计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20151176
54(4):832-843,2017

基于任务发生关系的流程模型相似性度量

宋金凤　闻立杰　王建民
(清华大学软件学院　北京　100084)
(632230913@qq.com)

A Similarity Measure for Process Models Based on Task Occurrence Relations

Song Jinfeng, Wen Lijie, and Wang Jianmin
(*School of Software, Tsinghua University, Beijing* 100084)

Abstract　Similarity measures between process models are increasingly important for management, reuse, and analysis of process models in modern enterprises. So far, several approaches have been proposed and behavioral profile (BP) is a good concept to judge the behavioral consistency of process models, which describes the observable relations between tasks. However, all those approaches have their own advantages and disadvantages. Towards the hard problem of behavioral similarity measure between process models, especially to improve the effectiveness of BP, a new method for measuring the behavioral similarity between process models named TOR based on the occurrence relation among tasks is proposed. Based on complete prefix unfolding (CPU) technique of Petri nets, we propose the algorithms for numbering the nodes in a CPU and computing the least common precursors for each pair of nodes. Then we define the three basic occurrence relations between tasks: causal relation, concurrent relation and conflict relation. The algorithm for efficiently computing the relations and the formalism for computing the similarity are also given. TOR can handle both invisible tasks and non-free choice constructs. The experimental results show the effectiveness and efficiency of TOR. Compared with the existing mainstream behavioral similarity algorithms for process models, TOR can satisfy all the five properties that a good similarity algorithm should have.

Key words　similarity measure; occurrence relations of tasks; process model; complete prefix unfolding (CPU); Petri nets

摘　要　针对流程模型行为相似性度量难题,提出了一种基于任务发生关系的流程模型相似性度量TOR.基于Petri网的完全前缀展开理论,提出了节点编号算法以及最近公共前驱计算方法,在此基础上定义了任务间3种基本的发生关系:因果、并行和互斥,并给出这些关系的高效计算方法和模型相似度计算公式.TOR能有效处理不可见任务和非自由选择结构,基于来自企业实际模型的实验证明了TOR具备较好的效果和性能,与已有算法相比,TOR能较好地满足行为相似性算法应具备的性质.

关键词　相似性度量;任务发生关系;流程模型;完全前缀展开;Petri网

中图法分类号　TP315

收稿日期:2015-12-22;**修回日期**:2016-06-02

基金项目:国家自然科学基金项目(61472207,61325008)

This work was supported by the National Natural Science Foundation of China (61472207, 61325008).

通信作者:闻立杰(wenlj@tsinghua.edu.cn)

流程模型对于一个企业来说,具有十分重要的价值,它的作用不仅仅是对企业业务流程的具体刻画,而且有利于企业对业务流程进行分析、验证和优化[1].流程模型的管理包括模型分析、模型检索和模型重用等方面[2].流程模型相似性度量在流程模型管理的各个方面都发挥着非常重要的作用.目前的流程模型相似性算法主要分为3类[2]:1)元素标签映射的相似性度量;2)模型结构的相似性度量;3)行为语义的相似性度量.

元素标签映射的相似性度量,是基于节点的成对标签比较.它是通过定义2个模型的节点标签之间的映射,从而计算出相似性.标签匹配相似度等于匹配的节点数除以总节点数.为了检测标签的相似性,常常运用模式匹配算法[3]和本体匹配算法[4].Dijkman等人[5]在相似性度量方面做出了很多研究,提出了较多的相似性度量算法,最简单的一种叫标签对齐的相似性度量算法.Ehrig等人[6]提出了基于标签语义的相似性度量方法.该类算法思路简单、计算快速,但未对模型的拓扑结构和行为语义加以考虑,导致计算结果不够精准.

结构相似性度量方法,是把模型看作一个图,利用公共子图同构和图编辑距离对模型的相似性进行度量.图编辑距离详细定义了从一个图转换到另一个图所需的最小原子级图操作.Dijkman等人[7]提出了一种结构相似性度量方法,该方法定义了进行每一种编辑操作都必须付出相应代价,通过从一个模型到另一个模型的编辑距离,达到求出相似性的目的.基于上述算法,La Rosa等人[8]又提出了一种算法,结合了图的编辑距离和活动匹配的方法.Li等人[9]提出了一种基于高级变更操作来计算模型相似性的方法,这种高级变更操作可以确保模型的完整性,并且使图的编辑距离具有了高层语义上的特征.该类算法往往无法辨别行为相同或相近而结构有较大差异的模型对儿.

基于行为语义的相似性度量主要从模型的行为语义(如执行序列、任务关系)角度去考虑提取模型的行为关系,进而进行相似性的计算.Weidlich等人[10]提出了行为轮廓(behavioral profile,BP)算法的概念,行为轮廓主要是弱顺序关系、严格顺序关系、互斥关系、交叉关系等一系列关系的统称.在行为轮廓概念定义的基础上,作者提出了基于行为轮廓度量模型相似性的算法,即行为轮廓算法.该算法虽然可以在一定程度上高效提取行为关系,但是其粒度过粗,往往不能区分一些特殊结构,如互斥与循环、不可见任务等.Polyvyanyy等人[11]提出了4C(co-occurrence,conflict,causality,concurrency)算法,该算法提出了一系列关系的定义,包括共现关系、冲突关系、因果关系以及并发关系,统称为4C关系,但由于关系过多且复杂,导致了其不易于计算和理解.Zha等人[12]提出了一种变迁毗邻关系(transition adjacent relations,TAR)算法,但该算法只考虑到了变迁的紧邻关系而忽略了变迁间接影响的关系,因此导致自由选择结构与非自由选择结构可能产生相同的结果.Wang等人[13]提出基于主变迁序列(principal transition sequences,PTS)的相似性计算方法,该算法可以有效处理循环结构,但是在并发分支较多的情况下会导致空间爆炸问题.汪抒浩等人[14]提出了SSDT算法(shortest succession distance between tasks),该算法定义了任务最短跟随距离,并在此基础上定义过程模型的任务最短跟随距离矩阵,将2个过程模型的对应距离矩阵进行同维化后可以进行相似性计算.Dijkman等人[5]提出一种用因果足迹(casual footprints,CF)计算相似性的方法,该算法的主要思路是把过程模型用向量表示,但是由于向量中有过多的冗余信息,因此高维度的向量导致计算非常低效.

针对BP算法和4C算法的不足,本文提出了新的行为相似性度量方法TOR(task occurrence relations),即基于任务发生关系的流程模型相似性算法.首先定义了完全前缀展开中的前驱、公共前驱、最近公共前驱等一系列概念,并且设计了节点编号算法、最近公共前驱求解算法,提出了任务间3种发生关系,即因果、并行和互斥,据此给出了相似性计算方法.TOR算法可以很好地处理不可见任务和非自由选择结构,并且可以有效提取存在于任务间的多关系.

1 预备知识

1.1 Petri网

定义1. Petri网[15].Petri网是三元组$N=(P,T,F)$,其中P和T是不相交的有限集合,P是所有库所的集合,而T是所有变迁的集合,它们之间的连接关系用F表示,$F\subseteq(P\times T)\cup(T\times P)$.Petri网中所有的节点集合为$X=P\cup T$,对于任意节点$x\in X$,$x$的前置集合为$\bullet x=\{y\in X|(y,x)\in F\}$,同理$x$的后置集合为$x\bullet=\{y\in X|(x,y)\in F\}$.

