软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(3):449−459 [doi: 10.13328/j.cnki.jos.004765]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于触发序列集合的过程模型行为相似性算法*

董子禾, 闻立杰, 黄浩未, 王建民

(清华大学 软件学院,北京 100084)

通讯作者: 闻立杰, E-mail: wenlj@tsinghua.edu.cn

摘 要: 过程模型的相似性计算是业务过程管理中不可缺少的任务,广泛应用于组织合并、用户需求变更、模型仓库管理等多个场景.对基于主变迁序列的相似性度量方法 PTS 进行研究,并提出了改进方案.通过定义完整触发序列表示模型行为,基于 A*算法结合剪枝策略实现触发序列集合间的映射,进而完成模型相似性计算.实验结果表明:该方法较主流的基于模型行为相似性算法,计算合理性有很大提升.

关键词: Petri 网;相似性度量;覆盖树;触发序列;A*搜索算法

中图法分类号: TP301

中文引用格式: 董子禾,闻立杰,黄浩未,王建民.基于触发序列集合的过程模型行为相似性算法.软件学报,2015,26(3):449−459. http://www.jos.org.cn/1000-9825/4765.htm

英文引用格式: Dong ZH, Wen LJ, Huang HW, Wang JM. Behavioral similarity algorithm for process models based on firing sequence collection. Ruan Jian Xue Bao/Journal of Software, 2015,26(3):449−459 (in Chinese). http://www.jos.org.cn/1000-9825/4765.htm

Behavioral Similarity Algorithm for Process Models Based on Firing Sequence Collection

DONG Zi-He, WEN Li-Jie, HUANG Hao-Wei, WANG Jian-Min

(School of Software, Tsinghua University, Beijing 100084, China)

Abstract: Similarity metric of process model is an indispensable task in business process management, which is widely used in many scenarios such as organizations merging, user requirements change, and model repository management. This paper focuses on the similarity metric algorithm based on the principal transition sequences (PTS) and puts forward improvement scheme by defining the complete firing sequences to express model behavior, matching fining sequences by A* algorithm with pruning strategy and proposing new similarity metric method. Experiments show that this method improves rationality than existing behavior-based similarity algorithms.

Key words: Petri net; similarity measure; coverability tree; firing sequences; A* search algorithm

近年来,随着工作流技术的发展和应用,很多大型企业或公共管理机构都使用过程模型形式化内部业务流程.随着组织在业务过程管理方面趋于成熟,大量模型被积累下来,如,中国移动通信集团公司的办公自动化系统的模型总数已经超过了 8 000 个.业务过程管理结合了信息技术和管理科学的知识并应用于操作业务流程[1],对于企业和组织的正常运转、决策变更、改革创新和绩效提升都具有重要意义.过程模型的相似性计算是业务过程管理中不可缺少的任务[2],广泛应用于组织合并、用户需求变更、模型仓库管理等多个场景.

过程模型相似性度量算法用于计算两个给定过程模型的相似性,目前,主流算法主要基于 3 个方面:(1) 任务标签、事件或其他建模元素;(2) 模型拓扑结构;(3) 过程模型的执行语义(即行为)[3].本文对基于主变迁序列的行为相似性度量算法 PTS 进行研究并提出了改进方案,将新的算法命名为 PTS++算法.

王建民等人提出了基于一种标签 Petri 网模型间行为相似性度量方法 PTS[4],该方法将 Petri 网的行为拆解

* 基金项目: 国家自然科学基金(61472207, 61325008); 教育部-中国移动科研基金(MCM20123011); 山东省自主创新专项基金(2013CXC 30001)

收稿时间: 2014-04-30; 修改时间: 2014-08-08; 定稿时间: 2014-11-21

为 3 类主变迁序列,分别表示非循环结构、有限循环结构和无限循环结构,通过求每类主变迁序列集合的相似度并加权得到模型相似度.该算法将模型中的循环结构与非循环结构分开考虑,打破了完整的行为语义,不能有效处理带循环结构的标签Petri网;同时,对主变迁序列集合相似性的计算缺少必要因素,在衡量序列与集合相似度时未考虑集合的整体情况,计算结果不够合理.

TAR 算法[5]通过定义变迁紧邻关系表达模型行为,通过 Jaccard 系数计算模型相似性,该算法不能区分循环结构和顺序结构,并且无法识别非自主选择结构.BP 算法[6]通过定义变迁间的多种关系构成模型的行为轮廓,利用多种关系的加权和得到模型间的相似度,该算法不能区分循环结构和并行结构,不能识别不可见任务.SSDT 算法[7]利用完全有向前缀计算模型的SSDT矩阵,进一步转化为同维可比较矩阵,通过对矩阵元素的比较得出模型的相似度,该算法不能直接作为模型索引,不同模型对会生成不同的同维化矩阵.CF 算法[8]将模型行为转化为因果足迹,包括活动集、后向连接集和前向连接集,进而将模型表达为空间向量中的点,通过计算向量夹角余弦值求模型相似度,该算法不能区分 XOR 连接、AND 连接和 OR 连接,因果足迹中包含很多冗余信息.

本文提出一种基于标签 Petri 网和覆盖树的模型行为相似性度量方法——PTS++算法,对 PTS 算法进行改进,并通过实验证明:PTS++算法能够全部满足文中提到的 6 个性质,优于 TAR 算法、BP 算法、SSDT 算法、CF 算法和 PTS 算法;同时,PTS++算法在三角不等式满足率方面达到 100%,优于 PTS 算法、CF 算法和 SSDT 算法.本文主要贡献如下:

(1) 定义完整触发序列表达模型行为,提出了利用覆盖树生成完整触发序列集合的方法;

(2) 提出的相似性计算方法考虑了集合的多种属性,更全面地刻画触发序列集合间的相似性;

(3) 利用 A*搜索算法求触发序列集合间最优映射,并设计了有效的剪枝策略;

(4) 通过实验对比 PTS++算法与主流算法,提出模型行为相似性度量算法应满足的新性质.

