α_H 算法：工作流挖掘中一种能挖掘隐含任务的扩展 α 算法

马 慧[1] 汤 庸[2] 吴凌坤[3]

(电子科技大学中山学院计算机工程系 中山 528402)[1] (华南师范大学计算机学院 广州 510631)[2]

(腾讯计算机系统有限公司 深圳 518057)[3]

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。工作流挖掘抽取系统日志信息，挖掘流程的真实运作模型。其中挖掘隐含任务是工作流挖掘中待研究问题之一。基于 α 算法，提出了能挖掘隐含任务的挖掘算法 α_H。分析了隐含任务出现的可能情况，通过判断并行任务的位置关系，往工作流网中添加隐含任务；然后合并相同的隐含任务，去掉冗余隐含任务，以完善结果模型。实现了 α_H 算法原型，实验证实了方法的可行性及有效性，并分析了方法的不足之处。

关键词 工作流模型，工作流挖掘，隐含任务，工作流网，Petri 网

中图法分类号 TP181 **文献标识码** A

α_H-Algorithm: An Extended α-Algorithm to Mine Hidden Tasks

MA Hui[1] TANG Yong[2] WU Ling-kun[3]

(Department of Computer Engineering, Zhongshan Institute, University of Electronic Science and Technology, Zhongshan 528402, China)[1]

(Computer School, South China Normal University, Guangzhou 510631, China)[2]

(Tencent Inc., Shenzhen 518057, China)[3]

Abstract A thorough understanding of the way in which a workflow process is executing is essential to workflow management. By extracting information from workflow traces, such as system log data, workflow mining aims to discover the actual behavior of a workflow process. One of the challenging problems in workflow mining is to mine hidden tasks. Based on the traditional α-algorithm, an extended one which is called α_H-algorithm to mine hidden tasks was proposed. After studying the situations where a hidden task may appear, the α_H-algorithm inserts hidden tasks by judging the presences of parallel tasks. The mined workflow model was refined by fusing the same hidden tasks and removing the redundant ones. A prototype based on α_H-algorithm was implemented. Experiments in the end show the fasibility and validity of the proposed algorithm. Furthermore, the restriction of the algorithm and related future work were also discussed and pointed out.

Keywords Workflow model, Workflow mining, Hidden tasks, Workflow net, Petri net

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]。随着工作流技术的发展，工作流管理广泛应用在企业信息系统中。在传统的工作流生命周期中，通常由相关领域专家设计一个符合企业流程的工作流模型，然后配置相关应用系统，最后系统投入使用。但是，要准确地设计一个工作流模型，尤其是对于大型的、流程复杂的应用系统来说，不是一件容易的事情。有研究指出，建模过程中收集材料得到模型信息需要花费60%的时间；设计、定义模型需要花费20%的时间；而实现一个工作流仅需要20%的时间[2]。实践应用表明，流程设计者受其对模型的理解和流程参与者主观经验的约束，设计出来的模型跟流程实际运作有一定的偏差。另一方面，某些业务流程存在很大的柔性，其执行过程并非严格按照模型定义执行。因此，流程设计者不能在工作流系统投入使用之前就设计出一个很完善的、固定不变的模型。相关领域专家希望从工作流系统真正运作情况中得到有用的信息，以进行工作流模型再造。这个过程称为工作流挖掘。很多应用系统均能提供日志，这些日志记录了系统的运作情况，为工作流挖掘提供了可能。

Agrawal 等人最早提出利用工作流管理系统日志挖掘工作流模型[3]，至今已有很多关于工作流挖掘的研究。工作流包括功能、行为、信息、组织、操作5个方面的信息。现有工作大部分研究工作流行为方面的挖掘：即挖掘流程中任务执行的先后顺序关系。这些研究工作采用机器学习[4]、概率统计[5]、启发式规则[6]、遗传算法[7]等不同方法体系挖掘模型。

到稿日期：2010-05-20 返修日期：2010-10-14 本文受国家自然科学基金(60970044)，电子科技大学中山学院科研启动基金项目(409YKQ04)资助。