定义2. Petri网系统[15].二元组$S=(N,M_0)$

是 Petri 网系统,其中 $N=(P,T,F)$为 Petri 网,M_0 是 Petri 网的初始标识,$M_0:P\to N$,N 是自然数集合.

定义 3. 就绪[15]. 给定 Petri 网系统 $S=(N,M_0)$,对于任意变迁 $t\in T$,M 是从 M_0 可达的标识,当 $\forall p\in {}^\bullet t$ 满足 $M(p)\geqslant 1$ 时,则称 t 就绪,记为 $(N,M)[t\rangle$. 若 S 的任意可达标识 M 和库所 p 都满足 $M(p)\leqslant n$(n 为任意正整数),则称 S 是有界的.

定义 4. 触发[15]. 给定 Petri 网系统 $S=(N,M_0)$,M 是从 M_0 可达的标识,若 $t\in T$ 且 $(N,M)[t\rangle$,则 t 可被触发,并得到新的标识 $M'=M-{}^\bullet t+t^\bullet$.

1.2 完全前缀展开技术

McMillan 等人[16]最先提出了一种新技术来避免 Petri 网系统验证中无限状态造成的空间爆炸问题. 他首先提出一种网展开的概念,即完全有限前缀,之后基于分支进程这一概念,对该理论进行了进一步阐述[17]. 该算法的创新之处在于:通过记录已经存在的标记状态,提出了截断事件的概念(即如果一个标记状态已经存在过,那么就对其之后的部分进行截断),从而避免同一状态重复出现多次的现象,确保了展开结构的简洁性. 因此,该算法有着较高的效率和比较小的展开规模,在解决 Petri 网状态空间爆炸问题方面有着比较出色的表现. Esparza 等人[18]对 McMillan 等人[16-17]的算法进行了改进,提出了一种新的规模更小的展开方法,即完全前缀展开.

下面对该完全前缀展开(complete prefix unfolding,CPU)技术的一些基本概念进行简要的介绍,更多相关概念和示例详见文献[18].

定义 5. 出现网[18]. 出现网为一类特殊的 Petri 网 $O=(C,E,G)$,当且仅当满足如下条件:

1) O 是无环的,C 为条件(即库所)的集合,E 为事件(即变迁)的集合,边集 $G\subseteq(C\times E)\cup(E\times C)$;

2) $\forall c\in C$,满足 $|{}^\bullet c|\leqslant 1$,即任意条件的输入都小于等于 1;

3) 对于所有 $x\in C\cup E$,都有 $\neg(x\sharp x)$,即所有的元素都不能是自冲突的;

4) $\forall x\in C\cup E$ 满足 $\{y\in C\cup E\mid y<x\}$ 必须是有限的.

在给定出现网中,因果关系 $x<y$ 表示 x 与 y 之间为偏序关系,即存在从 x 到 y 的路径,冲突关系 $x\sharp y$ 表示二者所在的不同路径开始于某公共条件或库所 c,且从 c 到 x 的路径与从 c 到 y 的路径不相交.

定义 6. 分支进程[18]. 对于网系统 $S=(N,M_0)$,其中 $N=(P,T,F)$,出现网为 $O=(C,E,G)$,则 $\pi=(O,h)$为 S 的一个分支进程,当且仅当满足如下条件:

1) h 为一个映射:$C\cup E\mapsto P\cup T$,同时 $h(C)\subseteq P$ 且 $h(E)\subseteq T$;

2) 对于所有 $e\in E$,${}^\bullet e$ 和 ${}^\bullet h(e)$ 之间是双射,$e^\bullet$ 和 $h(e)^\bullet$ 之间是双射,即 h 保留了变迁的外延;

3) h 使得 $Min(O)$ 和 M_0 之间为双射,$Min(O)$ 表示根据偏序关系 $<$ 得到的 $C\cup E$ 中的所有最小元素集合;

4) 对于所有 $e,f\in E$,当 $h(e)=h(f)$ 且 ${}^\bullet e={}^\bullet f$ 时,则必须满足 $e=f$.

定义 7. 配置[18]. $S=(N,M_0)$是 Petri 网系统,其中 $N=(P,T,F)$,$\pi=(O,h)$是 S 的一个分支进程,其中 $O=(C,E,G)$,分支进程 π 的某个配置 C' 是一组事件,满足如下关系:

1) $\forall e_1\in C',e_2<e_1\Rightarrow e_2\in C'$,即 C' 是因果关系的事件组成的闭包;

2) $\forall e_1,e_2\in C'$ 满足 $\neg(e_1\sharp e_2)$,即 e_1 和 e_2 不冲突.

其中,任意事件 $e\in E$ 的局部配置指的是,满足条件 $x<e\vee x=e$ 的所有事件 x 的集合,记作 $[e]$.

定义 8. 割集[18]. 对于 $\pi=(O,h)$的某有穷配置 C',$Cut(C')=(Min(O)\cup C'^\bullet)\setminus {}^\bullet C'$ 被称为一个割集.

定义 9. 充分关系[19]. 充分关系是在局部配置上的严格偏序关系,对于任意 2 个事件 $e,f\in E$,如果有 $[e]\subset[f]$,那么 $[e]<[f]$,且 $<$ 通过有穷扩展得以保持(即如果 $[e]<[f]$ 且 $Mark([e])=Mark([f])$,则对于 $[e]$ 的所有有穷扩展 $[e]\oplus E$,存在同构变换 I_1^2,使得 $[e]\oplus E<[f]\oplus I_1^2(E)$),则意味着 e 和 f 之间为充分关系.

其中,$Mark(C)$表示从原网的初始标识开始,依次触发配置 C 中的每个事件,最终得到可达标识.

定义 10. 截断事件、截断条件[18]. 对于 $\pi=(O,h)$中的一个事件 $e\in E$,如果有一个对应事件 $f\in E$,满足 $Mark([e])=Mark([f])$,并且 $[f]<[e]$,那么 e 就是截断事件,f 是其对应事件,记作 $f=corr(e)$. 此时,$e^\bullet$ 被称作截断条件集合. 通常来说,对于任意截断条件 $c\in e^\bullet$,满足 $c^\bullet=\varnothing$,即其后的事件均被截断了.

定义 11. 完全前缀展开[18]. 对于有界的网系统 $S=(N,M_0)$,其分支进程 $\pi=(O,h)$是 S 基于截断技术得到的前缀展开,则 π 是完全的当且仅当对于

S 中的每个可达标识 M,都存在 π 的一个配置 C,使得:

1) $Mark(C)=M$,即 M 在 π 中得以表达;

2) 对于 M 下就绪的每个变迁 t,都存在一个事件 e,满足 $e\notin C$ 且 $h(e)=t$ 且 $C\cup\{e\}$构成一个配置.