本文第 1 节主要介绍本文涉及的基本概念.第 2 节详细阐述 PTS++算法.第 3 节介绍实验设计和对比分析.第 4 节对本文进行总结并讨论未来工作.

1 相关知识

1.1 Petri网

定义 1(Petri 网系统). Petri 网系统是一个四元组$\Sigma=(P,T,F,M_0)$.P 是库所的有限集合;T 是变迁的有限集合;$F\subseteq(P\times T)\cup(T\times P)$是边的集合;$M_0$是 Petri 网的初始标识,$M_0:P\to N$(自然数集合).

本文提出的算法基于标签 Petri 网,标签的意义在于给变迁分配任务名称,这样变迁就可以与实际业务中的任务对应起来.标签 Petri 网的定义如下:

定义 2(标签 Petri 网系统). 标签 Petri 网系统是一个六元组 $L\Sigma=(P,T,F,M_0,A,L)$,其中,(P,T,F,M_0)表示 Petri 网系统,此外:A 是任务名称的有限集合;L 是变迁集合到 $A\cup\{\varepsilon\}$的映射,其中,ε为不可见任务或沉默任务.

为了便于叙述,将不区分变迁名和标签,同时,下文提到的 Petri 网均默认为标签 Petri 网.

定义 3(触发序列). 给定工作流网 $W\Sigma=(P,T,F,M_0,i,o)$,其中,i 为源库所,o 为汇结库所.令 $t\in T$,M_1 表示工作流网的一个标识.假设 t 在 M_1 时使能,即,对任意 $p\in\bullet t$,$M_1(p)\geqslant 1$:

$M_1[t>M_2$ 表示工作流网在 M_1 时 t 触发,状态标识变为 M_2,即:对任意 $p\in\bullet t\backslash t\bullet$,$M_2(p)=M_1(p)-1$;对任意 $p\in t\bullet\backslash\bullet t$,$M_2(p)=M_1(p)+1$;其他情况下,$M_2(p)=M_1(p)$;

$\sigma=\langle t_1,t_2,\ldots,t_n\rangle\in T^*$是一个从状态 M_1' 到状态 M_{n+1}' 的触发序列,表示为 $M_1'[\sigma>M_{n+1}'$.如果存在标识 $M_2',\ldots,M_n'$,使得 $M_1'[t_1>M_2'[t_2>\ldots>M_n'[t_n>M_{n+1}'$.

定义 4(完整触发序列). 给定工作流网 $W\Sigma=(P,T,F,M_0,i,o)$,$\sigma\in T^*$,令 M_i 表示初始标识,满足 $M_i(i)=1$,且对任意 p,如果 $p\neq i$,则 $M_i(p)=0$.令 M_o 表示结束标识,满足 $M_o(o)=1$,且对任意 p,如果 $p\neq o$,则 $M_o(p)=0$.若 $M_i[\sigma>M_o$,称σ为完整触发序列.

1.2 覆盖树

本文利用覆盖树分析 Petri 网的行为,覆盖树的生成过程本质上是从初始标识开始不断列举所有可达标识的过程,最后生成一棵以标识为节点、变迁的触发为边的树形结构.在 Petri 网是非有界的情况下,可达的标识有无限多个,这会导致覆盖树也是无穷的.为了解决这个问题,引入特殊标识ω,表示库所可能会拥有无限托肯,满足对任意整数 n,满足$\omega>n$,$\omega\pm n=\omega$且$\omega\geqslant\omega$.覆盖树生成算法见文献[9].

2 基于完整触发序列集合的行为相似性算法 PTS++

PTS++算法总体流程如图 1 所示:首先,通过覆盖树构造完整触发序列集合,其中给出循环的识别和计数方案以及完整触发序列构造方法;然后,利用完整触发序列集合计算模型相似性,其中包括应用 A*算法构造完整触发序列集合间映射以及相似性计算公式.

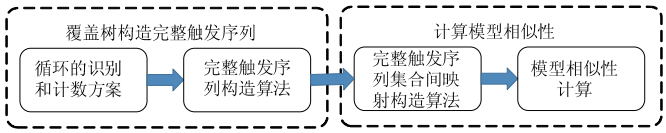


Fig.1 Overview of PTS++ algorithm

图 1 PTS++算法总体流程

2.1 完整触发序列集合的构造方法

在包含循环结构的模型中,完整触发序列的集合是无限的,不能直接用于相似性计算.本节给出一种通过指定循环次数得到有限的完整触发序列集合的方法,首先定义 Petri 网中的循环及其入口库所.

定义 5(循环). 令 Petri 网系统为$\Sigma=(P,T,F,M_0)$,循环 $N_C=(x_1,\ldots,x_n)\in(P\cup T)^*$,满足:

- 对任意整数 i,如果 $1\leqslant i<n$,那么$(x_i,x_{i+1})\in F$;
- $x_1=x_n$.

定义 6(循环入口库所). 令 Petri 网系统$\Sigma=(P,T,F,M_0)$是一个工作流网,其中,循环 $N_C=(x_1,\ldots,x_n)$,N_C 的入口库所 $p_e\in P$ 满足如下条件:

- 存在整数 i,满足 $1\leqslant i\leqslant n$,使得 $p_e=x_i$;
- 存在 $t\in T$,满足对任意整数 i,当 $1\leqslant i\leqslant n$ 时,$t\neq x_i$,且 $p_e\in t\bullet$.