马 慧(1981－)，女，博士，讲师，主要研究方向为工作流挖掘、协同软件、数据库，E-mail：mahui.sysu@gmail.com；汤 庸(1964－)，男，博士，教授，主要研究方向为数据库、协同软件与Web信息服务；吴凌坤(1981－)，男，博士，主要研究方向为数据库、数据挖掘。

但是，这些方法受到一定条件的约束，例如文献[8-10]的方法要求工作流模型的结构是嵌套的；启发式规则、遗传算法方法受到日志质量的影响等等。其中 Aalst 和他的研究小组提出的 α 算法是工作流挖掘经典算法之一。α 算法采用 Petri 网表示工作流网，通过检查任务间 4 种可能的时序关系，及分析 4 种时序关系对应在 Petri 网上的表现，来挖掘工作流模型。文献[11]中给出 α 算法的严格证明，并讨论了 α 算法可以挖掘的模型种类。对比其它算法，严格的数学证明使 α 算法具有了坚实的理论基础。在此基础上，Aalst 和他的研究小组提出了 α 系列算法，以完善 α 算法。Wen 等人通过检查任务执行时间是否重叠显式地判断任务并行关系，放宽了日志完备性的要求，并在此日志模型上提出了改进的 α 算法[12]。Medeiros 等人通过分析 α 算法缺陷，提出了能处理短循环的方法[13]。Wen 等人提出 α^+ 算法，使其能挖掘非自由选择结构的工作流网[14]。但是，α 算法的特点之一是挖掘出来的 Petri 网中的变迁仅对应日志中已记录的任务，即日志中记录的任务与 Petri 网中的变迁一一映射。而 Petri 网中往往需要一些隐含任务起路由的作用[15]，这些隐含任务由于没有记录在日志中，因此没有被 α 算法挖掘出来。

α 算法不能挖掘隐含任务将导致两个问题：一是缺少隐含任务会导致工作流网不满足合理性，引起死锁，如图 1 所示。原工作流网 $N1$，其可能日志是 $W=\{ABCDE, ACBDE, AFE\}$。$N2$ 是采用 ProM5.2[16] α 算法插件挖掘出来的工作流网。尽管 $N2$ 也能描述 W，但 $N2$ 存在死锁可能：变迁 A 产生两个托肯(token)，若这两个托肯分别被变迁 B，F 消耗，则 F 无法被使能，网络处于死锁状态。

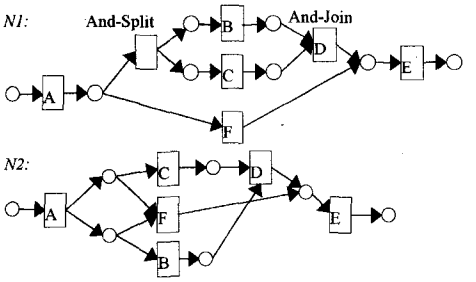


图 1 α 算法不能挖掘隐含任务，结果模型存在死锁

α 算法不能挖掘隐含任务导致的另一个问题是结果模型不正确，如图 2 所示。$N3$ 是原模型，$N4$ 是 α 算法挖掘结果。在 $N3$ 中，任务 B 和 C 可以同时执行；但是在 $N4$ 中，任务 B 和 C 构成选择结构，因为 α 算法不能挖掘出 $N3$ 的两个隐含任务：And-Split 与 And-Join。

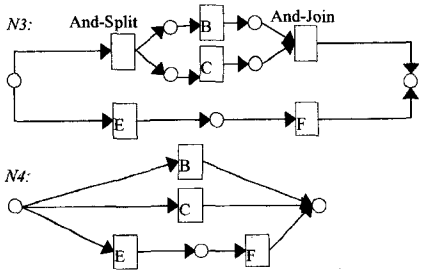


图 2 α 算法不能挖掘隐含任务，结果模型不正确

挖掘隐含任务是工作流挖掘中待解决的问题之一[17]。本文探讨了隐含任务挖掘问题，扩展 α 算法，提出 α_H 算法。α_H 算法首先采用机器学习方法分析隐含任务可能出现的情况，往工作流网添加隐含任务；然后通过合并隐含任务、去除冗余隐含任务等操作完善工作流网。本文第 2 节给出工作流网模型及日志模型的定义；第 3 节介绍并解释 α_H 算法；第 4 节验证算法的有效性，并讨论了算法的局限性；最后总结全文工作。