例 1. 如图 1(a)所示的 Petri 网,图 1(b)为其完全前缀展开,也是该网的一个分支进程,其中:局部配置$[A]=\{A\}$,$[C]=\{A,C\}$,$[E]=\{A,B,C,E\}$,而割集 $Cut([A])=\{P_1,P_2\}$,$Cut([C])=\{P_{41}\}$,$Cut([E])=\{P_5\}$;针对 C 与 D 组成的互斥结构,事件 D 为截断事件,其对应事件 $corr(D)=C$,而 P_{42} 为截断事件,其对应条件为 P_{41};针对 H 和 I 构成的循环结构,事件 I 为截断事件,其对应事件 $corr(I)=F$,而 P_{72} 为截断条件,P_{71} 为对应条件,这是由于 $Mark([F])=Mark([I])$且$[F]=\{A,B,C,E,F\}\prec[I]=\{A,B,C,E,F,H,I\}$.

2 基于任务间发生关系的模型相似性度量算法

基于任务间发生关系的模型相似性度量算法 TOR 的主要步骤如下:

1) 基于完全前缀展开. 首先把给定的 Petri 网模型进行完全前缀展开,这样可以保持流程模型的所有标识及任务间发生关系,便于后续提取流程模型的行为特征.

2) 节点遍历编号. 逐层广度优先遍历得到的完全前缀展开,对其节点按照遍历的层次编号,并存储节点和其对应的编号.

3) 求出任务发生关系. 根据最近公共前驱算法求出每 2 个任务的最近公共前驱并作相应的存储,从而进一步求出任务发生关系. 根据求出的最近公共前驱以及特殊结构的处理方式,确定任务之间的发生关系.

4) 模型相似性计算. 在相应的任务间关系集合的基础上,通过关系集合之间的加权相似性计算出模型之间的相似性.

2.1 任务发生关系相关定义

本节主要介绍围绕任务发生关系的一系列定义,包括完全前缀展开图中任意 2 个节点的前驱、公共前驱、最近公共前驱以及任务发生关系.

定义 12. 路径. $S=(N,M_0)$是有界系统,$\pi=(O,h)$是其包含截断事件的 CPU,其中 $N=(P,T,F)$,$O=(C,E,G)$. $a,b\in C\cup E$,有一条路径从 a 到 b,即存在一条从 a 到 b 的有向通路.

例 2. 在图 1(b)中,A 经过 P_1 到 B,则 A 到 B 可达,即存在一条从 A 到 B 的路径.

为了计算任务间发生的关系,需要对 CPU 节点的前驱、公共前驱以及最近公共前驱进行定义.

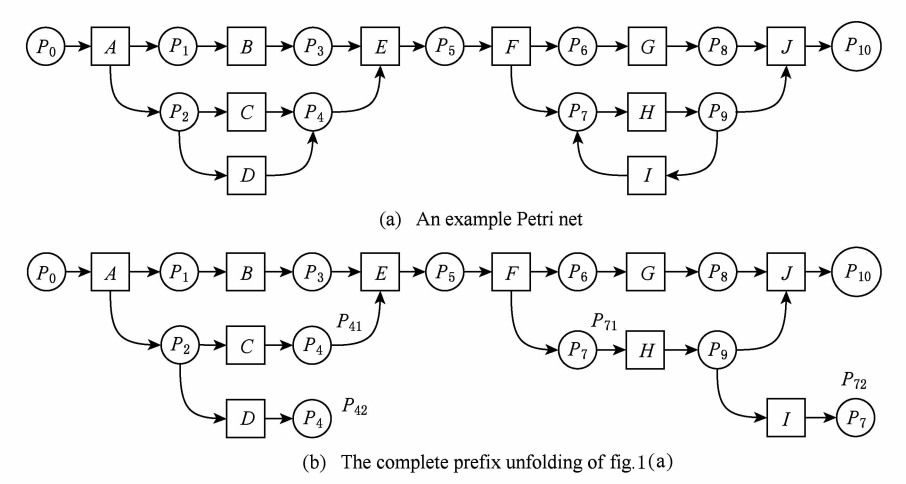


Fig. 1 An example for task occurrence relations

图 1 任务发生关系示例

定义 13. 前驱. $S=(N,M_0)$是有界系统,$\pi=(O,h)$是其包含截断事件的 CPU,其中 $N=(P,T,F)$,$O=(C,E,G)$. 从 p 到 e 存在一条路径,其中$p\in C\cup E$,$e\in E$,那么 p 是 e 的一个前驱,记为 $p\in pre(e)$,$pre(e)$表示 e 的所有前驱节点集合. 特别地,有 $e\in pre(e)$.

对于截断存在的情况,如果有一条路径从 c 到 e,并且 $c,c'\in C$,$e\in E$,如果有 $c=corr(c')$,那么认为有一条路径从 c'到 e,且所有 c'的前驱节点也是 e 的前驱,即 $\forall p\in pre(c')$,有 $p\in pre(e)$.

例 3. 在图 1(b)中,有一条路径从 B 到 E,B 是 E 的前驱. 同样地,有一条路径从 C 到 E,那么 C 也

是 E 的前驱.特别地,D 虽然没有直接到 E 的路径,但是可以通过截断条件 P_{42} 跳转到 P_{41} 后使得从 D 到 E 有路径相连,所以 D 也是 E 的前驱.同理,有一条路径从 G 到 J,G 是 J 的前驱;有一条路径从 H 到 J,H 也是 J 的前驱.虽然没有直接从 I 到 J 的路径,但是,可以通过截断条件 P_{72} 跳转到 P_{71} 后使得从 I 到 J 有路径相连,因此,I 也是 J 的前驱.

定义 14. 公共前驱. $S=(N,M_0)$ 是有界系统,$\pi=(O,h)$ 是其包含截断事件的 CPU,其中 $N=(P,T,F)$,$O=(C,E,G)$. 对任意 $e_1,e_2\in E$,如果存在 $cp\in C\cup E$,并且有 $cp\in pre(e_1)\cap pre(e_2)$,那么 cp 就是 e_1,e_2 的公共前驱,记作 $cp\in cmPre(e_1,e_2)$.

定义 15. 最近公共前驱. $S=(N,M_0)$ 是有界系统,$\pi=(O,h)$ 是其包含截断事件的 CPU,其中 $N=(P,T,F)$,$O=(C,E,G)$. 对任意 $e_1,e_2\in E$ 和 $lcp\in cmPre(e_1,e_2)$,若不存在 $lcp'\in cmPre(e_1,e_2)$,满足 $lcp'\neq lcp$ 而且 $lcp\in pre(lcp')$,那么 lcp 是 e_1 和 e_2 的最近公共前驱,记为 $lcp\in lcPre(e_1,e_2)$.

例 4. 在图 1(b)中,节点 A 既是 B 的前驱,也是 C 的前驱,则 A 是 B 和 C 的公共前驱.虽然 A 也是 C 和 D 的公共前驱,但 P_2 才是 C 和 D 的最近公共前驱.

定义 16. 因果关系. $S=(N,M_0)$ 是有界系统,$\pi=(O,h)$ 是其包含截断事件的 CPU,其中 $N=(P,T,F)$,$O=(C,E,G)$. $e_1,e_2\in E$,$t_1,t_2\in T$,$h(e_1)=t_1$ 且 $h(e_2)=t_2$,则 t_1 和 t_2 存在因果关系,当且仅当 $e_1\in lcPre(e_1,e_2)$(记作 $t_1\rightarrow t_2$ 或 $t_2\leftarrow t_1$,即逆序关系)或者 $e_2\in lcPre(e_1,e_2)$(记作 $t_2\rightarrow t_1$ 或 $t_1\leftarrow t_2$).

