流程增量挖掘中的模型更新方法

马 慧 汤 庸 吴凌坤
(中山大学计算机科学系 广州 510275)

摘 要 正确发现流程实际运作情况对工作流管理有着重要的意义。流程挖掘抽取系统日志信息,挖掘流程的真实运作模型。目前很多该方面的研究,着重于从一份日志中挖掘出工作流模型。然而,这些挖掘方法只关注日志信息,忽略了流程设计者的先验知识。而且,日志所包含信息量较大,进行一次挖掘耗费较大。因此,希望能结合已有工作流模型及新增日志信息,更新工作流模型。已有研究给出对模型及日志的增量挖掘算法。但是,业务流程会随着时间推移变更,可能已有的任务被取消了,因此在新增的一段日志中该任务没被记录。但由于该任务曾经在已有日志中记录下来,故应用已有挖掘算法或增量挖掘算法,在更新模型中,该任务也会被挖掘出来。提出了一种增量挖掘模型更新的改进算法。通过流程设计者的先验知识及统计任务出现的频率,判断该任务是否被取消。最后给出一个实验,验证算法的可行性。

关键词 流程挖掘,增量挖掘,工作流模型
中图法分类号 TP181 **文献标识码** A

Incremental Process Mining with New Model Update Method

MA Hui TANG Yong WU Ling-kun
(Department of Computer Science, Sun Yat-Sen University, Guangzhou, 510275, China)

Abstract A thorough understanding of the way in which a workflow process is executing is essential to workflow management. By extracting information from work traces, such as system log data, process mining aims to discover the actual behavior of a workflow process. Current researches mainly stress on discovering a workflow model from a whole log. These process mining approaches only use the log's information, while losing sight of the process disigners' prior knowledge. Besides, the large volumn of data which the log contains makes the process mining a time-consuming job. It is expected that the new model is derived by conbining information of the existing model and the new incremental log. The problem of incremental process mining was studied. However, workflow process may change as the time goes by. It is possible that a certain task has been canceled. Though it is not recorded in the incremental new log, it has been recorded down in the old ones. Thus by applying current process mining algorithms, or the proposed incremental process mining method, the new workflow model still contains the cancelled task. An improved incremental model update method was proposed. Whether a task is cancled or not is decided by prioror knowledge and its execution frequency. An experiment was given to show the validity of this method.

Keywords Process mining, Incremental mining, Workflow model

1 引言

工作流是一类能够完全或部分自动执行的经营过程[1]。对于大型的、流程复杂的应用系统,准确地设计一个工作流模型非常困难,设计者需要参阅企业资料或咨询过程参与者的经验,花费大量的时间和成本。但是,受设计者对模型的理解和流程参与者主观经验的约束,往往设计出来的模型跟流程实际运作有一定的偏差。此外,业务流程会随着时间推移而变更。因此,往往不能在工作流系统投入使用之前,就设计出一个完善的、固定的工作流模型。流程挖掘(Process Mining)从系统记录的大量流程执行实例数据中(例如企业应用系统中的事件日志等),挖掘符合企业业务流程的工作流模型,向工作流设计者反馈流程实际运作信息,以降低工作流模型再设计的成本。

现已有很多关于流程挖掘的研究[2-7]。这些研究工作大多数从一份系统日志中,采用基于机器学习[3]、概率统计[4,5]、启发式规则[6]、遗传算法[7]等方法,发现日志中任务之间的执行先后关系;利用任务执行的时序关系,重组成一个工作流模型。然而,这些方法主要存在3个缺陷:(1)在建模阶段,模型设计者根据已有经验或相关法规,知道模型的部分

到稿日期:2008-06-24 本文受国家自然科学基金(60673135,60373081)重点项目(60736020),教育部新世纪优秀人才支持计划(NCET-04-0805),广东省自然科学基金(7003721)资助。
马 慧 博士生,研究方向为工作流、协同软件,Email:mahui.sysu@gmail.com;汤 庸 博士生导师,研究方向为协同软件、数据库;吴凌坤 博士生,研究方向为数据库、数据挖掘。