2 基本概念

2.1 工作流模型

本文采用 Petri 网表示工作流模型。下面介绍其相关定义[15]。

定义 1 Petri 网 $N=(P,T,F)$ 其中 P 是库所(Place)的非空有限集合，T 是变迁(Transition)的非空有限集合，$F\subseteq(P\times T)\cup(T\times P)$ 是有向弧的集合。

定义 2 前驱集(PreSet)、后继集(PostSet) 令 $N=(P,T,F)$ 是 Petri 网，$x\in(P\cup T)$，$^\cdot x=\{y\mid y\in(P\cup T),(y,x)\in F\}$ 是 x 的前驱集，$x^\cdot=\{y\mid y\in(P\cup T),(x,y)\in F\}$ 是 x 的后继集。

定义 3 工作流网(Workflow Net)：令 $N=(P,T,F)$ 为 Petri 网，N 当且仅当：(1) 有一个输入库所 i，$^\cdot i=\emptyset$；(2) 有一个输出库所 o，$o^\cdot=\emptyset$；(3) 令 $t'\notin(P\cup T)$，$N'=(P,T\cup\{t'\},F\cup\{(o,t'),(t',i)\})$ 为强连通图。

关于 Petri 网与工作流元素的对应关系、Petri 网变迁的触发规则、状态转移规则、触发序列等详细讨论，可参见文献[15]。

2.2 日志模型

系统日志记录了工作流流程的实际执行情况。工作流建模时定义了流程需要执行的多个任务(Task)。任务的一次具体执行叫做活动(Activity)[14]。

定义 4(活动，Activity) 设 $Time$ 为时间域，$Task$ 为任务的有限非空集合。$A\subseteq Task\times Time\times Time$ 表示活动的集合。$a=(task,t_s,t_e)\in A$ 表示该活动执行了任务 $task$，起始时间为 t_s，终止时间为 t_e，且满足 $t_s<t_e$。分别用 $a.task$，$a.t_s$，$a.t_e$ 相应地指代元组 a 的 3 个分量。

定义 5(工作流轨迹，Workflow Trace) $\sigma\in A^+$ 表示一个流程的工作流轨迹。$\sigma=a_1a_2\cdots a_m$，对于 $1\leqslant i<j\leqslant m$，有 $a_i.t_s\leqslant a_j.t_s$。$\sigma_i=a_i$ 表示 σ 中第 i 个活动。$a\in\sigma$ 表示活动 a 在 σ 中。$t\in\sigma$ 当且仅当 $\exists a\in\sigma, a.Task=t$。$|\sigma|=m$ 表示 σ 的活动序列长度。$fist(\sigma)=\sigma_1.Task$，$last(\sigma)=\sigma_m.Task$。

σ 记录了工作流流程一次执行实例中各活动的执行时间。在流程挖掘中，我们只关注任务执行的先后顺序，不需要关心任务具体在哪个时刻开始执行、哪个时刻结束。因此在下文中，我们省略了每个活动的具体执行时间，用任务名称表示一个活动，用任务序列表示 σ。至于两个任务是否可以并行执行，则从 σ 中任务执行的时间重叠关系来推断。

定义 6(工作流日志，Workflow Log) 用 $W\subseteq A^+$ 表示工作流日志。

2.3 任务依赖关系

下面给出任务间依赖关系定义。对于工作流日志 W、任务 $t_1,t_2\in Task$，有：

定义 7 优先> $t_1>t_2$，如果 $\exists\sigma\in W$，对于 $1\leqslant i,j\leqslant|\sigma|$，$\sigma_i.Task=t_1$，$\sigma_j.Task=t_2$，$\sigma_i.t_e<\sigma_j.t_s$，且 $\neg\exists a\in\sigma,\sigma_i.t_e<a.t_s<a.t_e<\sigma_j.t_s$。

定义 8 重叠× $t_1 \times t_2$，如果 $\exists \sigma \in W \exists a_1, a_2 \in \sigma, a_1.Task=t_1, a_2.Task=t_2$，且 $a_1.T_s \leqslant a_2.T_s \leqslant a_1.T_e \vee a_1.T_s \leqslant a_2.T_s \leqslant a_1.T_e$。

