基于过程结构树的流程间差别检测算法

曹 斌,安卫士,王佳星,范 菁
(浙江工业大学计算机科学与技术学院,浙江杭州 310023)

摘 要: 流程模型差别检测是业务流程管理的关键技术之一.针对流程模型大多是由图结构建模,而流程图模型中有多种类型节点,因此经典的图编辑距离方法无法直接应用于流程差别检测的问题,提出了基于过程结构树的流程间差别检测算法.算法首先将流程模型转化为基于任务节点的过程结构树;然后采用分治思想快速获得流程间的最佳对等节点映射集合;最后基于节点映射集合生成一个近似最小代价编辑操作序列来表示两个流程的差别.实验结果表明,本文算法在准确率和效率两方面都能满足实际的应用需求.

关键词: 业务流程管理; 流程差别检测; 过程结构树; 编辑操作序列
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2018)04-0862-09
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.04.014

A Difference Detection Algorithm for Process Models Based on Process Structure Tree

CAO Bin, AN Wei-shi, WANG Jia-xing, FAN Jing
(*Department of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, Zhejiang* 310023, *China*)

Abstract: Detecting difference between process models is one of the key technologies in business process management. The classical graph edit distance cannot directly be used to detect the difference between process models because there are many kinds of nodes in a process model. To solve this problem, we present an algorithm for detecting difference between process models based on process structure tree. Firstly, the process models are converted to their corresponding task based process structure trees (TPSTs). Then the divide and conquer strategy is used to obtain the optimal mapped node set between two TPSTs. Finally, an edit script with approximate minimum cost is generated based on the mapped nodes, which is considered as the difference between two process models. The experimental results show that this algorithm can meet the real requirements in terms of precision and efficiency.

Key words: business process management; process difference detection; process structure tree; edit script

1 引言

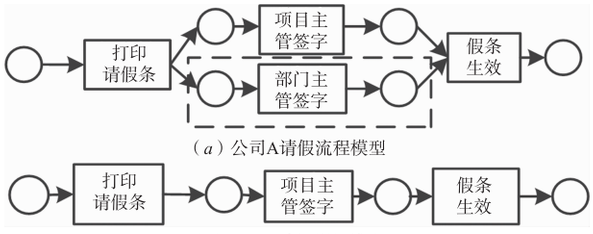
流程模型差别检测主要研究如何自动化的找出给定的两个流程模型间的不同.在业务流程管理(Business Process Management, BPM)过程中,很多应用场景都要对两个业务流程进行差别检测.如:(1)企业的业务流程会随着市场以及政府政策的变化而进行多次重建,因此一个流程会出现多个版本.在流程版本管理[1]中,管理人员要对同一个流程的不同版本进行比较,确定流程在哪些环节发生了变化;(2)在企业合并时,重叠业务的流程模型要进行合并[2],找出两个流程模型的差别是流程合并的第一步.

例如,图1中(*a*)、(*b*)分别对应公司A和公司B的请假流程,两者的差别如虚线框中所示.由两个请假流程可以看出,公司B的请假流程更加简洁高效.快速找出企业业务流程与其他公司相似业务流程的差别,便于业务流程人员对企业的业务流程进行优化,以此来增强企业的市场竞争力.

业务流程模型间的差别可以表示为一个编辑操作序列,即将一个流程模型变为另一个流程模型所需的编辑操作,如删除节点、插入节点等.例如图1中的流程(*a*)和流程(*b*),两者间的差别如虚线框中所示:流程

收稿日期: 2016-10-16; 修回日期:2017-04-27;责任编辑:李勇锋
基金项目: 国家自然科学基金(No. 61602411, No. 61572437);国家重点研发计划(No. 2016YFB1001403);浙江省重大科技专项重点工业项目(No. 2015C01029);杭州市重大科技创新项目(No. 20152011A03)

(*a*)比流程(*b*)多了一个步骤. 将流程(*a*)中虚线框中任务节点删除即转变为流程(*b*),同样在流程(*b*)中的相应位置插入该节点,则可将流程(*b*)转化为流程(*a*). 由图 1 可以看出,流程图模型中包含多种类型节点,如以 Petri 网建模流程为例,流程模型中包含库所(以圆圈表示)和变迁(以矩形表示)两种节点,因此经典的图编辑距离方法无法直接应用于流程差别检测问题上.



(*a*) 公司A请假流程模型

(*b*) 公司B请假流程模型

图 1　以petri网建模的请假流程模型

为解决上述问题,本文在文献[3]的过程结构树(Program Structure Tree,PST)的结构基础上,提出了基于任务节点的过程结构树(Task based Process Structure Tree,TPST). 随后通过计算两棵 TPST 之间的差别来表示两个对应流程模型的差别. TPST 是一棵半有序树,即部分节点的子节点是有顺序的,表示流程中的有序结构(如顺序结构、循环结构),部分节点的子节点是无序的,表示流程模型中的无序结构(如选择结构、并行结构). 与无序树 PST 相比,TPST 可以表示流程模型中包含的各种控制流结构.

