工作流模型挖掘算法及其应用研究

文一凭[1,2],刘 洁[2]
(1.湖南科技大学 数学与计算科学学院,湖南 湘潭 411201;
2.湖南科技大学 知识网格实验室,湖南 湘潭 411201)

摘 要:工作流管理系统由工作流模型所驱动,但产业界的实践表明定义工作流模型的工作不仅费时而且易错。工作流挖掘技术能够帮助解决这一问题,并能为现有工作流的分析与优化提供参考。简要介绍三种典型且具有应用价值的工作流模型挖掘算法,并应用其中一种挖掘算法,详细讨论了一个实际的工作流模型挖掘过程。挖掘过程以某Staffware系统的工作流日志文件为起点,包括数据预处理、初始工作流模型挖掘、初始工作流模型化简三个主要步骤,具体实现可通过一个工作流模型挖掘子系统参与完成。

关键词:工作流模型;工作流日志;工作流挖掘;挖掘算法
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2008)07-0093-03

Research on Workflow Model Mining Algorithm and Its Application

WEN Yi-ping[1,2], LIU Jie[2]
(1.School of Mathematics and Computing Science, Hunan Univ. of Science and Techn., Xiangtan 411201, China;
2.Knowledge Grid Laboratory, Hunan University of Science and Technology, Xiangtan 411201, China)

Abstract: Workflow management systems are driven by workflow models, but experience from industrial practice shows that the definition of workflow models is a time-consuming and error-prone task. Workflow mining can help to solve it. It also offers a mean to analyze and optimize existing workflows. Briefly introduces algorithms of the mining of workflow models and discusses a practical application process of an workflow model mining algorithm in detail. The mining process starts from workflow logs of a Staffware system.

Key words: workflow model; workflow log; workflow mining; mining algorithm

0 引 言

工作流管理系统(WfMS)是支持企业经营过程高效执行并监控其执行过程的计算机软件系统,能有效提高企业的信息化程度,降低企业管理成本,更好地实现企业经营目标。因此,自20世纪90年代开始,它引起了学术界与产业界的广泛关注,大量的工作流管理技术被应用到企业的实际运作。

工作流管理系统由预先建立的工作流模型所驱动,它最重要的功能之一就是工作流建模。商业化的工作流管理系统,如Staffware、COSA、ActionWorkflow等,都提供了一般的业务过程建模与执行功能。但是,产业界的实践表明定义工作流模型非常耗时而且易错,在大部分的情况下需要专家的参与。目前已有系统中建立时的过程定义与运行时的过程执行也并不能完全相符合,而动态的修改过程定义与过程实例又将带来一系列的困难。

针对这一问题,"工作流挖掘"(workflow mining)及类似的概念"过程挖掘"(process mining)被提出,即从系统工作日志中提取工作流模型。在过去的十年中,国内外学者对工作流模型挖掘算法做了大量研究。文中对现有算法进行了简要介绍,并对其具体应用进行了实例研究。应用实例以Staffware的某工作流日志文件及一个实际的挖掘算法为基础,处理过程包括数据预处理、初始工作流模型挖掘、初始工作流模型化简三个主要步骤,具体实现可通过一个工作流模型挖掘子系统参与完成。

1 工作流模型挖掘算法概述

工作流模型挖掘的目标是从业务记录中提取关于过程的工作流模型,工作起点是收集工作流过程的执

收稿日期:2007-10-02
基金项目:湖南省重点科技计划项目(2006GK3071,2007GK3054);湖南省教育科研项目(05C182)
作者简介:文一凭(1981-),男,湖南祁阳人,硕士研究生,研究方向为语义网格、工作流管理系统、数据挖掘等;导师:刘建勋,教授,研究方向为服务计算、电子商务、工作流等。

行信息,这些信息一般都保存在工作流日志中得到。但是,工作流日志中并不能记录它的每一种可能执行路线,另外,实际的工作流日志中往往是有噪声的,即部分记录不正确、不完整或者记录的仅仅是个例外。因此,工作流模型挖掘是一件富有挑战性的工作。自 Agrawal 与 Gunopulos 首次将过程挖掘的思想应用到工作流领域以来[1],一系列工作流模型的挖掘算法被提出。其中,较为典型并具有实际应用价值的有如下三个:

1)Aalst 等提出的基于工作流网的 α 算法[2];

2)Herbst 等提出的基于随机活动图的归纳式算法(InWoLvE)[3];