定义 9 任务时序关系→，$\|$ 与 #

(1) $t_1 \rightarrow t_2$，当且仅当 $t_1 > t_2 \wedge \neg(t_2 > t_1) \wedge \neg(t_1 \times t_2)$。

(2) $t_1 \| t_2$，当且仅当 $((t_1 > t_2) \wedge (t_2 > t_1)) \vee (t_1 \times t_2)$。

(3) $t_1 \# t_2$，当且仅当 $\neg(t_1 > t_2) \wedge \neg(t_2 > t_1) \wedge \neg(t_1 \times t_2)$。

由定义 9，→，$\|$ 与 # 划分了 $Task \times Task$ 上的二元关系。

3 隐含任务挖掘算法

本节讨论如何挖掘隐含任务。下面首先给出 α_H 算法，然后具体解释各步骤的含义。

3.1 α_H 算法描述

α_H 算法描述如下：

(1) 令 $T_W=\{t \mid \exists \sigma \in W, t \in \sigma\} \cup \{t_i, t_o\}$，$T_i=\{t \mid \exists \sigma \in W, t=first(\sigma)\}$，$T_o=\{t \mid \exists \sigma \in W, t=last(\sigma)\}$，$H_W=\{t_i, t_o\}$。

(2) $X_W=\{(A,B) \mid A,B \subseteq T_W \wedge \forall a \in A \forall b \in B, a \rightarrow b \wedge \forall a_1, a_2 \in A, a_1 \# a_2 \wedge \forall b_1, b_2 \in B, b_1 \# b_2\} \cup \{(t_i, t) \mid t \in T_i\} \cup \{(t, t_o) \mid t \in T_o\}$。

(3) $Y_W=\{(A,B) \mid \forall (A,B) \in X_W \wedge \forall (A',B') \in X_W, A \subseteq A' \wedge B \subseteq A' \Rightarrow (A,B)=(A',B')\}$。

(4) $prll_W=\{t \mid \exists t' \in T_W, t \| t'\}$。

(5) foreach $(A,B) \in Y_W$

{

(5.1) $S_A=A-prll_W$；$P_A=A-S_A$；$S_B=B-prll_W$；$P_B=B-S_B$；

(5.2) if($|P_A|=1 \wedge P_B=\emptyset$)

{

令 h 为新的隐含任务，$H_W=H_W \cup \{h\}$

$Y_W=Y_W-\{(A,B)\}+\{(S_A+\{h\},B),(P_A,\{h\})\}$

}

(5.3) if($P_A=\emptyset \wedge |P_B|=1$)

{

令 h 为新的隐含任务，$H_W=H_W \cup \{h\}$

$Y_W=Y_W-\{(A,B)\}+\{(A,S_B+\{h\}),(\{h\},P_B)\}$

}

}

(6) $Z_W=\{(A,B) \mid \forall (A,B) \in Y_W \forall (A',B') \in Y_W, (A=A' \wedge h \in (prll_W \cap B), \wedge h' \in (prll_W \cap B')) \vee (h \in (prll_W \cap A) \wedge h' \in (prll_W \cap A') \wedge B=B') \Rightarrow h=h'\}$。

(7) $R_W=\{h \mid h \in H_W \wedge \forall (\{h\},B) \in Z_W \forall (\{h\},B') \in Z_W B=B' \wedge |B|=1\}$

定义 $f: R_W \rightarrow T_W \mid f(h)=t$，其中 $(\{h\},\{t\}) \in Z_W$。

$R_W'=\{h \mid h \in H_W \wedge \forall (A',\{h\}) \in Z_W \forall (A',\{h\}) \in Z_W A=A' \wedge |A|=1\}$

定义 $f': R_W' \rightarrow T_W \mid f(h)=t$，其中 $(\{t\},\{h\}) \in Z_W$。

定义 $g: (R_W \cup R_W') \rightarrow T_W$，如果 $h \in R_W$，$g(h)=f(h)$；如果 $h \in R_W'$，$g(h)=f'(h)$。

(8) $Z_W=Z_W-\{(\{h\},\{f(h)\}) \mid h \in R_W\}-\{(\{f'(h)\},\{h\}) \mid h \in R_W'\}$。