定义 17. 并行关系. $S=(N,M_0)$ 是有界系统,$\pi=(O,h)$ 是其包含截断事件的 CPU,其中 $N=(P,T,F)$,$O=(C,E,G)$. $e_1,e_2\in E$,$t_1,t_2\in T$,$h(e_1)=t_1$ 且 $h(e_2)=t_2$,则 t_1 和 t_2 存在并行关系,当且仅当存在 $e\in E$,使得 $e\neq e_1$ 且 $e\neq e_2$ 且 $e\in lcPre(e_1,e_2)$,记作 $t_1\parallel t_2$ 或者 $t_2\parallel t_1$.

定义 18. 互斥关系. $\pi=(O,h)$ 是其包含截断事件的 CPU,其中 $N=(P,T,F)$,$O=(C,E,G)$. $e_1,e_2\in E$,$t_1,t_2\in T$,$h(e_1)=t_1$ 且 $h(e_2)=t_2$,则 t_1 和 t_2 存在互斥关系,当且仅当存在 $c\in C$,使得 $c\in lcPre(e_1,e_2)$,记作 $t_1\sharp t_2$ 或者 $t_2\sharp t_1$.

例 5. 在图 1(b)中,A 和 E 存在因果关系,B 和 C 存在并行关系,C 和 D 存在互斥关系;而对于 I 和 J,由于 $lcPre(I,J)=\{P_9\}$,二者之间存在明显的互斥关系.

定义 19. 任务发生关系. $S=(N,M_0)$ 是有界系统,$\pi=(O,h)$ 是其包含截断事件的 CPU,任务发生关系集合 $TOR=\{\leftarrow,\rightarrow,\parallel,\sharp\}$,统称为 S 的任务发生关系.

以上定义了与任务发生关系有关的一系列定义,基于上述定义才可以进行后续的模型相似性计算.

2.2 任务发生关系计算

本节介绍任务发生关系的计算方法,包括节点编号算法、最近公共前驱算法以及任务发生关系的确定.

节点编号算法(其伪代码如算法 1 所示)主要是通过遍历 CPU,确定每个节点的层级编号,是后续高效计算任意 2 节点最近公共前驱的基础.

算法 1. 节点编号算法 *generateLevel*.

输入:有界系统 $S=(N,M_0)$,$\pi=(O,h)$ 是其包含截断事件的 CPU,其中 $N=(P,T,F)$,$O=(C,E,G)$;

输出:$Map(node,level)$. /* 每个节点 $node$ 及其对应编号 $level$ 的散列表 Map */

① 查找 π 中所有源节点并放入队列 *startNodes*;
② for each $v\in startNodes$ do
③ $Map.put(v,1)$;
④ end for
⑤ while $startNodes.size()>0$ do
⑥ 从*startNodes* 中取出第 1 个元素,其编号为 $level$;
⑦ for each $n\in node\bullet$ do
⑧ if $n\notin Map$ or $Map.get(n)<level+1$ then
⑨ $Map.put(n,level+1)$;
⑩ add n to *startNodes*;
⑪ end if
⑫ end for
⑬ if 节点 n 是互斥结构引发的截断条件且 ($corr(n)\notin Map$ or $Map.get(corr(n))<level$) then
⑭ $Map.put(corr(n),level)$;
⑮ add $corr(n)$ to *startNodes*;
⑯ end if
⑰ end while

节点编号算法的步骤简介如下:1)取出全部首节点(一般有且仅有 1 个),编号为第 1 层;2)按照广度优先的原则,不断取出同一层次的后继节点,按照同一层编号相同的原则进行编号.同时,由于节点可从不同路径可达,因此必要时需要更新某节点及其

后继节点的编号,取较大的编号作为该节点最终的编号;如果遇到循环节点,则不进行编号更新.遍历的同时,把节点和其对应的编号信息存储在散列表 Map 中,当所有节点遍历完成后,算法结束.针对图 2(a)所示 Petri 网的 CPU 图应用节点编号算法,所得结果如图 2(b)所示.

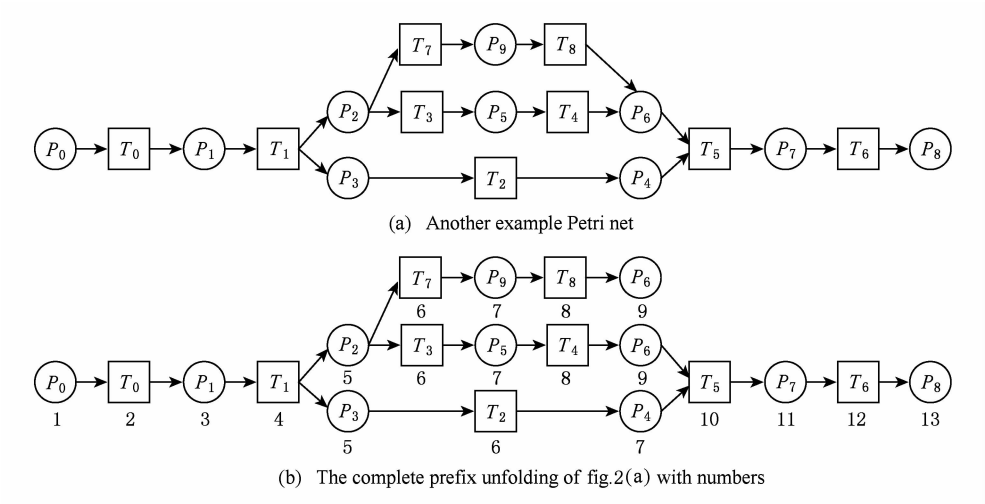


Fig. 2 An example for Algorithm 2

图 2 算法 2 步骤举例

最近公共前驱可以在一定程度上反映任务之间的发生关系,函数 $computeLCP$ 主要用于完成 CPU 中 2 节点最近公共前驱的计算,其伪代码如算法 2 所示,其主要思路如下:

1) 把事件 e_1 和事件 e_2 分别放入事件数组 $array_1$ 和 $array_2$ 中(行①~②).

2) 遍历 $array_1$ 求出编号最大的节点,记这个节点为 $node_1$,记最大编号为 max_1;同理,遍历 $array_2$ 求出编号最大的节点 $node_2$,记最大编号为 max_2(行④~⑦).

3) 比较 max_1 和 max_2.若 $max_1 > max_2$,则 $node_1$ 向前回溯一步;否则 $node_2$ 向前回溯一步(行⑧~⑪).

4) 不断重复思路 2 和思路 3,直到找到公共前驱节点为止(行③~⑫).

5) 返回公共前驱节点,这些公共节点就是 e_1,e_2 的最近公共前驱(行⑬).

算法 2. 最近公共前驱算法 $computeLCP$.

输入:有界系统 $S=(N,M_0)$,$\pi=(O,h)$ 是其包含截断事件的 CPU,其中 $N=(P,T,F)$,$O=(C,E,G)$;$e_1\in E$,$e_2\in E$;存储 CPU 节点编号结果的散列表 Map;

输出:最近公共前驱数组 $lcpSet$.