3)Schimm 的基于块状结构模型的多阶段挖掘方法[4]。

Aalst 等人的 α 算法能够成功处理一类与实际应用相关的工作流网(WF - net),并能构建出一个符合工作流日志行为要求的最简工作流网,但该算法在应用时要求过程中活动名是唯一的。Herbst 等人提出的归纳式算法解决了这一问题,但该算法在归纳阶段产生的随机活动图是工作流模型的次优描述,并不能清晰地表示出活动的并行行为。

与前两种方法不同,Schimm 的多阶段挖掘方法的初始挖掘模型是面向块的元模型而不是面向图形的元模型,对初始模型的优化基于符号重写规则而不是基于图形的相关技术。并且,它能挖掘出更准确的工作流模型,其生成的工作流模型符合完备性、最小性及无冗余这三个要求。该方法也有一定的局限性,例如,它没有解决多活动名问题,但这一点对实践影响不大,因为许多工作流管理系统(如 MQSeries Workflow)不会出现这一问题。此外,在应用该方法时,需要对原始工作流日志文件进行有效的数据预处理。

2 算法的具体应用实例

Staffware 是目前世界上应用最广泛的工作流管理系统之一,下面以某 Staffware 系统中的工作流日志文件为基础,讨论一个实际的工作流模型挖掘过程,在挖掘算法上使用文献[4]中 Schimm 提出的方法并假定日志记录是无噪的。

2.1 数据预处理

Staffware 的工作流日志文件记录了工作流在执行过程中的任务描述信息、事件类型信息、用户(即产生事件的用户)信息、时间(事件的发生时间)等信息。与其它数据挖掘过程相似,首先需要对此日志文件进行预处理。处理的步骤如下:

1)文件格式转换:即将原始日志文件中的记录导入到一个新的数据库文件中,各数据项依次为:实例名、活动名、事件类型、时间。在此,滤去了用户信息。

2)事件类型映射:即将原有的事件类型简单映射为"Start"与"End"两种基本类型,其分别表示活动的开始与结束,并将活动的超时与撤销等视为异常而不加处理。

3)垃圾数据清理:即清除转换后的新数据库文件中冗余的、不完整或不正确的记录。

2.2 初始工作流模型挖掘

本阶段首先生成非循环有向图(DAG),然后构建用块状结构模型表示的初始工作流模型。块状结构模型由一些嵌套的构造块组成,构造块可以分化为操作符与操作项,操作项可以是某个构造块或活动,操作符将构造块组合起来并定义了工作流的控制流。假设用 a 与 b 表示两个构造块或活动,定义下面四种操作符:

1)σ:表示嵌套块执行时的顺序关系。如 σ(a, b)表示 a 总是在 b 前执行;

2)θ:表示嵌套块执行时的并行关系。如 θ(a, b)表示 a 与 b 能同时被执行;

3)ϕ:表示嵌套块执行时的选择关系。如 ϕ(a, b)表示 a 与 b 中某一个能被执行;

4)o:表示嵌套块执行中的循环情况。如 o(a)表示 a 在循环条件满足时能被循环执行。

具体的初始工作流模型挖掘过程可分为下面几个步骤:

(1)消除多任务实例:处理记录中由于任务被循环执行等造成的多任务实例情况,保证记录中每一个任务或子工作流仅有一个实例。

(2)实例聚簇:将具有相同执行特征的工作流实例聚集到同一个记录簇中。相同执行特征主要基于任务及任务执行的先后关系,记录簇代表了工作流的执行模式。

(3)记录簇验证与合并:由 2)得到的工作流执行模式可能是一种伪模式,即其中的依赖关系并不是真正存在的。因此,有必要识别并清除这些伪模式,从而进一步合并记录簇。

(4)初始工作流模型生成:生成时仍然可以使用与 3)类似的合并思想,先根据记录簇的执行模式分别构建子工作流模型,然后将这些子模型合并为初始工作流模型。

这一阶段的算法描述如图 1 所示,其中 Step1 ~ 4 分别对应本节描述的四个步骤。

```
Input: P //由 3.1 得到的数据
Output: M //初始工作流模型
While iteration(P) exists //Step 1
```

```
Process_Multiple(P);
For records of the same case in P //Step 2
  Find_Task_Set(P);
For records of the same task set in P
  Find_Execution_Pattern(P);
  Assign_To_Cluster(P1,…, Pn);
For each cluster of P1,…, Pn //Step 3
  Find_Dependency(DP);
For each dependency_rule in DP
  Verify_Dependency(DP);
  Merge_Cluster((P1,…, Pn);
  Check_Cluster(P1,…, Pn, Q1,…, Qm)
For each cluster of Q1,…, Qm //Step 4
  Construct_Sub_Model(M1,…, Mm);
Merge_To_M(M1,…, Mm, M);
```