(9) foreach $(A,B) \in Z_W$

{

(9.1) if($\exists h, h \in A \wedge h \in (R_W \cup R_W')$)

{

$A=A-\{h\}+\{g(h)\}$；

}

(9.2) if($\exists h, h \in B \wedge h \in (R_W \cup R_W')$)

{

$B=B-\{h\}+\{g(h)\}$；

}

}

(10) $T_W=\{t \mid \exists (A,B) \in Z_W \wedge (t \in A \vee t \in B)\}$，$T_i=\{t \mid \forall (A,B) \in Z_W, t \notin B\}$，$To=\{t \mid \forall (A,B) \in Z_W, t \notin A\}$。

(11) $P_W=\{p_{(A,B)} \mid (A,B) \in Z_W\} \cup \{i_W, o_W\}$。

$F_W=\{(a, p_{(A,B)}) \mid (A,B) \in Z_W \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid (A,B) \in Z_W \wedge b \in B\} \cup \{(i_W, t) \mid t \in T_i\} \cup \{(t, o_W) \mid t \in T_o\}$。

(12) $\alpha_H(W)=(P_W, T_W, F_W)$。

3.2 挖掘任务对应变迁与库所的关系

α_H 算法基于任务间的时序关系构造工作流网，其前 3 步跟 α 算法相似：首先建立流程的任务集合 T_W、各工作流轨迹起始任务集合 T_i 及结束任务集合 T_o。其中，α_H 算法给每条工作流轨迹的首尾增加了两个隐含任务 t_i 与 t_o。t_i 与 t_o 的添加并不改变日志描述的流程行为。H_W 记录隐含任务集合。步骤 2、3 求出工作流网中有哪些库所。步骤 2 求出哪些任务之间是直接关联的（即满足→关系）。如果有 $t_1 \rightarrow t_2$，则在工作流网中，t_1，t_2 代表的变迁之间有一个库所相连。因此，对于元组 (A,B)，集合 A 中任意任务均直接关联集合 B 中所有任务，且集合 A 或 B 内的任务之间都不会直接关联。步骤 3 求出满足步骤 2 的最大元组集合，目的是合并“或结构”汇合库所（Or-Join）。关于步骤 2、3 的详细讨论参考文献[11]。

3.3 添加隐含任务

α 算法不能挖掘隐含任务的原因是其中没有对具有并行关系的任务做特殊检查。步骤 4 至步骤 9 通过检查 Y_W 中各元组的任务集合中并行任务的出现情况来挖掘隐含任务。步骤 4 求出具有 $\|$ 关系的任务集合 $prll_W$。步骤 5 检查 Y_W 的每个元组 (A,B)。如果 A（或 B）包含处于并行关系的任务，则这个任务后面（或前面）可能存在隐含任务。步骤 5.1 求出 $A(B)$ 中包含的并行任务集合 $P_A(P_B)$ 以及非并行任务集合 $S_A(S_B)$。根据 P_A/P_B 的取值具体分 3 种情况分析：

(1) A 包含处于并行关系的任务而 B 不包含处于并行关系的任务。首先，我们讨论 $|P_A|=1$ 的情况，如图 3 所示。

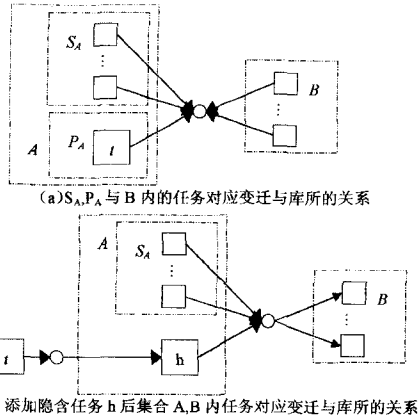


图 3 集合 A、B 内的变迁与库所对应关系

图 3(a)显示了集合 A、B 内的任务对应的变迁与库所之间的关系。因为 B 中不包含并行任务，所以 t 必定处于并行结构的某分支上的最后一个任务（否则，如果 t 所在分支后面