ZHANG 等人已经证明生成一个最小代价编辑操作序列作为两个树之间的差别是一个 NP-Complete 问题[6]. 本文提出把近似最优的编辑操作序列作为两个流程模型间的差别,近似最优编辑操作序列即近似最小代价编辑操作序列. 算法共分为两大步骤,第一步主要进行两棵 TPST 中对等节点的映射;第二步是基于上一步得到的对等节点映射集合和分片映射集合生成一个近似最优的编辑操作序列,该编辑操作序列为两个流程模型的差别.

2　流程差别检测的相关概念

2.1　基于任务节点过程结构树

基于任务节点的过程结构树可由过程结构树转化得到,首先介绍过程结构树的概念. 过程结构树是将流程模型划分成一系列单入口单出口的模块(SESE)[4],并按照嵌套关系组织成的树型结构.

以图 2 中两个流程模型为例,首先将用 Petri 网建模的流程模型中的库所和变迁按照执行依赖关系转化为对应的网关节点,如图 3 所示. 随后将其划分成一系列流程模型片段,将流程模型的片段按照嵌套关系组织成树型结构就构成了流程模型的过程结构树,如图 4 所示.

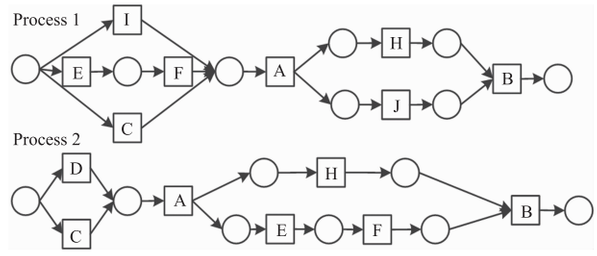


图2　两个Petri网建模的流程模型

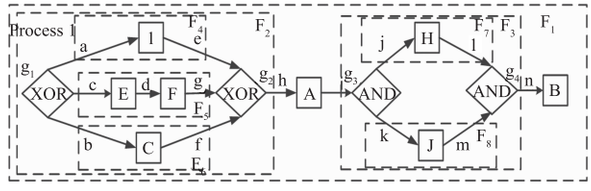


图3　Process 1对应的含有网关的流程模型

由图 4 可知,过程结构树中的叶子节点对应流程模型中的一条边,非叶子节点对应包含大于两条边的流程模型片段. 过程结构树是无序树,因此无法体现流程模型中任务执行的先后关系. 本文在过程结构树基础上提出了基于任务节点的过程结构树 TPST.

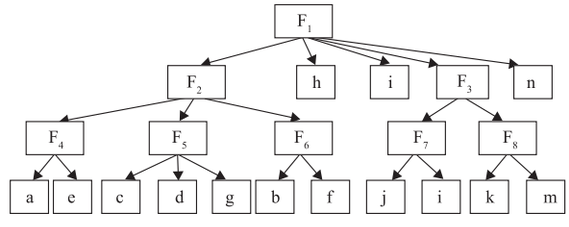


图4　图2中“Process 1”对应的PST

基于任务节点的过程结构树与过程结构树的差别在于,TPST 的叶子节点都是任务节点,非叶子节点为网关节点,并且 TPST 是半有序树结构. 下面举例说明 TPST 的特点,如图 5 所示:

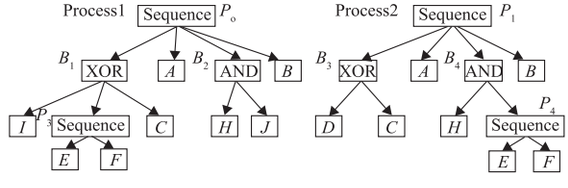


图5　图2中“Process 1”和“Process 2”的TPST结构

(1)叶子节点是任务节点,其子节点为空;

(2)非叶子节点是非任务节点,并有以下四种类型:XOR、AND、LOOP、Sequence,这四种类型节点分别对应 Petri 网中的选择结构、并行结构、循环结构、顺序结构,其中 XOR、AND 的子节点是无序的;LOOP、Sequence

的子节点是有顺序的;

(3)每个节点都有 label 和 type 两个属性,如图5中 Process 1 的任务节点 A,它的 label 为 A,type 为 Active,非任务节点(P_0, Sequence),label 为 P_0,type 为 Sequence;

2.2 分片及其映射规则

定义1 分片,由一个非任务节点加上与其直接相连的子节点组成,可以表示为一个三元组 $F=(root, node, type)$,其中 $root$ 为该分片的根节点(即非任务节点),$node$ 节点为分片中的所有节点集合,$type$ 为分片的类型,由 $root$ 的类型决定.