① add e_1 to $array_1$;

② add e_2 to $array_2$;

③ while $array_1 \cap array_2 = \varnothing$ do

④ get $node_1$ from $array_1$ with the max number;

⑤ $max_1 = Map.get(node_1)$;

⑥ get $node_2$ from $array_2$ with the max number;

⑦ $max_2 = Map.get(node_2)$;

⑧ if $max_1 > max_2$ then

⑨ $stepBackward(node_1, array_1)$;

⑩ else

⑪ $stepBackward(node_2, array_2)$;

⑫ end if

⑬ end while

⑭ $lcpSet = array_1 \cap array_2$.

回溯一步算法 $stepBackward$(其伪代码如算法 3 所示)的主要思想是从 $startNode$ 节点开始,向前回溯一步,同时更新数组 $array$,把 $startNode$ 的所有前驱节点加入数组 $array$ 中,并把 $startNode$ 节点从数组中移除.

算法 3. 回溯一步算法 $stepBackward$.

输入:有界系统 $S=(N,M_0)$,$\pi=(O,h)$ 是其包含截断事件的 CPU,其中 $N=(P,T,F)$,$O=(C,E,G)$;$startNode \in C \cup E$;存储待遍历节点的数组 $array$.

① for each $n \in {}^\bullet startNode$ do

② add n to $array$;

③ end for

④ if $startNode \in C$ then

⑤ for each $c \in C$ do

⑥ if $startNode=corr(c)$ then
⑦ add c to $array$;
⑧ end if
⑨ end for
⑩ end if
⑪ remove $startNode$ from $array$.

下面用一个具体的流程模型来说明2节点最近公共前驱的求解过程.

例6. 以图2(a)中的流程模型为例,求解其CPU中事件T_2和T_8的最近公共前驱的计算过程如下:

1) 将该模型进行完全前缀展开,并用编号算法对所有进行编号,得到的CPU如图2(b)所示;

2) 把T_8放入$array_1$中,把T_2放入$array_2$中,此时$array_1$中节点的最大编号$max_1=8$,$array_2$中节点的最大编号$max_2=6$,由于$max_1>max_2$,应该从T_8向前回溯一步,调用函数$stepBackward$,此时$array_1$更新为$\{P_9\}$,$array_2$不变,仍为$\{T_2\}$;

3) 此时$max_1=7$,$max_2=6$,由于$max_1>max_2$,应该从P_9向前回溯一步,调用函数$stepBackward$,此时$array_1$更新为$\{T_7\}$;

4) 此时$max_1=6$,$max_2=6$,由于$max_1=max_2$,T_2向前回溯一步,调用函数$stepBackward$,此时$array_2$更新为$\{P_3\}$;

5) 此时$max_1=6$,$max_2=5$,由于$max_1>max_2$,应该从T_7向前回溯一步,调用函数$stepBackward$,此时$array_1$更新为$\{P_2\}$;

6) 此时$max_1=5$,$max_2=5$,由于$max_1=max_2$,应该从P_3向前回溯一步,调用函数$stepBackward$,此时$array_2$更新为$\{T_1\}$;

7) 此时$max_1=5$,$max_2=4$,由于$max_1>max_2$,应该从P_2向前回溯一步,调用函数$stepBackward$,此时$array_1$更新为$\{T_1\}$;

8) 到此为止,已经找到最近公共前驱集合$\{T_1\}$,算法结束.

根据求出的事件间最近公共前驱集合以及之前对于任务发生关系的定义,可以确定任意2个给定任务间的发生关系,对应算法的伪代码如算法4所示.

算法4. 任务发生关系的确定算法$computeTOR$.

输入:有界系统$S=(N,M_0)$,$\pi=(O,h)$是其包含截断事件的CPU,其中$N=(P,T,F)$,$O=(C,E,G)$;$t_1,t_2\in T$;

输出:任务发生关系集合TOR.

① for each $e_1\in E\wedge h(e_1)=t_1$ do
② for each $e_2\in E\wedge h(e_2)=t_2$ do
③ $lcp=computeLCP(e_1,e_2).get(0)$;
④ if $lcp=e_1$ then
⑤ $TOR.add(t_1,t_2,\rightarrow)$;
⑥ else if $lcp=e_2$ then
⑦ $TOR.add(t_1,t_2,\leftarrow)$;
⑧ else if $lcp\in E$ then
⑨ $TOR.add(t_1,t_2,\|)$;
⑩ $TOR.add(t_2,t_1,\|)$;
⑪ else if $lcp\in C$ then
⑫ $TOR.add(t_1,t_2,\#)$;
⑬ $TOR.add(t_2,t_1,\#)$;
⑭ end if
⑮ end for
⑯ end for

通过该算法,可以基于CPU中任意2个事件间的最近公共前驱,确定流程模型中每对儿任务之间的发生关系,进一步可以求出流程模型的任务发生关系矩阵.

2.3 流程模型相似性计算

通过2.1节和2.2节介绍的算法和定义,已经可以确定模型的关系矩阵,基于关系矩阵可以计算过程模型的相似性.相似性计算主要思路为:首先分别介绍并行关系、互斥关系以及因果关系的权重和相似度;然后用权重乘以对应的关系相似度,从而得出整个模型的相似性.

并行关系、互斥关系以及因果关系的权重分别用如下公式计算,其中P,Q为给定的2个流程模型,$P_{\|}$,$P_{\#}$,$P_{\rightarrow}$,$P_{\leftarrow}$分别为P的并行关系、互斥关系、因果关系和逆序关系集合.对于模型Q,亦有对应的关系集合.

w_1为并行关系的权重,其计算为

$$w_1=\frac{|P_{\|}\cup Q_{\|}|}{|P_{\|}\cup Q_{\|}|+|P_{\#}\cup Q_{\#}|+|(P_{\rightarrow}\cup P_{\leftarrow})\cup(Q_{\rightarrow}\cup Q_{\leftarrow})|}. \tag{1}$$

w_2为互斥关系的权重,其计算为

$$w_2=\frac{|P_{\#}\cup Q_{\#}|}{|P_{\|}\cup Q_{\|}|+|P_{\#}\cup Q_{\#}|+|(P_{\rightarrow}\cup P_{\leftarrow})\cup(Q_{\rightarrow}\cup Q_{\leftarrow})|}. \tag{2}$$

w_3为因果关系的权重,其计算为

$$w_3=\frac{|(P_{\rightarrow}\cup P_{\leftarrow})\cup(Q_{\rightarrow}\cup Q_{\leftarrow})|}{|P_{\parallel}\cup Q_{\parallel}|+|P_{\#}\cup Q_{\#}|+|(P_{\rightarrow}\cup P_{\leftarrow})\cup(Q_{\rightarrow}\cup Q_{\leftarrow})|}. \tag{3}$$

在上述权值计算公式中,借鉴了 Weidlich 等人[10]在 BP 算法中的思路,直接采用各类二元关系个数在所有二元关系总数中的比重,认为并行关系、互斥关系和因果关系具有相同的重要性,这些权值支持用户自定义.

根据 Jaccard 系数,并行关系、互斥关系以及因果关系的相似性分别用如下公式计算. 本文假定前任务名称、发生关系、后任务名称分别相同,为判定关系集合中元素相同的标准,如 $T_1\#T_2\neq T_1\#T_3$.