使用循环入口库所来标识循环结构,当多个循环结构包含同一入口库所时,认为它们属于同一循环.根据该方案,可定义同一循环集合:

定义 7(同一循环集合). 令 Petri 网系统$\Sigma=(P,T,F,M_0)$是一个工作流网,若循环 $N_1,N_2,\ldots,N_n$ 都包含同一入口库所,那么它们属于同一循环集合;若不存在其他循环包含该入口库所,那么同一循环集合 C 可表示为

$$C=\{N_1,N_2,\ldots,N_n\}.$$

下面将同一循环集合的概念映射到覆盖树中,用以表达过程模型行为中的循环展开.

PTS 算法标注了覆盖树中 3 种标识节点:*anchor*,*old*,*dead-end*.根据覆盖树的定义,Petri 网中的循环结构对应到覆盖树中即为 *anchor* 节点到 *old* 节点的路径,将该路径定义如下:

定义 8(循环子路径). 令 Petri 网系统$\Sigma=(P,T,F,M_0)$,对应的覆盖树 CT_Σ中,每一个 *old* 节点可找到唯一与其对应的 *anchor* 节点,从该 *anchor* 节点到 *old* 节点的路径,表达了 Petri 网中的循环结构,称为循环子路径 p,用 $anchorMarking\xrightarrow{t_1,\ldots,t_n}oldMarking$ 表示(任取 $1\leqslant i\leqslant n$,满足 t_i 处在 *anchor* 节点到 *old* 节点的路径上,且依次连接),同时,称 *oldMarking* 为该循环子路径的循环标识,用 m_p 表示.

根据覆盖树的定义可知:Petri 网的同一循环集合对应到覆盖树中,即为循环标识相同的循环子路径集合.因此在覆盖树中,可以用 $C_{\text{循环标识}}$来表示同一循环集合.下面给出在覆盖树中计算同一循环集合执行次数的方案.

设覆盖树包含循环子路径 $p: m_p \xrightarrow{t_1 t_2 \ldots t_i} m_p (i \geqslant 1)$,当前搜索覆盖树得到的触发序列为$\sigma$,如果 $t_1 t_2 \ldots t_n$ 为σ的非连续子序列,则说明σ在生成过程中,经过了循环子路径 p.为便于计算触发序列经过循环子路径的次数,用 t_{p_end} 表示 p 的最后一个变迁,当序列生成过程中每经过 t_{p_end} 所在边一次,即经过 p 一次.令触发序列经过 p 的次数 $path\_count(p)$,触发序列经过 t_{p_end} 所在边次数为 $edge\_count(t_{p\_end})$,则

$$path\_count(p)=edge\_count(t_{p\_end}) \tag{1}$$

设覆盖树 CT 中所有循环子路径的集合为 P,当前触发序列σ中,经过循环子路径 p 的次数为 $path\_count(p)$,那么触发序列中同一循环集合 $C_{marking}$ 的执行次数 $cycle_count(C_{marking})$计算公式如下:

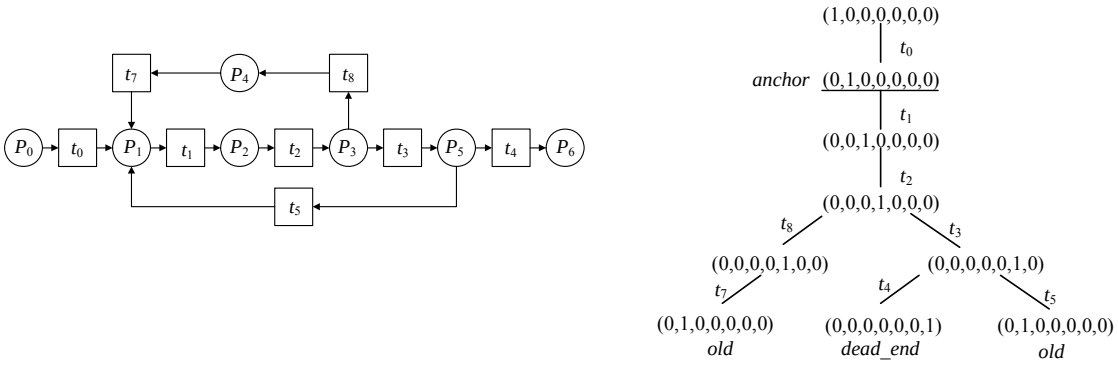
$$cycle\_count(C_{marking}) = \sum_{p \in P \text{且} m_p = marking} path\_count(p) \tag{2}$$

如图 2(a)的 Petri 网中包含同一循环集合:

$$C=\{(P_1,t_1,P_2,t_2,P_3,t_8,P_4,t_7,P_1),(P_1,t_1,P_2,t_2,P_3,t_3,P_5,t_5,P_1)\}.$$

对应到覆盖树图 2(b)中,同一循环集合 $C_{(0,1,0,0,0,0,0)}$包含循环子路径 $p_1:(0,1,0,0,0,0,0) \xrightarrow{t_1 t_2 t_8 t_7} (0,1,0,0,0,0,0)$ 和 $p_2:(0,1,0,0,0,0,0) \xrightarrow{t_1 t_2 t_3 t_5} (0,1,0,0,0,0,0)$.假设当前求得触发序列为 $t_0 t_1 t_2 t_3 t_5 t_1 t_2 t_8 t_7 t_1 t_2 t_8 t_7 t_1 t_2 t_2$,那么该同一循环的执行次数为

$$cycle\_count(C_{(0,1,0,0,0,0,0)})=path\_count(p_1)+path\_count(p_2)=edge\_count(t_7)+edge\_count(t_5)=2+1=3.$$



(a) Petri 网 (b) 图(a)中 Petri 网对应的覆盖树

Fig.2 Petri net and coverability tree

图 2 Petri 网及其对应的覆盖树

当遍历覆盖树求完整触发序列时,规定同一循环集合的执行次数上界为 k.对于每一个触发序列,若其中存在同一循环集合的执行次数大于 k,那么该触发序列将被丢弃;否则继续扩展.当生成完整触发序列并且所有同一循环集合在该序列中的执行次数不超过 k,将该序列加入完整触发序列集合中.