信息。而现有挖掘算法的研究仅参考了日志信息，忽略了建模时的先验知识[8]。(2) 日志中往往数据量较大。如果需要更新已有模型，若对整个日志信息重新挖掘一遍，则耗费较大。若能采用“增量式”的方法，利用已有模型的信息，整合已有模型及新增日志信息，得到新的模型，则能提高更新模型的效率。(3) 业务流程会随着时间推移变更，可能已有的任务被取消了，在新增的一段日志中该任务已经没被记录。但由于该任务曾经在已有日志中记录下来，故应用已有挖掘算法，在更新模型中，该任务也会被挖掘出来。

以上3点，均要求对一个已知的、部分正确的模型，利用新增日志信息，挖掘出一个更新的、更完善的模型。目前，这种增量挖掘的研究还不是很多。文献[9]根据日志中的第一条流程记录，生成一个符合该流程信息的模型；然后逐条扫描其余流程记录，根据当前扫描流程的活动执行先后顺序关系，判断该流程是否与已有模型一致。若不一致，则修正已有模型，得到新的工作流模型。算法每一步均用一条新流程记录更新已有模型，因此也可以实现增量挖掘。文献[10]的算法思想跟文献[9]的类似，不同之处在于，该文针对每个执行生成对应的模型，再将当前模型与已有模型合并得到新模型，直到新模型跟已有模型相同为止，停止更新。但是，它们的共同缺点是：算法均使用了每一条流程记录的信息，包括正确的记录以及噪音记录，但含有噪音的记录会影响到模型的更新。因此，该算法对噪音敏感。文献[8]中对增量挖掘作了更为深入的研究，给出了增量挖掘的3种操作：完善(Complete)、更新(Update)及比较(Compare)。文献[8]给出的方法能较好地处理噪音。具体内容在下文中将会讨论到。但是，更新操作只考虑到增量日志中新增任务的信息，没有讨论到已有模型中某些任务可能被“删除”的情况，例如，这些任务可能被取消了，或根本没必要执行。

本文就上述问题给出了一个基于文献[8]的日志更新操作改进算法。本文第2节，简要介绍文献[8]的增量挖掘算法，第3节对该算法进行扩展，第4节进行实验验证。最后，对工作进行总结并讨论了下一步的工作方向。

2 增量挖掘算法介绍

文献[8]给出的增量挖掘算法是基于文献[4]提出的流程挖掘算法。下面首先简单介绍文献[4]的流程挖掘算法，然后介绍文献[8]的增量挖掘算法。

2.1 流程挖掘算法

文献[4]的算法采用有向无环图(Directed Acyclic Graph)表示工作流模型。图中结点，可以分成以下三类：

- 普通结点：结点入度、出度均不大于1；
- 分支结点：出度大于1、入度不大于1的结点；
- 合并结点：入度大于1、出度不大于1的结点。

其中普通结点表示日志中的任务，分支、合并结点控制模型结构。算法假设表示模型的有向图不包含环路(即模型不包含循环结构)，而且模型中的结构(并行结构、选择结构)是相互嵌套的。

算法将活动的发生概率分布对应到相应的工作流模型结构上：如果在流程记录中两个活动的出现在某个活动出现的前提下是独立的，那么对应在有向无环图中该两个顶点被d-分离(d-separation)。例如，在日志信息统计中，活动A，B在以活动C的前提下是独立的，那么在相应的DAG中，A和B被C d-分离。

文献[4]的算法用到两个矩阵：O矩阵(Ordering Oracle)与I矩阵(Independence Oracle)。O矩阵记录活动间的时序依赖信息。对于两个任务 t_1,t_2，$O(t_1,t_2)$返回true，false或者exclusive：

- $O(t_1,t_2)$=exclusive，当且仅当 t_1,t_2 不会同时出现在同一条日志流程记录中。
- $O(t_1,t_2)$=true，如果 t_1 是 t_2 的祖先，或者 t_1,t_2 分别是一个与分支结点的两条分支路径上的第一个任务。换句话说，如果 $O(t_1,t_2)$=true，则 t_2 不是 t_1 的祖先。

