,a_3^e,a_5^s,a_4^e,a_5^e,a_6^s,a_6^e,a_7^s,a_7^e\}$$

$$\sigma^{(2)}=\{a_1^s,a_1^e,a_2^s,a_3^s,a_3^e,a_5^s,a_2^e,a_4^s,a_5^e,a_6^s,a_4^e,a_6^e,a_7^s,a_7^e\}$$

$$\sigma^{(3)}=\{a_1^s,a_1^e,a_3^s,a_2^s,a_3^e,a_6^s,a_2^e,a_4^s,a_4^e,a_6^e,a_7^s,a_7^e\}$$

$$\sigma^{(4)}=\{a_1^s,a_1^e,a_2^s,a_3^s,a_3^e,a_6^s,a_6^e,a_2^e,a_4^s,a_4^e,a_5^s,a_7^e\}$$

$$\sigma^{(5)}=\{a_1^s,a_1^e,a_2^s,a_3^s,a_3^e,a_5^s,a_5^e,a_6^s,a_2^e,a_4^s,a_6^e,a_4^e,a_7^s,a_7^e\}$$

第 2 步 根据定义 4 和定义 5 提取各运行轨迹中蕴含的直接依赖和交叠关系，将活动运行轨迹表示成如下形式：

$$\sigma^{(1)}=\{d_{12},d_{13},d_{24},d_{35},d_{46},d_{56},d_{67}\}\cup\{p_{23},p_{34},p_{45}\}$$

$\sigma^{(2)}=\{d_{12},d_{13},d_{35},d_{24},d_{56},d_{67}\}\cup\{p_{23},p_{25},p_{54},p_{46}\}$

$\sigma^{(3)}=\{d_{13},d_{12},d_{36},d_{24},d_{47},d_{67}\}\cup\{p_{32},p_{26},p_{64}\}$

$\sigma^{(4)}=\{d_{12},d_{13},d_{36},d_{64},d_{24},d_{47}\}\cup\{p_{23},p_{26}\}$

$\sigma^{(5)}=\{d_{12},d_{13},d_{35},d_{24},d_{56},d_{67},d_{47}\}\cup\{p_{23},p_{25},p_{26},p_{64}\}$

第3步　根据定义10得到*DO*矩阵如下：

$$DO_1=\begin{array}{c}a_1\\a_2\\a_3\\a_4\\a_5\\a_6\\a_7\end{array}\begin{bmatrix}0&3&3&0&0&0&0\\0&0&-3&3&-2&-1&0\\0&0&0&-1&3&0&0\\0&0&0&0&-1&-1&0\\0&0&0&-1&0&3&0\\0&0&0&-1&0&0&3\\0&0&0&0&0&0&0\end{bmatrix}$$

$$DO_2=\begin{array}{c}a_1\\a_2\\a_3\\a_4\\a_5\\a_6\\a_7\end{array}\begin{bmatrix}0&2&2&0&0&0\\0&0&-1&2&-2&0\\0&-1&0&0&2&0\\0&0&0&0&0&2\\0&0&0&-1&0&0\\0&0&0&0&0&0\end{bmatrix}$$

第4步　计算1-*itemset*，根据各项的权重进行排序得*f-list*=$\{d_{36}\times(34),d_{47}\times(34),p_{45}\times(12),p_{25}\times(25),p_{26}\times(345),p_{46}\times(235),d_{35}\times(125),d_{56}\times(125),d_{67}\times(125),d_{12}\times(12345),d_{13}\times(12345),d_{24}\times(12345),p_{23}\times(12345)\}$。

第5步　将第4步中得到的*f-list*作为参数传入函数*CHARM-Extened*([*P*],*C*)，挖掘得到的频繁闭合活动集按支持度由大到小排列：

$d_{36}d_{47}p_{26}d_{12}d_{13}d_{24}p_{23}$(40%)

$p_{45}d_{35}d_{56}d_{67}d_{12}d_{24}d_{13}p_{23}$(40%)