2.2 基于完整触发序列集合的相似性计算方法

由标签 Petri 网定义可知:每个变迁都被映射到一个活动名称,即标签.用 $L(\sigma)$表示触发序列σ对应的标签序列,那么可以从完整触发序列集合得到对应的标签序列集合.如图 3 所示,标签 Petri 网的执行可得到完整触发序列 $t_0 t_1 t_3 t_4 t_1 t_2$,对应标签序列为 $L(t_0 t_1 t_3 t_4 t_1 t_2)=XYZYYZ$.

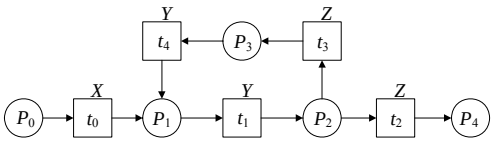


Fig.3 Labeled Petri net $L\Sigma_1$

图 3 标签 Petri 网 $L\Sigma_1$

首先,通过最长公共子序列方法比较两个触发序列.令触发序列为σ和σ',$lcs(L(\sigma),L(\sigma'))$表示两个序列的最长公共子序列,触发序列间的相似性公式为

$$sims(\sigma,\sigma')=\frac{length(lcs(L(\sigma),L(\sigma')))}{\max(length(L(\sigma)),length(L(\sigma')))} \tag{3}$$

接下来定义完整触发序列集合的相似度,该定义考虑了集合元素间的相似度和集合间的基数差异,并且定义集合间的单射;对于映射不到的完整触发序列,则应用了平均相似度.

令 A,B 是两个完整触发序列集合,假设$|A|\leqslant|B|$,令 $avg(A,\sigma')$表示完整触发序列集合 A 与完整触发序列σ'的相似度,那么,

$$avg(A,\sigma')=\frac{\sum_{\sigma\in A}sims(\sigma,\sigma')}{|A|} \tag{4}$$

令 $dis(A,B)$表示两个完整触发序列集合 A 和 B 的基数距离,那么,

$$dis(A,B)=\frac{||A|-|B||}{|B|} \tag{5}$$

令 $M:A\to B$ 为一个单射,该单射将 A 中的完整触发序列映射到 B 中的完整触发序列.令 $B'=B-cod(M)$,$n\geqslant 1$,那么两个完整触发序列集合 A,B 的相似公式为

$$simc(A,B)=\frac{\sum_{(\sigma,\sigma')\in M}sims(\sigma,\sigma')+\sum_{\sigma'\in B'}avg(A,\sigma')}{|B|}\times\left(1-\frac{dis(A,B)}{n}\right) \tag{6}$$

PTS++算法使用完整触发序列集合表示标签 Petri 网的行为,那么求标签 Petri 网的相似性,就转化为求完整触发序列集合的相似性.令标签 Petri 网为 $L\Sigma_A$ 和 $L\Sigma_B$,对应的完整触发序列集合为 A,B,假设$|A|\leqslant|B|$,那么标签 Petri 网相似度计算公式如下:

$$Sim(L\Sigma_A,L\Sigma_B)=simc(A,B) \tag{7}$$

以图 3 和图 4 中的标签 Petri 网为例,令同一循环集合的执行次数上界为 2,那么可得两模型的完整触发序列集合.假设集合间的单射如图 5 所示,$M=\{(\sigma_1',\sigma_1),(\sigma_2',\sigma_2)\}$,$B'=\{\sigma_3\}$,令公式(7)中的 n 取 10^3,那么,

$$Sim(L\Sigma_A,L\Sigma_B)=\frac{\frac{2}{3}+\frac{1}{3}+\frac{1}{2}\left(\frac{1}{3}+\frac{2}{9}\right)}{3}\times\left(1-\frac{\frac{1}{3}}{10^3}\right)=0.426.$$

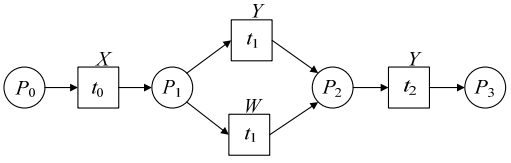


Fig.4 Labeled Petri net $L\Sigma_1$

图 4 标签 Petri 网 $L\Sigma_2$

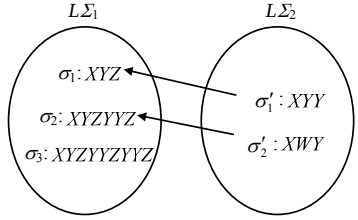


Fig.5 Injection between $L\Sigma_1$ and $L\Sigma_2$

图 5 $L\Sigma_1$ 和 $L\Sigma_2$ 的标签序列集合间的单射

2.3 完整触发序列集合间映射的构造算法

在计算两个标签 Petri 网的相似度时,采用不同的完整触发序列集合映射方案会得出不同的相似度结果.规定两个标签 Petri 网的相似度为所有可能的相似度结果中的最大值,对应的映射称之为最优映射.

由公式(7)可知:对于确定的 n,为求标签 Petri 网的相似度最大值,只需求 $sum=\sum_{(\sigma,\sigma')\in M}sims(\sigma,\sigma')+\sum_{\sigma'\in B'}avg(A,\sigma')$ 部分的最大值,因此需求出导致 sum 值最大的最优映射方案.遍历方法具有阶乘级的复杂度,而贪心算法通过局部最优选择求出的解往往不是最优解,下面利用 A*算法求最优映射方案.