如果有 $O(t_1,t_2)$=true，$O(t_2,t_1)$=true，则说明 t_1,t_2 之间的执行没有时序要求，可以判断 t_1,t_2 是两个并行任务。但是，如果有 $O(t_1,t_2)$=true，$O(t_2,t_1)$=false，不能说明 t_1 是 t_2 的祖先。判断两个任务是否处于并行结构，还要借助独立性测试。

I 矩阵记录了任务间的独立性信息。$I(t_1,t_2,t_3)$=true，如果在日志中，在 t_3 的前提下，t_1,t_2 的出现是独立的。

该文中 O,I 矩阵的计算，通过对日志数据进行如二项测试、卡方测试等统计获得。该文的实验表明，该计算方法能解决噪音问题。

针对工作流模型的后向确定性(backward determination)，该文给出了一个保证学习一致性的假设，并给出了利用 O,I 矩阵信息进行模型挖掘的学习算法。该文证明了在该假设下，利用 O,I 矩阵学习出来的模型是唯一的。关于 O,I 矩阵、学习算法及证明的详细讨论，请参考文献[4]。

2.2 增量挖掘算法

增量挖掘需要将已有模型信息跟新增日志信息进行合并，产生新的工作流模型。若直接产生新增日志表示的模型，再与已有模型合并，是件不容易的事情。因为表示工作流流程有多种模型，例如Petri网、有向图等。对于同一组日志数据，采用不同算法能产生不同类型的工作流模型。对两个不同类型的模型进行比较是比较困难的。然而，对不同模型均可以获得模型中任务间的时序关系，即得到 O,I 矩阵。基于文献[4]的工作，如果能得到 O 矩阵和 I 矩阵的信息，则可以挖掘出唯一的一个工作流模型。文献[8]利用该结论，设计了增量挖掘的算法。该算法分成4个步骤：(1) 从已有模型中得到任务之间的关系矩阵 O_1,I_1；(2) 从新增日志中，得到矩阵 O_2,I_2；(3)采用某种操作(完善或更新操作)合并 O_1,O_2，得到 O，及合并 I_1,I_2 得到 I；(4) 根据 O,I，调用文献[4]的算法，挖掘工作流模型。

文献[8]给出了从已有模型中得到 O,I 矩阵的算法。关于 O,I 矩阵的合并，文献[8]给出了两种操作，供用户选择。

- 完善操作：设计者根据先验知识或相关法规，可得到模型的部分信息。完善操作在一个已进行部分建模的模型基础上，根据新增日志信息，在模型中添加新增任务，完成整个模型的生成。具体操作中，O,I 矩阵的信息以 O_1,I_1 矩阵为主，O_2,I_2 矩阵为附加信息。即对于在已有模型或新增日志中的任务 t_1,t_2,t_3：

$$O(t_1,t_2)=\begin{cases}O_1(t_1,t_2), & \text{如果 } t_1,t_2 \text{ 均在已有模型中；}\\ O_2(t_1,t_2), & \text{如果 } t_1 \text{ 或 } t_2 \text{ 不在已有模型中。}\end{cases}$$

$I(t_1,t_2,t_3)=$

$$\begin{cases} I_1(t_1,t_2,t_3), \text{如果 } t_1,t_2,t_3 \text{ 均在已有模型中;} \\ I_2(t_1,t_2,t_3), \text{如果 } t_1 \text{ 或 } t_2 \text{ 或 } t_3 \text{ 不在已有模型中。} \end{cases}$$

·更新操作:更新操作用近期的日志信息更新已有模型。在具体操作中,与完善操作相反:O,I 矩阵信息以 O_2,I_2 矩阵为主,O_1,I_1 矩阵为附加信息。

$$O(t_1,t_2)=\begin{cases} O_2(t_1,t_2), \text{如果 } t_1,t_2 \text{ 均在新增日志中;} \\ O_1(t_1,t_2), \text{如果 } t_1 \text{ 或 } t_2 \text{ 不在新增日志中。} \end{cases}$$

$$I(t_1,t_2,t_3)=$$

$$\begin{cases} I_2(t_1,t_2,t_3), \text{如果 } t_1,t_2,t_3 \text{ 均在新增日志中;} \\ I_1(t_1,t_2,t_3), \text{如果 } t_1 \text{ 或 } t_2 \text{ 或 } t_3 \text{ 不在新增日志中。} \end{cases}$$