还有一个任务 t'，则有 $t' \in B$，违背了 $P_B = \emptyset$ 这一前提）。因此，t 后面可能存在一个隐含任务。步骤 5.2 增加一个隐含任务 h，并将 h 与 t，h 与 B 的关系加进 Y_W 中，如图 3(b)所示。当 $|P_A|>1$ 时，设 $t_1, t_2 \in P_A$。由步骤 2 得 $t_1 \# t_2$，所以 t_1, t_2 不属于图 4 所示的情况：t_1, t_2 分别是一个并行结构的不同分支的最后一个任务。此时不能往 t_1, t_2 后面添加隐含任务。关于此种情况的详细讨论见第 4.2 节。

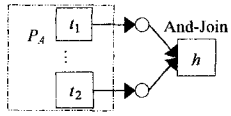


图 4　t_1, t_2 分别是一个并行结构的不同分支的最后一个任务

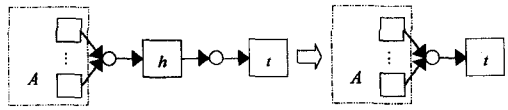
(2) B 包含处于并行关系的任务而 A 不包含处于并行关系的任务，此时处理方法类似情况(1)。

(3) A, B 均包含处于并行关系的任务。此情况下，P_A, P_B 的任务处于并行结构中，P_A, P_B 之间不需要添加隐含任务。

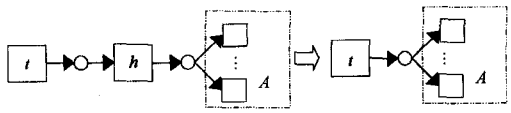
3.4　去掉冗余隐含任务

步骤 6 合并相同的隐含任务。在步骤 5 中只是添加潜在的隐含任务，但这些隐含任务可能被重复添加了。在 Y_W 中，如果存在两元组 (A,B) 与 (A',B')，$A=A'$，B 与 B' 各包含隐含任务 h 与 h'，则将 h 与 h' 合并。类似地，如果 $B=B'$，A 与 A' 各包含隐含任务 h 与 h'，将 h 与 h' 合并。

步骤 7 至步骤 9 去掉冗余的隐含任务。图 5(a)中左图表示步骤 5、6 添加了隐含任务后的状态：隐含任务 h，后面有且只有一个任务 t。在这种情况下，h 是多余的。因为在网中把 h 去掉，将 h 的前驱变成 t 的前驱（如图 5(a)右图所示），并不改变工作流网的行为。图 5(a)所示的情况对应在 Z_W 中表现为：Z_W 有且仅有一个元组 $(\{h\},\{t\})$，其中 $h \in H_W$，$t \in T_W$；如果有 $(\{h\},B) \in Z_W$，则必有 $B=\{t\}$。步骤 7 求出所有满足上述情况的 h，记为集合 R_W。函数 f 记录了替换 h 的任务 t。步骤 8 将 $(\{h\},\{t\})$ 去掉。步骤 9 把 Z_W 中所有 h 都用 $f(h)$ 代替。另一种隐含任务冗余情况如图 5(b)所示，此时做类似的处理：在步骤 7 中求出满足图 5(b)情况的冗余的隐含任务集合 R_W' 及函数 f'；步骤 8 把 $(\{t\},\{h\})$ 去掉；步骤 9 把 Z_W 中所有 h 都用 $f'(h)$ 代替。



(a) 隐含任务 h 只有一个后继任务 t



(b) 隐含任务 h 只有一个前驱任务 t

图 5　两种冗余隐含任务情形

步骤 10 构造工作流网：Z_W 中每个元组 (A,B) 代表一个库所，将库所与对应的输入变迁（A 内的任务）/输出变迁（B 内的任务）相连，生成工作流网。

4　实验验证

本文采用 Java 语言编写 α_H 算法的实验原型，日志采用 MXML 格式。实验验证了算法的有效性。本节首先以图 1 所示的业务过程日志为例，具体解释 α_H 算法的挖掘过程。然后讨论 α_H 算法的局限性。