对于并行关系的相似性,用 P 模型和 Q 模型所有并行关系的交集元素数除以 P 模型和 Q 模型所有并行关系的并集元素数:

$$sim_{\parallel}(P,Q)=\frac{|P_{\parallel}\cap Q_{\parallel}|}{|P_{\parallel}\cup Q_{\parallel}|}. \tag{4}$$

对于互斥关系的相似性,用 P 模型和 Q 模型所有互斥关系的交集元素数除以 P 模型和 Q 模型所有互斥关系的并集元素数:

$$sim_{\#}(P,Q)=\frac{|P_{\#}\cap Q_{\#}|}{|P_{\#}\cup Q_{\#}|}. \tag{5}$$

对于因果关系的相似性,用 P 模型和 Q 模型所有因果关系的交集元素数除以 P 模型和 Q 模型所有因果关系的并集元素数:

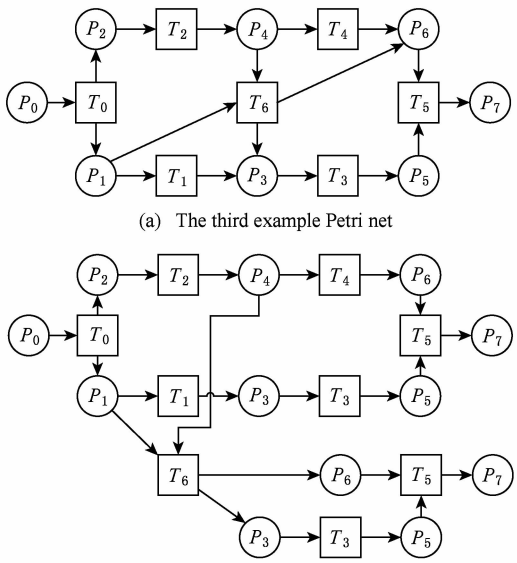
$$sim_{\rightarrow}(P,Q)=\frac{|(P_{\rightarrow}\cup P_{\leftarrow})\cap(Q_{\rightarrow}\cup Q_{\leftarrow})|}{|(P_{\rightarrow}\cup P_{\leftarrow})\cup(Q_{\rightarrow}\cup Q_{\leftarrow})|}. \tag{6}$$

2 个模型 P 和 Q 的相似性为并行关系的权重乘以并行关系相似性、互斥关系的权重乘以互斥关系的相似性、因果关系的相似性乘以因果关系的权重这三者的累加和,其计算为

$$sim(P,Q)=w_1\times sim_{\parallel}(P,Q)+w_2\times sim_{\#}(P,Q)+w_3\times sim_{\rightarrow}(P,Q). \tag{7}$$

2.4 任务间多发生关系处理

图 3 展示了包含多关系的典型实例,同时该例也体现了 TOR 算法对非自由选择结构的有效处理. 图 3(a)为 Petri 网表达的原模型,图 3(b)是其对应的 CPU. 在一个分支进程里,T_2 和 T_3 是并行关系,在另一分支进程里面,T_2 和 T_3 是因果关系,这 2 种关系都是有效的. 在算法的设计中已充分考虑了不同分支进程的情况,因此,在关系的提取过程中,这 2 种关系都会被提取出来,对应的任务发生关系矩阵如表 1 所示.



(a) The third example Petri net

(b) The complete prefix unfolding of fig.3(a)

Fig. 3 An example for multiple occurrence relations between tasks

图 3 任务间多发生关系实例

Table 1 TOR Matrix of the Process Model in Fig. 3

表 1 图 3 中模型的任务发生关系矩阵

Task	T_0	T_1	T_2	T_3	T_4	T_5	T_6
T_0	#	→	→	→	→	→	→
T_1	←	#	‖	→	‖	→	#
T_2	←	‖	#	‖,→	→	→	→
T_3	←	←	‖,←	#	‖	→	←
T_4	←	‖	←	‖	#	→	#
T_5	←	←	←	←	←	#	←
T_6	←	#	←	→	#	→	#

Notes: →means causality; ←means inverse causality; #means conflict; ‖means concurrency.

2.5 不可见任务处理

在实际业务过程模型中会出现一种特殊的任务,称作不可见任务[19],即空标签任务,这种任务的存在会影响模型的行为. 然而,现有的模型相似性算法大多忽略了不可见任务的处理,本文给出其具体处理方法. 主要处理 3 种类型的不可见任务,分别是 SKIP 类型、REDO 类型以及 SWITCH 类型. SKIP 类型不可见任务用于跳过某些任务,如图 4 所示;REDO 类型用于重复执行某些任务,如图 5 所示;SWITCH 类型不可见任务,用于在过程模型的不同可选分支间进行执行顺序切换,如图 6 所示. 关于不可见任务的详细分类,读者可参阅文献[19].

对于 SKIP 类型的不可见任务,可以看出在图 4 所示模型中,不可见任务(称其为 $Inv1$)跳过了任务

B,也就是该不可见任务的存在,导致了 B 被跳过.换个角度来说,该不可见任务和 B 是互斥关系,二者不能同时执行.在处理这一类不可见任务的时候,只需要考虑该不可见任务与其跳过范围内的可见任务的互斥关系即可.确定这个范围,就需要通过该不可见任务的输入输出边来确定其作用范围.也就是说,在 SKIP 类型的不可见任务处理时,不可见任务只参与互斥关系的计算,在图 4 中体现为 $B \# Inv1$.该模型的关系矩阵如表 2 所示.

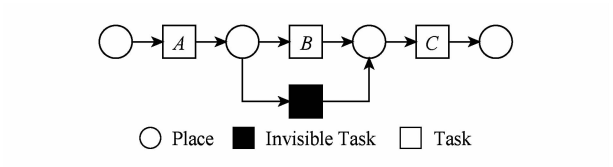


Fig. 4 An invisible task of SKIP type

图 4 SKIP 类型不可见任务

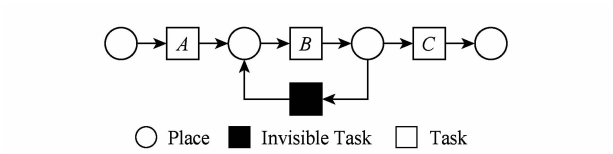


Fig. 5 An invisible task of REDO type

图 5 REDO 类型不可见任务

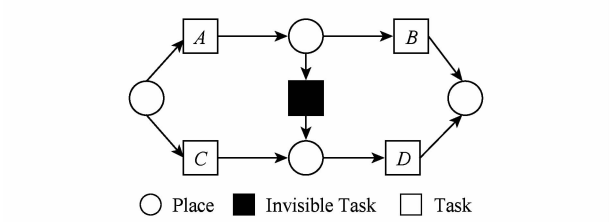


Fig. 6 An invisible task of SWITCH type

图 6 SWITCH 类型不可见任务

Table 2 TOR Matrix Involving an Invisible Task of SKIP Type

表 2 SKIP 类型不可见任务的任务发生关系矩阵

Task	A	B	C	$Inv1$
A	#	→	→	
B	←	#	→	#
C	←	←	#	
$Inv1$		#		

Notes: →means causality; ←means inverse causality; #means conflict.