随着流程的变更,已有模型中的某些任务可能是被取消了的,这些任务在新增日志中不再出现。但是,文献[8]的更新操作并没有对这个可能性做判断,仍然将这些任务保留在更新后的模型中。下一节中给出更新操作的改进算法。

3 更新操作改进算法

针对上述问题,可采取以下更新策略:任务 t 是否属于更新模型,取决于该任务所被执行的频率高低。日志中记录了工作流中每个流程的执行情况。每次调用挖掘算法的时候,可以通过扫描日志,记录该任务被执行的流程次数及日志流程的总数。这些信息可以保存下来,以供下一次更新操作使用,如图 1 所示。

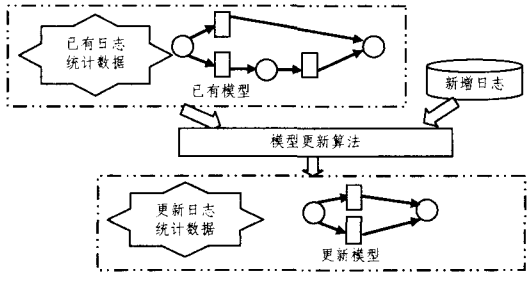

图 1 更新算法过程示意图

对于已有模型及新增日志中的任务,统计其在新增日志中出现次数与在历史数据及新增日志中总共出现的次数的比率。若该比率超过一定阈值,则将该任务添加进更新模型中,否则舍弃该任务。

但是,这种更新策略存在一个问题:在原有模型中,尽管有的任务被执行的频率比较低,但并不代表该任务已被取消。例如,原有模型中有一个异常处理任务:当流程某个环节出现异常时,则执行该任务。流程执行中出现异常频率低,导致该异常处理任务执行频率低。然而,上述策略会把该任务删除掉。不过,流程挖掘向设计者提供一个比较符合实际运作情况的大致模型。期间需要设计者的参与。在这种情况下,需要借助于设计者的先验知识。若是先验知识已知某任务在模型是必须的,则不删除该任务。下面给出算法中相关变量的含义及算法伪代码。

ExistWfN:已有工作流模型。模型可采用 petri 网、有向图等方式表示。算法对模型类型没有要求。文献[8]中已给出从模型得到 O、I 矩阵的算法。

IncLog:新增日志。

Task(ExistWfN):表示模型 ExistWfN 中包含的任务集合。

ReqTask(ExistWfN):表示模型 ExistWfN 中必须的任务的集合。

Task(IncLog):表示日志 IncLog 中包含的任务的集合。

$CExist_t$:表示任务 t 在历史记录中出现的次数。

$CInc_t$:表示任务 t 在新增日志中所被执行的流程的次数。

CExistAll,CIncAll:分别表示历史记录流程个数、新增日志流程个数。

σ:阈值,对于任务 t,当 $(CExist_t+CInc_t)/(CExistAll+CIncAll)\geqslant\sigma$ 时,t 才被保留在新模型。