图1　初始工作流模型挖掘算法伪代码描述

2.3　初始工作流模型化简

由2.2得到的初始工作流模型离实际的工作流模型还有一定距离,需要应用下列的转换规则对模型中的顺序关系进行同类项合并化简,直到块状结构模型不能再被化简为止:

1)$\theta-\sigma$左同类项合并规则:$\theta(\sigma(a,b_1),\cdots,\sigma(a,b_n))\rightarrow\sigma(a,\theta(b_1,\cdots,b_n))$;

2)$\theta-\sigma$右同类项合并规则:$\theta(\sigma(a_1,b),\cdots,\sigma(a_n,b))\rightarrow\sigma(\theta(a_1,\cdots,a_n),b)$;

3)$\phi-\sigma$左同类项合并规则:$\phi(\sigma(a,b_1),\cdots,\sigma(a,b_n))\rightarrow\sigma(a,\phi(b_1,\cdots,b_n))$;

4)$\phi-\sigma$右同类项合并规则:$\phi(\sigma(a_1,b),\cdots,\sigma(a_n,b))\rightarrow\sigma(\phi(a_1,\cdots,a_n),b)$。

这一阶段的算法可用图2进行描述。

```
Input: M //初始工作流模型
Output: M //最终工作流模型
While Rules_Satified(M) is true //如果可应用合并规则
  Transform_Model(M); //对模型进行化简
End while
```

图2　初始工作流模型化简算法伪代码描述

2.4　工作流模型挖掘子系统设计

工作流模型挖掘子系统的体系结构如图3所示。子系统主要分为如下六个模块:

1)数据预处理:将原始工作流日志文件转换为XML格式文件,并进行事件类型映射与垃圾数据清理。

2)初始工作流模型挖掘:完成初始工作流模型的挖掘。

3)块状结构模型简化处理:应用转换规则对初始的块状结构工作流模型进行化简,并以图形方式输出最终的工作流模型。模型的简化合并规则存储在文件中,方便模块进行功能分解,使模块在结构上更具有灵活性。

4)XML文件处理:此模块为辅助模块,为模块1提供XML功能支持。它主要包括XML文件转换函数与XML文件操作函数。

5)图形显示与输出:此模块为辅助模块,主要为模块2,3提供图形功能支持。它由中间DAG图形与块状结构模型图形的显示与输出函数构成,能方便用户跟踪挖掘过程,并有助于用户分析与理解现有系统的工作流模型。

6)用户界面:提供友好的前台用户界面,方便用户操作工作流模型挖掘的具体过程。

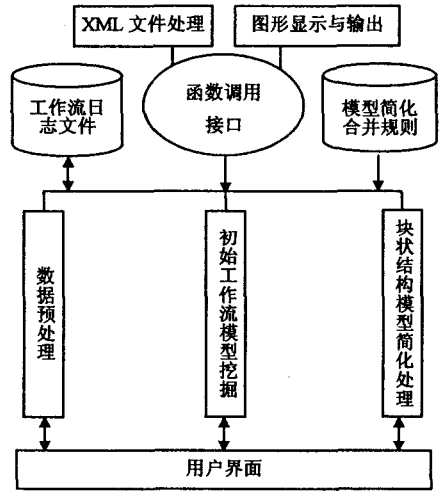

图3　工作流模型挖掘子系统体系结构图

3　结束语

工作流挖掘是一种新的工作流建模方法,能解决工作流技术应用中的模型构建问题,并能为工作流模型的再设计与优化提供参考。简要介绍了三种较典型的工作流模型挖掘算法,并对其中一种算法的具体应用进行了实例研究与探讨。但是,工作流模型挖掘在实际应用时还存在种种问题,文献[5]等对此进行了讨论。因此,如何改进算法并结合具体的应用环境以提高挖掘的有效性与准确性,使之更有效地为实践服务,都值得进一步研究讨论。

参考文献:

[1] Agrawal R, Gunopulos D, Leymann F. Mining process models from workflow logs[C]// Proceedings of the Sixth International Conference on Extending Database Technology. [s.l.]: [s.n.], 1998:469-483.