对于 REDO 类型的不可见任务,需要考虑任务间因果关系的变化.可以看出在图 5 所示模型中,不可见任务(称其为 $Inv2$)和任务 B 构成了循环结构,导致了任务 B 可以重复执行.在处理这类不可见任务的时候,只需要考虑该不可见任务作用范围内可见任务的因果关系即可.虽然有 $Inv2 \rightarrow B$ 和 $B \rightarrow Inv2$,但是最终填写矩阵的时候不考虑这 2 个依赖关系,而只考虑 $B \rightarrow B$.确定不可见任务的作用范围时,同样需要借助其输入输出边.将这个范围内的可见任务彼此之间的关系更新为正确的因果关系即可.该模型的关系矩阵如表 3 所示:

Table 3 TOR Matrix Involving an Invisible Task of REDO Type

表 3 REDO 类型不可见任务的任务发生关系矩阵

Task	A	B	C	$Inv2$
A	#	→	→	
B	←	→	→	
C	←	←	#	
$Inv2$				

Notes: →means causality; ←means inverse causality; #means conflict.

对于 SWITCH 类型的不可见任务,可以看出在图 6 所示模型中,不可见任务(称之为 $Inv3$)导致了 A 和 D 之间多了一条可达路径.这种情况下,导致了 A 和 D 因果关系的产生.而不可见任务本身是不参与因果、并行和互斥 3 种关系中计算的.而该不可见任务的存在,使 A 和 D 之间多了一种因果关系.确定该不可见任务的作用范围,只需检测其导致了哪些任务从不可达变为可达(即因果关系).该模型的关系矩阵如表 4 所示:

Table 4 TOR Matrix Involving an Invisible Task of SWITCH Type

表 4 SWITCH 类型不可见任务的任务发生关系矩阵

Task	A	B	C	D	$Inv3$
A	#	→	#	→	
B	←	#	#	#	
C	#	#	#	→	
D	←	#	←	#	
$Inv3$					

Notes: →means causality; ←means inverse causality; #means conflict.

以上是对不可见任务造成影响的处理,通过这种方式可以很好地计算在不可见任务存在情况下的模型相似性,从而使结果更加合理和全面.

3 实验设计与分析

本文的实验环境如下:MacBook Pro, Intel Dual Core i5 CPU@2.6 GHz, 8 GB DDR3@1 600 MHz,操作系统为 OS X 10.8.3, Java Development Kit 1.8, Java 虚拟机最大内存设置为 2 GB.

实验所涉及的工业数据集主要包括 3 个:

1) 唐山轨道客车有限责任公司数据集(TC).

本文实验的工业数据集包含从 TC 收集的 91 个流程模型. TC 是中国第 1 家轨道交通制造企业,具有 100 多年的悠久历史,具有着超强的创新能力和研发能力,从动车组到中低速客车,从城轨车到特种车,有着完备的生产制造技术平台,其设备通用度高并且注重流程规范.

2) 东方锅炉股份有限公司数据集(DG). 在本文实验的工业数据集包含从 DG 收集的 82 个流程模型. DG 是国资委下属中国东方电气集团子公司下属公司,是一个热力发电系统设备、核能系统设备、煤气化设备等于一体的制造和服务供销商,拥有锅炉、核能发电、焊接、环境保护、材料、热强、自动化控制等科研机构.

3) SAP 参考过程模型数据集(SAP). 本文实验的工业数据集包含从 SAP 收集的流程 381 个模型. SAP 是全球最大的企业资源规划和智能化解决方案的提供商,其业务包括高科技企业资源规划、消费品企业资源规划、零售企业资源规划、医疗企业资源规划、金融企业资源规划、公共事业企业资源规划、化工企业资源规划等.

表 5 列出了上述 3 个工业数据集的基本结构特征,可以看到每个模型集中的平均/最大变迁数、平均/最大库所数和平均/最大弧数.

Table 5 The Basic Structural Features of Three Industrial Datasets

表 5 3 个工业过程模型集的基本结构特征

Dataset	Size	Average			Maximum		
		Transitions	Places	Arcs	Transitions	Places	Arcs
TC	91	12.97	11.46	26.30	39	32	80
DG	82	9.18	9.17	19.11	34	33	70
SAP	381	6.74	8.70	15.99	52	62	137

下面通过对比 TAR 算法、PTS 算法、BP 算法、CF 算法、SSDT 算法及 TOR 算法在 TC,DG,SAP 这 3 个工业数据集上的运行时间,来评估 TOR 算法在实际工业数据集上的性能表现,具体实验结果如表 6 所示:

Table 6 Time Costs of Different Algorithms on the Three Industrial Datasets

表 6 各个算法在不同工业数据集的时间消耗 ms

Dataset	TAR	PTS	BP	CF	SSDT	TOR
TC	1.85	0.88	2.79	1049.33	2.12	1.92
DG	1.21	1.27	1.56	3403.11	1.33	1.21
SAP	0.37	0.12	0.43	222.75	0.39	0.35

可以看出 CF 算法的时间复杂度较高,时间消耗远远高于其他算法;TAR 算法、PTS 算法、BP 算法、SSDT 算法、TOR 算法基本在一个数量级上;TOR 算法略优于 BP 算法、SSDT 算法,与 TAR 算法持平,略慢于 PTS 算法. 因此,TOR 算法在实际工业数据集上运行性能表现良好.

汪抒浩等人[14]给出了流程模型行为相似性算法应当满足的 5 个性质,这些性质一定程度上可以作为评估不同相似性算法的依据. 利用这 5 个性质比较 TOR 算法与主流模型行为相似性算法,结果如表 7 所示:

Table 7 The Satisfaction of Different Algorithms on the Five Properties that a Good Similarity Measure Should Have

表 7 各个算法针对流程模型相似度的 5 个性质满足情况

Properties	TAR	PTS	SSDT	BP	CF	TOR
Sequential structure drift invariance	×	√	√	√	√	√
Negative correlation by loop length	×	×	√	√	×	√
Unrelated task regression	√	×	√	√	√	√
Conflict structure drift invariance	×	√	√	√	×	√
Negative correlation by conflict span	×	√	√	×	×	√

Notes: √ means satisfaction; × means unsatisfaction.

可以看出,TAR 算法只满足一条性质,即互斥无关递减性;PTS 算法满足 3 条性质,不满足循环长度负相关性和互斥无关递减性;BP 算法满足 4 条性质,不满足互斥长度负相关性;而 CF 算法只满足 2 条性质;只有 SSDT 算法和 TOR 算法满足全部 5 条性质. 由此可以看出,TOR 算法的表现非常好,SSDT 算法虽然也能满足全部 5 条性质,但其性能上表现得有所欠缺,且计算相似性时需要先对 SSDT 矩阵做同维化处理.