4.1　α_H 算法挖掘过程

图 1 所示的业务流程日志为 $W=\{ABCD, ACBD, AED\}$。α_H 算法前 3 步求得的 Y_W 如表 1 所列。

表 1　Y_W 的值

集合 A	集合 B	集合 A	集合 B
t_i	A	C	D
A	B,F	D,F	E
A	C,F	E	t_o
	B	D	

步骤 5 添加隐含任务。$prll_W=\{B,C\}$。元组 $(\{A\},\{B,F\})$ 中，$\{B,F\}$ 包含一个并行任务 B，所以将该元组拆成 $(\{A\},\{h_1,F\})$ 与 $(\{h_1\},\{B\})$。同理，元组 $(\{A\},\{C,F\})$，$(\{B\},\{D\})$，$(\{C\},\{D\})$ 中均包含并行任务，也要往这些元组添加隐含任务。添加隐含任务后 Y_W 的值如表 2 所列。

表 2　Y_W 的值

集合 A	集合 B	集合 A	集合 B
t_i	A	h_3	D
A	h_1,F	C	h_4
h_1	B	h_4	D
A	h_2,F	D,F	E
h_2	C	E	t_o
	B	h_3	

步骤 6 合并相同的隐含任务。$(\{A\},\{h_1,F\})$ 与 $(\{A\},\{h_2,F\})$ 两元组中，前驱集合相同，后继集合均包含隐含任务，所以令 $h_1=h_2$。同理，得 $h_3=h_4$。计算所得 Z_W 的值如表 3 所列。

表 3　Z_W 的值

集合 A	集合 B	集合 A	集合 B
t_i	A	C	h_3
A	h_1,F	h_3	D
h_1	B	D,F	E
h_1	C	E	t_o
	B	h_3	

步骤 7—步骤 9 删除多余的隐含任务。Z_W 中关于 h_1 的元组有 $(\{A\},\{h_1,F\})$，$(\{h_1\},\{B\})$ 与 $(\{h_1\},\{C\})$。h_1 后面分别接着 B 和 C 两个任务，所以 h_1 不能被删除。关于 h_3 的元组有 $(\{B\},\{h_3\})$，$(\{C\},\{h_3\})$ 和 $(\{h_3\},\{D\})$。因为有且仅有 $(\{h_3\},\{D\})$ 这个元组，所以 h_3 可以被删掉，用 D 代替。同理，t_i 与 t_o 也可以被删除，如表 4 所列。

表 4　去除冗余隐含任务后的 Z_W 值

集合 A	集合 B	集合 A	集合 B
A	h_1,F	B	D
h_1	B	C	D
h_1	C	D,F	E

步骤 10—步骤 12 根据 Z_W 生成工作流网，其结果如图 6 所示。

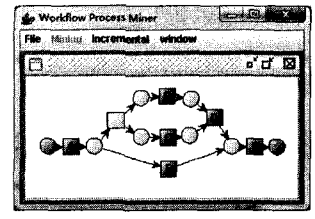


图 6　采用 α_H 算法的实验原型挖掘结果

4.2 α_H 算法的局限性

在 α_H 算法步骤 5.2 中，如果 $|P_A|>1$（或 $|P_B|>1$），则工作流网中出现这种情况：两个并行结构处于选择结构的不同分支，而同时并行结构的 And-Split(And-Join)都是隐含任务，没有记录在日志中。如图 7 所示，此时 α_H 算法不能区分 $N5$ 中的两个 And-Join 隐含任务具体汇合了 B,D,E,F 中哪些任务，不能添加隐含任务。如果两个 And-Split(或 And-Join)中，只有一个是隐含任务，则 α_H 可以挖掘出来，如图 8 所示。