例如图5中的 Process 1 中的 B_1 节点,B_1 和与它直接点相连的子节点 I、P_3、C 组成一个分片,根节点 B_1 是类型为 XOR 的无序节点,所以该分片为无序分片;而 P_0 节点和其子节点$\{B_1, A, B_2, B\}$组成的分片是有序分片. 由定义1可知,将图5中 Process 1 划分为如下分片:$F_1(P_0, \{B_1, A, B_2, B\})$,$F_2(B_1, \{I, P_3, C\})$,$F_3(B_2, \{H, J\})$,$F_4(P_3, \{E, F\})$,其中 F_1 和 F_4 是有序分片,而 F_2 和 F_3 是无序分片. Process 2 的分片划分结果为:$F_5(P_1, \{B_3, A, B_4, B\})$,$F_6(B_3, \{D, C\})$,$F_7(B_4, \{H, P_4\})$,$F_8(P_4, \{E, F\})$,其中 F_5 和 F_8 是有序分片,而 F_6 和 F_7 是无序分片.

分片映射是基于分片中节点进行的映射,首先介绍单个节点的映射规则和两个分片间相似度的概念.

节点映射规则:x,y 节点分别来自两个不同的流程模型:

(1)若 type(x)≠type(y),则 x 和 y 不能映射;

(2)若 x,y 都是任务节点,并且 label(x) = label(y),则 x 与 y 映射;若 label(x)≠label(y),则 x 与 y 不映射;

(3)若 x,y 都是非任务节点,并且 type(x) = type(y),则 x 与 y 映射.

定义2 两个分片的相似度是由两个分片中映射的节点个数决定的,映射的节点个数越多则相似性越大. 相似度计算如式(1)所示,$|F_1|$代表分片 F_1 中所包含的节点个数,|MapNodes|代表两个分片中映射的节点对个数,两个分片的相似度是两个分片中映射节点个数的两倍,除以两个分片中节点数之和. 根据相似度来衡量两个分片映射的可能性,相似度越大,则两个分片映射的可能越大.

$$Sim(F_1, F_2) = \frac{2 \times |MapNodes|}{|F_1| + |F_2|} \tag{1}$$

由定义1可知,分片分为有序和无序两种,在考虑分片映射时也采取了两种映射策略:有序分片间的映射和无序分片间的映射.

无序分片映射规则:若 F_1,F_2 是无序分片,映射节点为两个分片的节点集合的交集,即$|MapNodes| = |F_1 \cap F_2|$;不考虑节点之间的顺序,只需按照上面的节点映射规则求两个分片中节点的映射个数. 如图6中所示,f_1 和 f_2 是两个无序分片,可知$|f_1 \cap f_2| = |\{XOR, A, B, C\}| = 4$,则$|MapNodes| = 4$,$Sim(f_1, f_2) = 1$,也就说 f_1 和 f_2 完全相同.

定义3 最长公共子节点序列(Longest Common Node Subsequence, LCNS),SN_1 和 SN_2 分别表示将分片 F_1、F_2 的节点按照根节点、子节点的顺序组成的节点串,S 为 SN_1 和 SN_2 的公共子节点序列,对任意的 SN_1 和 SN_2 的公共子节点序列 $S' \neq S$,满足$|S| > |S'|$,则 S 为 SN_1 和 SN_2 的最长公共子节点序列,记为 LCNS(F_1, F_2) = LCNS(SN_1, SN_2) = S.

有序分片映射规则:若 F_1,F_2 为有序分片,则映射节点个数为|MapNodes| = |$LCNS(F_1, F_2)$|.

有序分片映射要考虑节点顺序,因为流程模型中相同的任务节点,不同的执行顺序,代表不同的流程. 对有序分片的映射采用最大公共子序列的思想. 求两个分片中最大公共子节点序列,即把分片中的节点标识按执行顺序组成字符串,采用动态规划求两个分片的最大公共子节点序列,序列的长度作为两个分片的映射节点对数.