A*搜索算法构造最优映射的过程可以看作映射树的生成过程,如图 6 所示,每个节点表示中间结果(M,A',B'),其中,M 表示当前的部分映射,A'表示完整触发序列集合 A 中尚未被映射的序列,B'表示完整触发序列 B 中尚未被映射的序列.

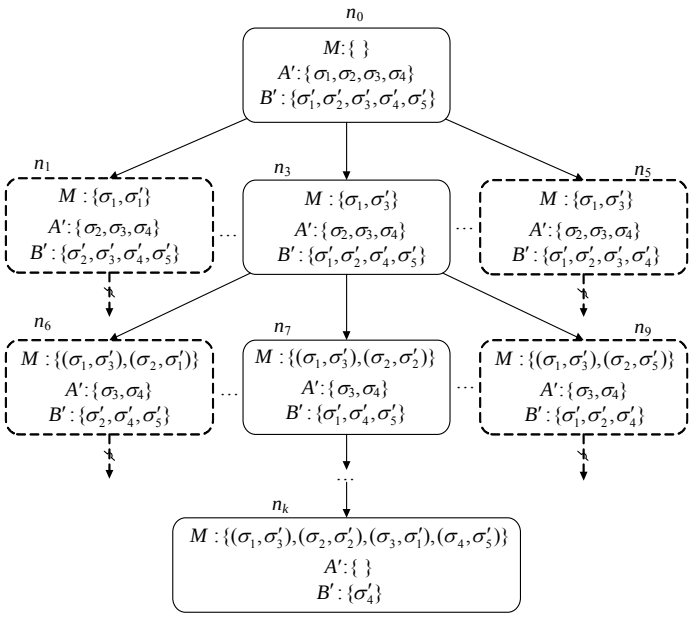


Fig.6 A* algorithm process in tree structure

图 6 以树形结构表示 A*搜索算法执行过程

在算法执行过程中,对于每个叶子节点,计算评估函数 $f(M)$,$f(M)$包含两个部分:

$$f(M)=g(M)+h(M) \tag{8}$$

其中,$g(M)$表示当前部分映射的解,$h(M)$表示映射剩余的序列可得到的解的上界.因此,$f(M)$表达的是从 M 进行扩展生成的所有映射解的上界.算法每次选择 $f(M)$最大的叶子节点进行扩展,当该叶子节点恰对应完整映射时,算法结束,该完整解即为最优解.A*搜索算法的伪代码如下:

输入:两个完整触发序列集合 A,B,假设$|A|\leqslant|B|$;

输出:最优映射 M 对应的 $f(M)$.

AStarAlgorithmforMap(A,B)

1. *open*:={{(σ,σ')}|$\sigma'\in B$}, for some $\sigma\in A$
2. while *open*$\neq\varnothing$
3. begin
4. $M:=\arg\max_{m'\in open} g(M')+h(M')$
5. *open*:=*open*−{M}
6. if *dom*(M)=A then return $g(M)+h(M)$
7. else
8. begin
9. select $\sigma\in A$, such that $\sigma\notin dom(M)$
10. for each $\sigma'\in B$, such that $\sigma'\notin dom(M)$
11. begin

12. M'':=M∪{(σ,σ')}
13. *open*:=*open*∪{M''}
14. end
15. end
16. end

该算法首先初始化映射集合 *open*,相当于树结构中的当前叶子节点集合.在循环过程中,算法从 *open* 中取出 $g(M)+h(M)$最大的映射 M 进行扩展,并将 M 从 *open* 中删除.如果当前 M 的定义域等于集合 A,那么 M 即为最优映射,返回 $g(M)+h(M)$;否则,从 A 中选出一个未被映射的序列σ,对于 B 中每个没有被映射的序列σ',形成映射对(σ,σ'),并加入 M 中形成新映射 M'',作为新的叶子节点加入到映射集合 *open* 中.

下面给出 $g(M)$和 $h(M)$的定义.

A*算法求出最优映射,最终返回 $\sum_{(\sigma,\sigma')\in M} sims(\sigma,\sigma')+\sum_{\sigma'\in B'} avg(A,\sigma')$ 的最大值,那么评估函数 $f(M')=g(M')+h(M')$应为从 M'进行扩展得到的所有可能的映射 M 对应的 $\sum_{(\sigma,\sigma')\in M} sims(\sigma,\sigma')+\sum_{\sigma'\in B'} avg(A,\sigma')$ 的上界.其中,$g(M)$表示当前部分映射的解,$h(M)$表示未来扩展部分对应解的上界.

假设当前节点的映射为 M,则 A 中未匹配的完整触发序列集合为 $A'=A-dom(M)$,B 中未匹配的完整触发序列集合为 $B'=B-cod(M)$.假设$|A|\leqslant|B|$,那么 $g(M)$定义如下:

$$g(M)=\begin{cases}\sum_{(\sigma,\sigma')\in M} sims(\sigma,\sigma'), & A'\neq\varnothing\\ \sum_{(\sigma,\sigma')\in M} sims(\sigma,\sigma')+\sum_{\sigma'\in B'} avg(A,\sigma'), & A'=\varnothing\end{cases} \tag{9}$$

$g(M)$函数值代表当前部分映射的相似度值,代表确定值部分.当 $A'\neq\varnothing$时,A 中存在未被映射的序列,这时,$g(M)$函数值为当前部分映射中每对序列的相似度之和;当$A'=\varnothing$时,A中的序列均被映射,此时得到的映射 M 为完整解,所有值都被确定,$g(M)$值为映射 M 对应的 $\sum_{(\sigma,\sigma')\in M} sims(\sigma,\sigma')+\sum_{\sigma'\in B'} avg(A,\sigma')$ 的值.