[2] van der Aalst W M P, Weijters T, Maruster L. Workflow

(下转第99页)

表1 尾递归形式与尾递归形式的时间复杂度比较(μs)

Elements Num.	ReverseList		MapList		FilterList		InsertionSortList		SumList	
	NonTail	Tail	NonTail	Tail	NonTail	Tail	NonTail	Tail	NonTail	Tail
10	17825	18025	17575	15825	2400	2655	17775	17925	755	750
50	18275	18125	19525	17625	20080	20230	18575	18680	800	800
100	19530	18180	21335	19130	20130	20330	20680	20830	850	805
500	65280	18620	22020	19820	22440	22420	96340	95940	1400	1200
1000	246400	19000	23100	21000	24000	24100	373600	368500	2000	2000
5000	11371500	25000	30000	30500	35000	30000	14946500	14090500	10000	5000

(modular programming)[9]方面相当的成熟。因此支持函数的组合来构造新的函数。

unfold递归算法的时间复杂度是二次方(n^2)。虽然将unfold算法改为尾递归(tail recursion)形式从理论上可以把时间复杂度降低到等同于循环结构,但是需要添加额外的工作:由于尾递归形式使得一维数据结构存储元素的顺序颠倒,需要额外地调用ReverseList函数来纠正顺序。

```
Unfold - tailrecUnfold - tailrec GOper Base Nil = Base
    Unfold - tailrec GOper Base (Cons x List) = Unfold - tailrec
GOper (GOper x Base) List
```

表1给出了非尾递归形式与尾递归形式的时间复杂度的实验比较数据(Pentium 4 1.8GHz平台)。

从数据来看,除了ReverseList之外,尾递归对于其他的算法在时间复杂度上优势并不明显。此外,递归抽象算法随着元素增加,时间复杂度增长温和,具有可行性。

5 结束语

递归算法构造子unfold提升了泛型算法的抽象程度。该抽象不但可以提高软件构建的复用度,而且也可以用于构造泛型递归算法,扩充算法库且不需要重复构造结构、功能相似的算法。此外,在带证明的代码系统(proof - carrying code)的研究领域,递归算法的抽象可以用于构造复用性和抽象度比较高的公理和证明,从而可以减少构造公理和算法的繁琐过程。

将来的工作也可以包括对生成算子的研究和利用该抽象用于证明算法结构的相等性,从而证明算法功能的等价性。

参考文献:

[1] Pierce B. Types and programming languages[M]. [s. l.]: MIT Press, 2002.

[2] Pitts A. Parametric polymorphism and operational equivalence [J]. Mathematical Structures in Computer Science, 2000 (10):231 - 259.

[3] Gibbons J. Design patterns as higher - order datatype - generic programs[C]// International Conference on Functional Programming, Proceedings of 2006 ACM SIGPLAN Workshop on Generic Programming. Portland, Oregon, USA:[s. n.], 2006:1 - 12.

[4] Ruehr K. Analytical and structural polymorphism expressed using patterns over types[D]. USA:University of Michigan, 1992.

[5] Pierce B. Advanced topics in types and programming languages[M]. [s. l.]: MIT Press, 2005.

[6] Sumii E, Pierce B. A bisimulation for type abstraction and recursion[J]. Journal of the ACM,2007,54(5):26 - 30.

[7] Musser D, Shao Zhiqing. Concept use or concept refinement: an important distinction in building generic specifications[C] //4th International Conference on Formal Engineering Methods, volume 2495 of Lecture Notes in Computer Science. New York: Springer - Verlag, 2002.

[8] Musser D, Shao Zhiqing. The Tecton concept description language (revised version)[R]. Computer Science Department, Rensselaer Polytechnic Institute. Troy, New York:[s. n.], 2002.

[9] Dreyer D, Crary K, Harper R. A type system for higher - order modules[C]//30th ACM SIGPLAN - SIGACT Symposium on Principles of Programming Langauges (POPL'03). New Orleans, Louisiana, USA:[s. n.],2003:236 - 249.

(上接第95页)

Mining: Discovering Process Models from Event Logs[J]. IEEE Transactions on Knowledge and Data Engineering, 2004,16(9):1128 - 1142.

[3] Herbst J, Karagiannis D. Workflow mining with InWoLve [J]. Computers in Industry,2004,53(3):245 - 264.

[4] Schimm G. Mining exact models of concurrent workflows[J]. Computers in Industry,2004,53(3):265 - 281.

[5] Hammori M, Herbst J, Kleiner N. Interactive workflow mining - requirements, concepts and implementation[J]. Data and Knowledge Engineering, 2006,56(1):41 - 63.