评估相似性算法时往往要求其对应的距离度量满足空间度量性质[20],包括非负性、对称性、恒等性以及三角不等式,非负性、对称性显而易见,恒等性要求过于苛刻,本文对三角不等式满足率进行验证. 由于各个算法得到的值都是相似性,所以需要将其转化为距离,用 $dist(M_1,M_2)=1-sim(M_1,M_2)$ 把相似性转化为距离. 从包含 n 个模型的数据集中任意选出 3 个模型,共有 C_n^3 种组合,对于这些组合分别验证 3 个不等式:

$$dist(M_1,M_2)\leqslant dist(M_1,M_3)+dist(M_2,M_3);\tag{8}$$

$$dist(M_1,M_3)\leqslant dist(M_1,M_2)+dist(M_2,M_3); \tag{9}$$

$$dist(M_2,M_3)\leqslant dist(M_1,M_2)+dist(M_1,M_3). \tag{10}$$

若针对 C_n^3 组模型,共 m 组模型满足三角不等式性质,则该数据集的三角不等式满足率为

$$rate=\frac{m}{C_n^3}. \tag{11}$$

TAR 算法、PTS 算法、SSDT 算法、BP 算法、CF 算法以及 TOR 算法的三角不等式满足率如表 8 所示:

Table 8 The Satisfaction Rate of Different Algorithms on the Triangle Inequality

表 8 各个相似性算法针对三角不等式的满足率 %

Dataset	TAR	PTS	SSDT	BP	CF	TOR
TC	100.0	99.9	96.7	100.0	100.0	100.0
DG	99.8	99.8	98.8	100.0	97.5	100.0
SAP	100.0	100.0	100.0	100.0	92.6	100.0

可以看出并不是所有的算法都满足三角不等式. TAR 算法在 DG 数据集上不满足三角不等式; PTS 算法在 DG 和 TC 数据集上不满足三角不等式;SSDT 在 TC 和 DG 数据集上不满足三角不等式;CF 算法在 DG 以及 SAP 数据集上不满足三角不等式;而 BP 算法和 TOR 算在 3 个数据集上均满足三角不等式. 而与 BP 算法相比,TOR 算法具有能处理不可见任务、区分循环和并发造成的任务间行为差异等优点.

4 总结与展望

针对现有流程模型行为相似性算法的不足,本文提出了一种能准确提取模型行为关系而又高效的算法——基于任务发生关系的模型相似性度量算法 TOR. TOR 算法首先把给定 Petri 网模型进行完全前缀展开,以清晰地表达模型的行为特征;接着,通过遍历对 CPU 图中的节点进行编号,以便于后续关系的求解;然后,对两两节点求解最近公共前驱,以确定节点对应的任务间的发生关系;最后,通过相应的公式计算模型的相似性. 在算法的适用性上,TOR 算法可以有效区分并发和循环的行为差异,高效处理非自由选择结构、不可见任务等复杂结构,TOR 算法同时还可以有效地提取任务间多关系. 基于来自 3 个企业的过程模型集的实验表明,TOR 算法具有很好的性能优势,能满足相似性度量算法的全部 5 个优良性质,且其对应距离满足三角不等式.

在流程模型相似性算法评估方面,仍然缺乏一个权威的公认评估框架,在未来的工作中,将尝试提出一个合理评价相似性算法优劣的评估框架. 同时,在进一步改进 TOR 算法以提升性能的同时,尝试将其应用于大量模型的索引方面.

参 考 文 献

[1] Dumas M, García-Bañuelos L, Dijkman R M. Similarity search of business process models [J]. IEEE Data Engineering Bulletin, 2009, 32(3): 23-28

[2] Becker M, Laue R. A comparative survey of business process similarity measures [J]. Computers in Industry, 2012, 63(2): 148-167

[3] Rahm E, Bernstein P A. A survey of approaches to automatic schema matching [J]. The VLDB Journal, 2001, 10(4): 334-350

[4] Doan A H, Madhavan J, Domingos P, et al. Ontology matching: A machine learning approach [G] //Handbook on Ontologies. Berlin: Springer, 2004: 385-403

[5] Dijkman R, Dumas M, Van Dongen B, et al. Similarity of business process models: Metrics and evaluation [J]. Information Systems, 2011, 36(2): 498-516

[6] Ehrig M, Koschmider A, Oberweis A. Measuring similarity between semantic business process models [C] //Proc of the 4th Asia-Pacific Conf on Conceptual Modeling. Sydney, Australia: ACS, 2007: 71-80

[7] Dijkman R, Dumas M, García-Bañuelos L. Graph matching algorithms for business process model similarity search [C] //Proc of the 7th Int Conf on Business Process Management. Berlin: Springer, 2009: 48-63

[8] La Rosa M, Dumas M, Uba R, et al. Merging business process models [C] //Proc of OTM 2010. Berlin: Springer, 2010: 96-113

[9] Li Chen, Reichert M, Wombacher A. On measuring process model similarity based on high-level change operations [C] //Proc of the 27th Int Conf on Conceptual Modeling. Berlin: Springer, 2008: 248-264

[10] Weidlich M, Mendling J, Weske M. Efficient consistency measurement based on behavioral profiles of process models [J]. IEEE Trans on Software Engineering, 2011, 37(3): 410-429

[11] Polyvyanyy A, Weidlich M, Conforti R, et al. The 4C spectrum of fundamental behavioral relations for concurrent systems [C] //Proc of the 35th Int Conf on Application and Theory of Petri Nets and Concurrency. Berlin: Springer, 2014: 210-232

[12] Zha Haiping, Wang Jianmin, Wen Lijie, et al. A workflow net similarity measure based on transition adjacency relations [J]. Computers in Industry, 2010, 61(5): 463-471

[13] Wang Jianmin, He Tengfei, Wen Lijie, et al. A behavioral similarity measure between labeled Petri nets based on principal transition sequences [C] //Proc of OTM 2010. Berlin: Springer, 2010: 394-401

[14] Wang Shuhao, Wen Lijie, Wei Daisen, et al. SSDT matrix-based behavioral similarity algorithm for process models [J]. Computer Integrated Manufacturing Systems, 2013, 19(8): 1822-1831 (in Chinese)
(汪抒浩, 闻立杰, 魏代森, 等. 基于任务最短跟随距离矩阵的流程模型行为相似性算法[J]. 计算机集成制造系统, 2013, 19(8): 1822-1831)

[15] Murata T. Petri nets: Properties, analysis and applications [J]. Proceedings of the IEEE, 1989, 77(4): 541-580

[16] McMillan K L, Probst D K. A technique of state space search based on unfolding [J]. Formal Methods in System Design, 1995, 6(1): 45-65

[17] Engelfriet J. Branching processes of Petri nets [J]. Acta Informatica, 1991, 28(6): 575-591

[18] Esparza J, Römer S, Vogler W. An improvement of McMillan's unfolding algorithm [J]. Formal Methods in System Design, 2002, 20(3): 285-310

[19] Wen Lijie, Wang Jianmin, van der Aalst W M P, et al. Mining process models with prime invisible tasks [J]. Data & Knowledge Engineering, 2010, 69(10): 999-1021

[20] Santini S, Jain R. Similarity measures [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 1999, 21(9): 871-883

Song Jinfeng, born in 1990. Master. Her main research interests include process model similarity and workflow technology.

Wen Lijie, born in 1977. PhD, associate professor, PhD supervisor. His main research interests include process data management, big process data, business process management and workflow technology, etc.

Wang Jianmin, born in 1968. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include business process management, workflow technology, big data and enterprise information systems, etc (jimwang@tsinghua.edu.cn).