假设当前节点的映射为 M,则 A 中未匹配的完整触发序列集合为 $A'=A-dom(M)$,B 中未匹配的完整触发序列集合为 $B'=B-cod(M)$,假设$|A|\leqslant|B|$,令 $S=\{B''|B''\subseteq B'$,且$|B''|=|B|-|A|\}$,$h(M)$定义如下:

$$h(M)=\begin{cases}\sum_{\sigma\in A'}(\max_{\sigma'\in B'} sims(\sigma,\sigma'))+\max_{B''\in S}\sum_{\sigma''\in B''} avg(A,\sigma''), & A'\neq\varnothing\\ 0, & A'=\varnothing\end{cases} \tag{10}$$

当 $A'\neq\varnothing$时,$h(M)$由两部分构成:一是求 A'中每个元素与 B'中元素的最大相似度的和;二是取 B'中长度为$|B|-|A|$的子集 B'',求 B''中每个元素与 A 集合的相似度的和.其中,满足条件的 B''可能有多个,取对应的和最大的B''进行运算.当$A'=\varnothing$时,此时,A集合中的序列全部被匹配,没有需要扩展的序列;此时,M为完整解,没有需要估算的部分,因此 $h(M)=0$.

Hart 在文献[10]中定义了 A*搜索算法的可采纳性,该性质确保 A*搜索算法能够求出最优解,该问题的可采纳性描述如下:

定理 1. 对 A*搜索算法生成的任意部分映射 M,令 S 表示从 M 扩展可得到的所有完整解的集合,令 $M'=\arg\max_{M\in S} g(M)$,若 $h(M)\geqslant g(M')-g(M)$,即,$h(M)$是匹配剩余序列得到的部分解的上界,称 A*算法是可采纳的.

证明:假设 A*算法的完整解为 $A\to B$ 的单射,对任意部分映射 M,令 $A'=A-dom(M)$,$B'=B-cod(M)$.

当 $A'\neq\varnothing$时,令 $M_{new}=M'-M$,$B_{un}=B-cod(M')$表示最终未被匹配的序列,那么:

$$g(M')-g(M)=\sum_{(\sigma,\sigma')\in M_{new}} sims(\sigma,\sigma')+\sum_{\sigma''\in B_{un}} avg(A,\sigma'').$$

由于 $dom(M_{new})=A'$且 $cod(M_{new})\subseteq B'$,因此,$\sum_{\sigma\in A'}(\max_{\sigma'\in B'} sims(\sigma,\sigma'))\geqslant\sum_{(\sigma,\sigma')\in M_{new}} sims(\sigma,\sigma')$.

令 $S=\{B''|B''\subseteq B'$,且$|B''|=|B|-|A|\}$,则 $B_{un}\in S$,那么 $\max_{B''\in S}\sum_{\sigma''\in B''} avg(A,\sigma'')\geqslant\sum_{\sigma''\in B_{un}} avg(A,\sigma'')$,因此:

$$h(M)=\sum_{\sigma\in A'}(\max_{\sigma'\in B'} sims(\sigma,\sigma'))+\max_{B''\in S}\sum_{\sigma''\in B''} avg(A,\sigma'')$$
$$\geqslant \sum_{(\sigma,\sigma')\in M_{new}} sims(\sigma,\sigma')+\sum_{\sigma''\in B_{un}} avg(A,\sigma'')$$
$$=g(M')-g(M).$$

当 $A'=\varnothing$,此时,部分映射 M 为完整解,即 $M=M'$.此时,$h(M)=0$,$g(M')=g(M)$.因此,$h(M)=g(M')-g(M)=0$,满足 $h(M)\geqslant g(M')-g(M)$. □

3 实验设计与分析

通过在实际模型集的基础上衡量三角不等式的满足率,确定了算法中的参数值.当同一循环集合的执行次数上界 k 取 2、相似性公式中的 n 取 10^3 时,对应的三角不等式满足率最高.下面利用推荐参数值,首先考察 A*算法剪枝策略的效果,然后将 PTS++算法与主流基于行为的模型相似性算法进行对比.实验中使用的模型集分别来自唐山轨道客车有限责任公司(即 TC,简称唐车)的业务过程模型 124 个、东方锅炉股份有限公司(即 DG,简称东锅)的业务过程模型 115 个、赛普的业务流程参考模型(即 SAP)592 个,以及利用 BeehiveZ[11]自动生成的模型 200 个.

3.1 A*算法剪枝策略实验

从空间角度,衡量应用剪枝策略前后 A*算法可执行到的模型对比率,即在程序结束前计算的模型对数与模型总对数的比值.实验结果如图 7 所示:

- 如果不应用剪枝策略,那么 A*算法可执行到的模型对比率最高为 69.26%.由于 A*算法具有阶乘级复杂度,运行中,程序报出内存溢出错误,提前结束;
- 在应用剪枝策略后,A*算法可执行到的模型对比率达到 100%.这是因为剪枝策略只对有可能达到最优解的部分映射进行扩展,节省了存储空间.

从时间角度,以应用剪枝策略前 A*算法可执行到模型对作为数据基础,测量每个模型对的平均执行时间,如图 8 所示.从图中可以明显看到:应用剪枝策略后,算法的运行时间大幅度减少,最大减幅可达到应用剪枝策略前的 78.7%,应用剪枝策略可以有效地提高运行效率.