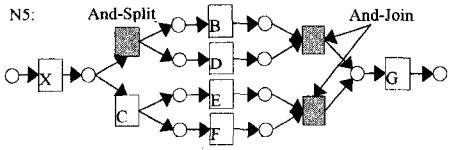


图 7　选择结构结束前出现两个隐含任务，α_H 算法不能挖掘

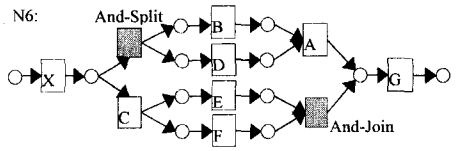


图 8　选择结构结束前只有一个隐含任务，α_H 算法可以挖掘

结束语　工作流挖掘中的 α 算法被证明能挖掘一大类工作流网，是工作流挖掘中较好的算法。但是，α 算法的缺点是不能挖掘隐含任务，而工作流网中需要一些隐含任务起路由作用。缺少隐含任务会破坏工作流网的合理性，导致死锁。本文探讨了基于 α 算法的隐含任务挖掘问题，归纳出了隐含任务可能出现的情形，提出了 α_H 算法，以在一定程度挖掘隐含任务。α_H 首先求出任务集合组对；然后根据组对中并行任务的出现位置，添加隐含任务；通过合并相同任务，去除冗余隐含任务操作来完善结果模型。本文采用 Java 语言编写了 α_H 算法程序，验证了算法的有效性，并指出了算法的不足之处。对于多个隐含任务分别处于选择结构的不同分支上的情形，α_H 不能将其挖掘出来，这将是进一步研究的工作。

参 考 文 献

[1] WFMC. Workflow Management Coalition Terminology and Glossary[Z]. Brussels, 1996

[2] Herbst J, Karagiannis D. An Inductive Ppproach to the Acquisition and Adaptation of Workflow Models[C]// Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The new Frontier for AI in Business. Stockholm, Sweden, 1999: 52-57

[3] Agrawal R, Gunopulos D, Leymann F. Mining process models from workflow logs[C]// Proceedings of the 6th. International Conference of Extending Database Technology. Valencia, Spain, 1998: 469-483

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

[5] 高昂，杨扬，王玥薇. 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

[6] Maruster L, Weijters A, van der Aalst W, et al. Process mining: discovering direct successors in process logs[C]// Proceedings of the 5th. International Conference on Discovery Science. Lübeck, Germany, 2002: 364-373

[7] Maruster L, Weijters A, van der Aalst W. Genetic process mining: an experimental evaluation [J]. Data Mining and Knowledge Discovery, 2007, 14(2): 245-304

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

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

[10] 马慧，汤庸，吴凌坤. 流程增量挖掘中的模型更新方法 [J]. 计算机科学，2009，36(5)：154-157

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

[12] Wen Lijie, van der Aalst W, Wang Jianmin, et al. A novel approach for process mining based on event types [J]. J. Intelligent Information Systems, 2009, 32(2): 163-190

[13] de Medeiros A K A, van Dongen B F, van der Aalst W M P, et al. Process mining: extending the α-algorithm to mine short loops[M]. BETA Working Paper Series, WP 113. Eindhoven: Eindhoven University of Technology, 2004

[14] Wen L, et al. Mining process models with non-free-choice constructs [J]. Data Mining and Knowledge Discovery, 2007, 15 (2): 145-180

[15] van der Aalst W. The application of Petri nets to workflow management [J]. J. Circuits Systems and Computers, 1998, 8(1): 21-66

[16] van der Aalst W. ProM Framework 5.2 [EB/OL]. http://prom.win.tue.nl/tools/prom/, 2009-10

[17] van der Aalst W, Weijters A. Process mining: a research agenda [J]. J. Computers in Industry, 2004, 53(3): 231-244

(上接第 215 页)

[2] Gupta S, Kaiser G, Neistadt D, et al. DOM based content extraction of HTML documents[C]// Proceedings of the 12th international conference on World Wide Web. 2003: 207-214

[3] 孙承杰，关毅. 基于统计的网页正文信息抽取方法的研究[J]. 中文信息学报，2004，18(5)：17-22

[4] 宋明秋，张瑞雪，吴新涛，等. 网页正文信息抽取新方法[J]. 大连理工大学学报，2009，49(4)：594-597

[5] 何昕，谢志鹏. 基于简单树匹配算法的 Web 页面结构相似性度量[J]. 计算机研究与发展，2007，44(z3)：1-6

[6] 潘有能. XML 文档自动聚类研究[J]. 情报学报，2006，25(2)：215-220

[7] 王志琪，王永成. HTML 文件的文本信息预处理技术[J]. 计算机工程，2006，32(5)：46-67

[8] 陈琼，苏文健. 基于网页结构树的 Web 信息抽取方法[J]. 计算机工程，2005，31(20)：54-150

[9] 常育红，姜哲，朱小燕. 基于标记树表示方法的页面结构分析[J]. 计算机工程与应用，2004，40(16)：129-132

[10] 朱明，王庆伟. 半结构化网页中多记录信息的自动抽取方法[J]. 计算机仿真，2005，22(12)：95-97