$p_{25}p_{46}d_{35}d_{56}d_{67}d_{12}d_{13}d_{24}p_{23}$(40%)

$p_{26}p_{46}d_{12}d_{13}d_{24}p_{23}$(40%)

$p_{26}d_{12}d_{13}d_{24}p_{23}$(60%)

$p_{46}d_{12}d_{13}d_{24}p_{23}$(60%)

$d_{35}d_{56}d_{67}d_{12}d_{24}d_{13}p_{23}$(60%)

$d_{12}d_{13}d_{24}p_{23}$(80%)

第6步　生成有向图表示的频繁活动模式，结果见图2。图3为文献[7]的挖掘结果。工作流模型中的活动a_2和a_3具有并行关系。对比挖掘结果发现，本文提出的算法可以更好地发现工作流中的频繁并行关系模式。

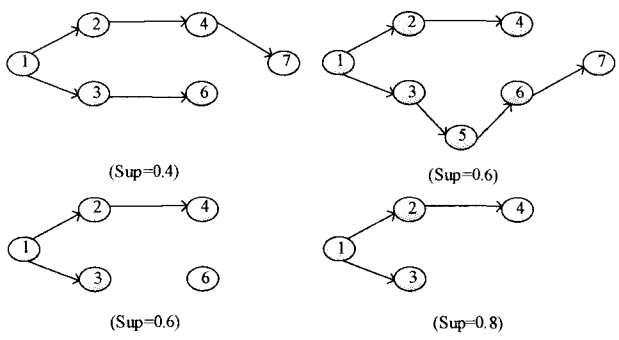

图2　工作流频繁活动模式

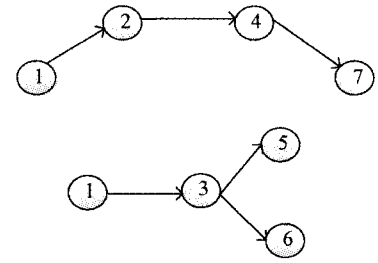

图3　文献[7]挖掘结果

结束语　工作流频繁模式挖掘是工作流挖掘领域的一个重要的研究分支，它可以实现对商业流程的智能化管理、在关键决策点进行流程预测和风险预测。本文提出的算法能够较好地处理工作流中的并行和选择关系，与其他的工作流模式挖掘算法相比，在处理活动间的并行关系时具有一定的优越性。

参 考 文 献

[1] van der Aalst W M P, van Dongen B F, Herbst J, et al. Workflow mining: A survey of issues and approaches[J]. Data and Know ledge Engineering, 2003, 47(2): 237-267

[2] van der Aalst W M P, Weijters T, Maruster L. Workflow mining: Discovering process models from event logs[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(9): 1128-1142

[3] Agrawal R, Srikant R. Mining sequential patterns [C] // Proceedings of International Conference on Data Engineering. Washington D C, USA: IEEE, 1995: 135-139

[4] Srikant R, Agrawal R. Mining sequential patterns: generalizations and performance improvement s[C] // Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology. London, UK: Springer-Verlag, 1996: 3-17

[5] Mannil H, Toivonen H, Verkamo A I. Discovering frequent episodes in sequences[C] // Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining. Menlo Park, Cal., USA: AAAI, 1995: 194-204

[6] Greco G, Guzzo A, Manco G. Mining and reasoning on workflows[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(4): 519-534

[7] 高昂，杨扬，王钥薇. 一种新的工作流频繁模式挖掘算法研究[J]. 计算机科学，2009，36(9)：231-233，251

[8] Zaki M J, Hsiao C. CHARM: An efficient algorithm for closed association rule mining[R]. Technical Report. 1999: 99-100

[9] Pasquier N, Bastide Y, Taouil R, et al. Discovering frequent closed itemsets for association rules[C] // Proc. 7th Int. Conf. Database Theory. 1999: 398-416