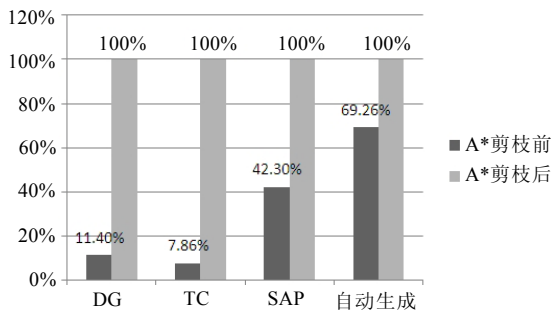


Fig.7 Model pair ritio before and after pruning in A*
图 7 剪枝前后 A*算法可执行到的模型对比率

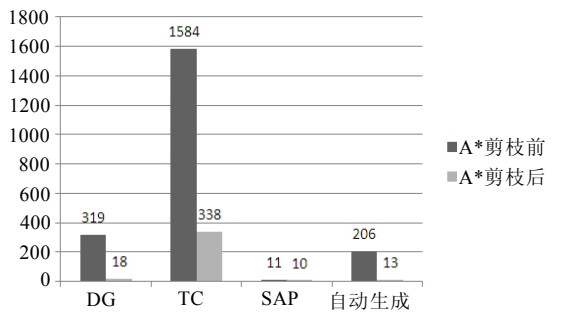


Fig.8 Average time before and after pruning in A*(ms)
图 8 剪枝前后 A*算法计算模型相似度的平均时间(ms)

3.2 PTS++算法与主流模型行为相似性算法对比

汪抒浩等人在文献[7]中给出了相似性算法应当满足的 5 个性质,这些性质符合业务实际和人类的主观直觉,一定程度上可以作为衡量不同相似性算法的依据.首先利用这 5 个性质比较 PTS++算法与主流模型行为相似性算法,结果见表 1.

Table 1 Result of five properties

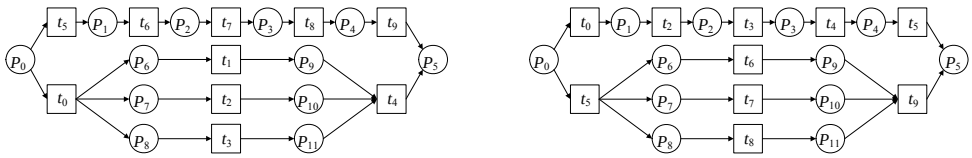
表 1 5 个性质的满足情况

性质	SSDT	PTS	TAR	CF	BP	PTS++
互斥结构漂移不变性	√	√			√	√
跨度负相关性	√	√				√
无关递减性	√		√	√	√	√
循环序列长度负相关性	√				√	√
顺序结构漂移不变性	√	√		√	√	√

本文在对实际业务模型分析的基础上,进行归纳总结并提出了如下新的性质:

假设两个业务过程模型Σ_1和Σ_2,均包含两类(设为A,B)完整触发序列,已知A类和B类属于不同业务,相似度很低,但同类完整触发序列相似度很高.模型Σ_1中,A 类完整触发序列占大多数,B 类占少数;模型Σ_2 反之.那么在两模型完整触发序列集合间进行一一映射后,少数映射对的相似度极高,其余映射对相似度极低.对于这样两个模型,他们之间的相似度偏低.不同类业务在这两个模型的行为中所占的比重不同,因此称其为业务行为分布的不平衡性.

如图 9 所示模型Σ_1和Σ_2,利用 PTS++算法求出的完整触发序列集合及其映射如图 10 所示.根据相似性公式,得出两模型相似度为 0.286.其他主流模型行为相似性度量算法得到的模型相似度见表 2.可以看出,TAR 算法和 PTS++算法的计算结果比较符合预期,而 SSDT、PTS、CF 和 BP 算法的计算结果均偏高.



(a) 模型Σ_1 (b) 模型Σ_2

Fig.9 Two Petri nets used for the unbalance of business behavior distribution

图 9 用于业务行为分布的不平衡性的两个 Petri 网

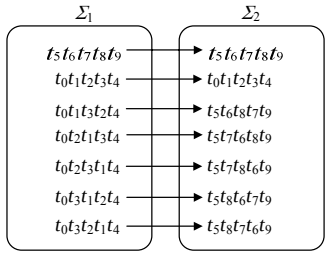


Fig.10 Best matching between models in Fig.9

图 10 图 9 中模型的触发序列集合间最优映射

Table 2 Similarity of models in Fig.9 in different algorithms

表 2 图 9 中模型对在不同算法下的相似性结果

算法名称	Σ_1和Σ_2的相似性
SSDT	0.84
PTS	0.829
TAR	0.333
CF	0.858
BP	0.68
PTS++	0.286

接下来考察 PTS,TAR,CF 算法和 PTS++算法在三角不等式满足率方面的表现.设定算法的执行时间阈值为 5 分钟,考察在阈值内计算出的相似度结果的三角不等式满足率.

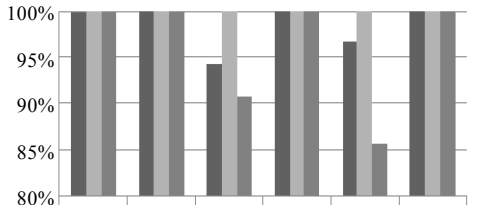
Matthias 等人在文献[12]中提出了三角不等式.假设模型集合为 S,且$|S|=n$,其中任意 3 个模型可组成一组,那么集合 S 中共有 C_n^3 组.若其中 m 组满足三角不等式,那么可以得到集合 S 的三角不等式满足率为

$$Rate(S)=\frac{m}{C_n^3} \tag{11}$$

本文用如下方法将相似度转化为距离,用以计算三角不等式:令标签 Petri 网为 $L\Sigma_A$ 和 $L\Sigma_B$,利用公式(7)计算相似度为 $Sim(L\Sigma_A,L\Sigma_B)$:

$$Dis(L\Sigma_A,L\Sigma_B)=1-Sim(L\Sigma_A,L\Sigma_B) \tag{12}$$

不同算法的三角不等式满足率如图 11 所示,其中,TAR 算法、BP 算法和 PTS++算法在实验数据集上可以完全满足三角不等式;PTS 算法在东锅和 SAP 数据集上有少部分不能满足三角不等式;CF 算法和 SSDT 算法的三角不等式满足率表现较差,只在唐车数据集上满足三角不等式.