更新算法伪代码如图 2 所示。

```
算法 IncrementalUpdate
输入:ExistWfN,已有工作流模型
      IncLog,新增日志信息
输出:WfN,更新后的模型
1. 读取已有模型信息 ExistWfN、CExistAll;
2. 对每个历史记录任务 t,读取 CExistt;
3. 从 ExistWfN 中,计算 O1,I1;
4. 扫描 IncLog,计算 CIncAll。并对每个任务 t,计算 CInct;
5. 对 IncLog 进行统计,计算 O2,I2;
6. 更新 CExistAll,令 CExistAll←CExistAll+CIncAll
7. 令集合 T=Task(IncLog);
8. 对任意 t∈Task(ExistWfN)∪Task(IncLog)
a. CExistt←CExistt+CInct
  b. 若 t∈Task(ExistWfN)- ReqTask(ExistWfN)且 CExistt / CExistAll≥σ,则 T←T∪{t};
9. 对于任意 t1,t2∈T
  c. 若 t1∈Task(IncLog)且 t2∈Task(IncLog),则 O(t1,t2)= O2(t1,t2);
  d. 否则,若 t1∈Task(ExistWfN)且 t2∈Task(ExistWfN),则 O(t1,t2)= O1(t1,t2);
  e. 否则,O(t1,t2) =false;
10. 对于任意 t1,t2,t3∈T
  f. 若 t1∈Task(IncLog)且 t2∈Task(IncLog) 且 t3∈Task(IncLog),则 I(t1,t2,t3)= I2(t1,t2,t3);
  g. 否则,若 t1∈Task(ExistWfN)且 t2∈Task(ExistWfN)且 t3∈Task(ExistWfN),则 I(t1,t2,t3)= I1(t1,t2,t3);
  h. 否则,I(t1,t2,t3) =false;
11. 根据 O,I,挖掘 WfN;
12. Return WfN;
```

图 2 改进更新算法

令新增日志中记录了 N 条流程。算法中对任务的出现次数的统计,只需要扫描一遍日志即可。对 O,I 矩阵的计算也可扫描一遍日志得到,故复杂度为 $O(N)$。若模型、日志中共有 n 个任务,则算法中第 9 步对 O 矩阵的更新时间复杂度为 $O(n^2)$,第 10 步对 I 矩阵的更新时间复杂度为 $O(n^3)$。第 11 步对模型挖掘的时间复杂度为 $O(n^3)$[4],故整个算法的时间复杂度为 $O(N+n^3)$。

4 实验

为了验证算法的有效性,编写了一个 Workflow Process Miner 原型。原型采用 java 语言编写。由于实际系统日志数据较难获得[4],本文采用模拟实验生成数据。(1)首先由程序随机生成一个工作流模型 Model 及其中任务执行次数、流程记录数等数据。采用 XML 格式文件记录模型及其相关数据信息。(2)调用程序随机修改 Model(包括随机地从中去掉一

些任务，及随机增加一些任务），得到 Model'，然后再根据 Model'成多条工作流实例，作为新增日志。(3)调用改进后的更新算法进行测试。下面是一个执行例子。新增日志记录了 1000 条流程记录。σ 取值 0.1。首先，读入已有模型，再读入新增日志信息。模型更新结果如图 3 所示。在新增日志中，新增了任务 H 及 I，而缺少了任务 C 及 F。由于任务 C 被指定要求保留在模型中，故在新增模型中仍然保留，而任务 F 则被去掉。

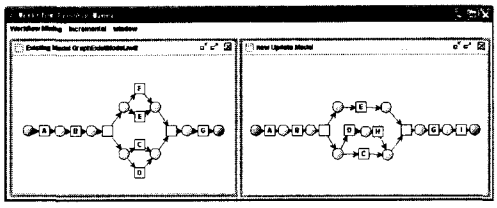

图 3 Workflow Process Miner 增量挖掘模型更新图例

结束语 工作流流程挖掘，能从系统日志记录中挖掘系统任务的执行时序关系，以工作流模型的形式向模型设计者反馈工作流流程的真正执行状态。目前很多研究仅针对一份日志信息进行挖掘处理，而关于增量挖掘方面的研究却不多。文献[8]提出一种利用新增日志更新已有模型的算法。但是，该算法没有考虑到已有模型中某些任务可能被取消的情况。本文在该算法的基础上提出了一种改进方法：对于原有模型中已存在的、在新增日志中未被执行过的任务，通过流程设计者的先验知识及任务在总记录数中出现的频率，判断该任务是否被取消。最后，通过模拟实验验证该方法的可行性。但是，对于未被指定为模型中必须存在的任务，在通过其出现频率判断其是否应保留在模型中的步骤上，仍存在一定缺陷：对于处于选择结构的任务，由于流程不一定执行其选择分支，故该任务被执行的频率低于其余流程中必须要执行的任务的频率。这样，对于不同的任务，有不同的阈值 σ。下一步的工作，将结合任务在模型中所处的结构，动态选取其阈值。