	PTS	TAR	SSDT	BP	CF	PTS++
DG	99.9%	100%	94.3%	100%	96.7%	100%
TC	100%	100%	100%	100%	100%	100%
SAP	99.9%	100%	90.8%	100%	85.6%	100%

Fig.11 Rate of triangle inequality

图 11 三角不等式的满足率

综上可知:PTS++算法和主流的基于行为的模型相似性算法相比,在满足 6 个性质方面具有优势.首先,对于文献[7]给出的 5 个性质,考察 PTS++,SSDT,TAR,PTS,CF 和 BP 算法,只有 PTS++和 SSDT 算法能够满足全部 5 个性质,但 SSDT 算法不能满足本文提出的新性质,因此从性质满足度方面来看,只有 PTS++算法能够满足全部性质,计算结果更加合理直观.且已知 SSDT 算法在相似性计算时,其特征依赖于被比较的另一模型,不能直接提取用于计算.在三角不等式满足率方面,PTS++算法能够满足三角不等式,优于 PTS,SSDT 和 CF 算法.

4 总结与展望

本文提出了一种基于执行语义的过程模型相似性度量算法,该算法建立在标签 Petri 网和覆盖树的基础上,利用完整触发序列来表示过程模型的行为,可应用于包含循环结构的模型,并能够有效处理 Petri 网中的各类结构.此外,本文通过设计有效的剪枝策略提升了 A*搜索算法的运行效率.本文在衡量过程模型相似性算法方面提出了新的性质,并通过实验表明,该算法较其他基于行为的相似性算法计算结果更加合理.

在未来工作中,我们将致力于进一步提升算法的计算效率,设计更为有效的剪枝策略,并针对过程模型相似性算法提出一种更为全面的评估标准.

References:

[1] van der Aalst WMP. Business process management demystified: A tutorial on models, systems and standards for workflow management. In: Desel J, Reisig W, Rozenberg G, eds. Proc. of the Lectures on Concurrency and Petri Nets. Berlin, Heidelberg: Springer-Verlag, 2004. 1−65. [doi: 10.1007/978-3-540-27755-2_1]

[2] Becker M, Laue R. A comparative survey of business process similarity measures. Computers in Industry, 2012,63(2):148−167. [doi: 10.1016/j.compind.2011.11.003]

[3] Dumas M, García-Bañuelos L, Dijkman R. Similarity search of business process models. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2009,32(3):23−28.

[4] Wang JM, He TF, Wen LJ, Wu NH, ter Hofstede AHM, Su JW. A behavioral similarity measure between labeled Petri nets based on principal transition sequences (short paper). In: Meersman R, Dillon T, Herrero P, eds. Proc. of the Move to Meaningful Internet Systems (OTM 2010). Berlin, Heidelberg: Springer-Verlag, 2010. 394−401. [doi: 10.1007/978-3-642-16934-2_27]

[5] Zha HP, Wang JM, Wen LJ, Wang CK, Sun JG. A workflow net similarity measure based on transition adjacency relations. Computers in Industry, 2010,61(5):463−471. [doi: 10.1016/j.compind.2010.01.001]

[6] Kunze M, Weidlich M, Weske M. Behavioral similarity—A proper metric. In: Rinderle-Ma S, Toumani F, Wolf K, eds. Proc. of the 9th Int'l Conf. on Business Process Management (BPM 2010). Berlin, Heidelberg: Springer-Verlag, 2011. 166−181. [doi: 10.1007/978-3-642-23059-2_15]

[7] Wang SH, Wen LJ, Wei DS, Wang JM, Yan ZQ. SSDT matrix-based behavioral similarity algorithm for process models. Computer Integrated Manufacturing Systems, 2013,19(8):1822−1831 (in Chinese with English abstract).

[8] Dijkman R, Dumas M, van Dongen B, Käärik R, Mendling J. Similarity of business process models: metrics and evaluation. Information Systems, 2011,36(2):498−516. [doi: 10.1016/j.is.2010.09.006]

[9] Murata T. Petri nets: Properties, analysis and applications. Proc. of the IEEE, 1989,77(4):541−580. [doi: 10.1109/5.24143]

[10] Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. on Systems Science and Cybernetics, 1968,4(2):100−107. [doi: 10.1109/TSSC.1968.300136]

[11] Wu NH, Jin T, Zha HP, He TF, Wen LJ, Wang JM. BeehiveZ: An open framework for business process model management. Journal of Computer Research and Development, 2010,47(Suppl.):450−454 (in Chinese with English abstract).

[12] Kunze M, Weske M. Metric trees for efficient similarity search in process model repositories. In: zur Muehlen M, Su JW, eds. Proc. of the Business Process Management (BPM 2010) Workshops. Berlin, Heidelberg: Springer-Verlag, 2011. 535−546. [doi: 10.1007/978-3-642-20511-8_49]

附中文参考文献:

[7] 汪抒浩,闻立杰,魏代森,王建民,闫志强.基于 SSDT 矩阵的流程模型行为相似性算法.计算机集成制造系统,2013,19(8):1822−1831.

[11] 武年华,金涛,查海平,何腾飞,闻立杰,王建民.BeehiveZ:一个开放的业务过程模型管理框架.计算机研究与发展,2010,47(增刊):450−454.

董子禾(1988－),女,河北保定人,硕士生,主要研究领域为流程模型相似性度量.

黄浩未(1992－),男,硕士生,主要研究领域为流程模型相似性度量.

闻立杰(1977－),男,博士,副教授,主要研究领域为工作流技术,流程数据管理.

王建民(1968－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为大数据管理,信息系统与工程,流程数据管理与挖掘.