参 考 文 献

[1] 罗海滨，范玉顺，吴澄. 工作流技术综述[J]. 软件学报，2000，11(7)：899-907

[2] Aalst W M P, Weijters A J M M, Marster L. Workflow Mining: Discovering process models from event logs[J]. IEEE Transaction on Knowledge and Data Engineering, 2004, 16(9): 1128-1142

[3] Herbst J, Karagiannis K. Workflow Mining with InWoLvE[J]. Computers in Industry, 2004, 53(3): 245-264

[4] Silva R, Zhang J J, Shanahan J G. Probabilistic Workflow Mining [C]// Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. 2005: 275-284

[5] Cook J E, Wolf A L. Discovering Models of Software Processes from Event-based Data[J]. ACM Transactions on Software Engineering and Methodology, 1998, 7(3): 215-249

[6] Maruster L, Weijters A J M M, van der Aalst W M P, et al. Process Mining: Discovering Direct Successors in Process Logs [C]// Proceedings of the 5th International Conference on Discovery Science. 2002: 364-373

[7] de Medeiros A K A, Weijters A J M M, van der Aalst W M P. Genetic Process Mining: An Experimental Evaluation[J]. Data Mining and Knowledge Discovery, 2007, 14: 245-304

[8] Sun Weixiang, Li Tao, Peng Wei, et al. Incremental Workflow Mining with Optional Patterns[C]// International Conference on Systems, Man and Cybernetics. 2006, 4: 2764-2771

[9] Kim K, Ellis C A. σ-Algorithm: Structured Workflow Process Mining Through Amalgamating Temporal Workcases[C]. Advances in Knowledge Discovery and Data Mining, LNCS. 2007, 4426: 119-130

[10] Kindler E, Rubin V, Schafer W. Incremental Workflow Mining for Process Flexibility[C]// The 7th Business Process Modeling, Development and Support. 2006

(上接第 107 页)

下一步的工作可以从以下两方面展开：

(1)实现用于协同报警分析技术中的安全本体的全面构建。本文给出了基于 CIM 扩展模式、OWL 语言与 SWRL 语言的安全本体标准化构建方法，但具体的构建过程是一项庞大的系统工程，需要了解和总结现阶段发生的所有攻击的关联规则，而且有必要考虑到扩展性的需求。

(2)研究如何应用构建的安全本体在关联分析的基础上实现安全设备间自动响应。本文引入的安全本体不仅定义了环境资产信息，还定义了防御措施信息。考虑在 OWL+SWRL 本体描述的基础上，通过定义 OWL-S 来规范一组用来描述服务的知识本体，而语义标记的使用将保证联动响应控制策略这类 Web 服务能够被人和机器理解。

参 考 文 献

[1] Debar H, Curry D, Feinstein B. The Intrusion Detection Message Exchange Format (IDMEF). RFC4765, 2007

[2] Tsoumas B, Gritzalis D. Towards an Ontology-based Security Management [A]// Proceeding of 20th International Conference on Advanced Information Networking and Applications [C]. Washington, DC: IEEE Press, 2006: 985-992

[3] Patel-Schneider P F, Hayes P, Horrocks I. OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation, 2004

[4] Pras A, et al. Key Research Challenges in Network Management [J]. IEEE Communications Magazine, 2007, 45(10): 104-110

[5] Noy N, McGuiness D. Ontology Development 101: A Guide to Creating Your First Ontology [R]. No. KSL-01-05. Palo Alto: Knowledge Systems, AI Laboratory, Stanford University, 2001

[6] Holsapple C, Joshi K. A Collaborative Approach to Ontology Design [J]. Communication of the ACM, 2002, 45(2): 42-47

[7] Quirolgico S, Assis A, Westerinen A, et al. Toward a Formal Common Information Model Ontology [A]// Bussler C, et al., eds. Web Information Systems - WISE 2004 Workshops, Lecture Note in Computer Science 3307 [C]. Berlin: Springer, 2004: 11-21

[8] Horrocks I, et al. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, 2004

[9] 卢继军，黄刘生，吴树峰. 基于攻击树的网络攻击建模方法[J]. 计算机工程与应用，2003，39(27)：160-163