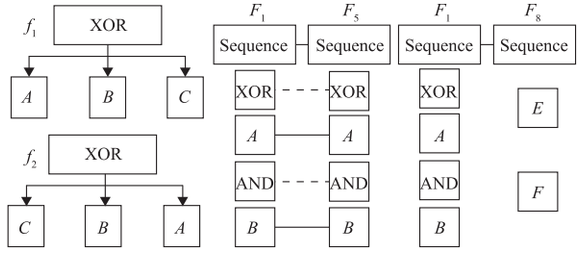


图6 两个无序分片

图7 分片F_1和F_5、F_8的映射结果

如以图5中 Process 1 和 Process 2 的有序分片集合映射为例,有序分片集合分别为$\{F_1, F_4\}$和$\{F_5, F_8\}$,首先取 F_1 和 F_5、F_8 分别进行映射,将 F_1 和 F_5 中的节点标识按照节点的顺序组成字符串,结果为:S_1 = {Sequence, XOR, A, AND, B}, S_5 = {Sequence, XOR, A, AND, B},LCNS(S_1, S_5) = {Sequence, XOR, A, AND, B}. 但其中除根节点外还有两个非任务节点 XOR 和 AND,它们分别为 F_2, F_3 的根节点(F_5 中的 XOR, AND 分别为 F_6, F_7 的根节点),后面对无序分片映射时也会映射这两个节点,这样会造成重复映射. 所以这里规定,分片映射过程中,除根节点外非任务节点暂不映射,所以最后的映射结果为{Sequence, A, B}. 如图7中所示,两个节点实线相连代表两个节点映射,由式(1)可得 Sim(F_1, F_5) = (3 * 2)/(5 + 5) = 0.6, Sim(F_1, F_8) = (1 * 2)/(5 + 3) = 0.25.

2.3　编辑操作

对于 TPST 间的差别检测,定义了三种类型的编辑操作:删除节点、插入节点、移动分片,通过不同的操作组合可以将一个 TPST 转化为另一个 TPST.

删除节点:Del($x(l,t)$),表示删除一个 label 为 l,type 为 t 的节点 x,若 x 为叶子节点,则直接删除即可;若 x 是非叶子节点(即非任务节点),要先将它的子节点的父节点指针指向该节点的父节点,然后删除 x.

插入节点:Insert($x(l,t)$,a,pos),表示把 label 为 l,type 为 t 的节点 x 插入到 a 节点的第 pos 个子节点位置上,若节点 a 为无序非任务节点,则插入的位置默认为 0. 对于有序非任务节点,插入位置为从左往右数该节点是其父节点的第 pos 个子节点.

移动分片:Move(x,a,pos),表示移动以节点 x 为根节点的分片,使其作为节点 a 的第 pos 个子节点,若节点 a 为无序非任务节点,则插入的位置默认为 0.

本文假设每个操作的代价均为 1,那么编辑操作序列的代价为其包含的操作数量.

定义 4　编辑操作序列(editScript),给定两个流程模型 P_1 和 P_2,一个有序编辑操作集合 $E(O_1,O_2,\cdots,O_n)$,其中 $O_i(0<i<n)$ 代表一个编辑操作,若流程模型 P_1 按照顺序执行 E 中的编辑操作可以转化为流程模型 P_2,则称 E 为 P_1 到 P_2 的编辑操作序列,记为 $E = editScript(P_1 \rightarrow P_2)$.

例如图 5 中的两个流程模型,把 Process 1 转换为 Process 2 的编辑操作序列为 *editScript*(Process 1 → Process 2) = {*Del*(*I*), *Del*(*J*), *Insert*(*D*, B_1, 0), *Move*(F_4, B_2, 0)},代价为 4.

3　差别检测算法实现

本节将具体介绍差别检测算法,算法主要分为两个阶段:(1)节点映射:该阶段分为两步,首先对流程模型的 TPST 的分片进行映射;然后基于分片映射集合进行节点映射;(2)基于第 1 步的节点映射集合和分片映射集合生成编辑操作序列. 3.1 和 3.2 节分别介绍了第一阶段的分片映射和节点映射,3.3 介绍了算法的第二阶段.

3.1　分片映射

在分片映射时要维护一个表结构,即分片映射表. 该结构是基于二维数组进行构建的,定义如下:

定义 5　分片映射表(*Fragment_Mapping_Table*)给定两个 Petri 网建模的流程模型的基于任务节点的过程结构树 t_1 与 t_2,将 t_1,t_2 按照定义 5 进行分片,将 t_1 中的每个有序分片 $F_i \in \{F_1,\cdots,F_i,\cdots,F_m\}$ 分别与 t_2 中的所有有序分片 $F_1',\cdots,F_j',\cdots,F_n'$ 进行映射,并记录 F_i $(1 \leqslant i \leqslant m)$ 与 t_2 中所有有序分片的相似度,生成一张 $m \times n$ 有序分片映射表(*Order_fragmentMapT*);同理建立无序分片映射表(*Unorder_fragmentMapT*),两张表合起来称为分片映射表.

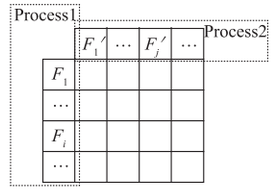


图8　分片映射表

	F_5	F_8
F_1	3/5	1/4
F_4	1/4	1

(1)

	F_6	F_7
F_2	4/7	0
F_3	0	1/3

(2)

图9　基于图5样例的分片映射表

如图 8 所示,表的第一列为流程 Process 1 中的分片,表的第一行是流程 Process 2 的分片,(i,j) 位置的值为分片 F_i 和 F_j' 的相似度. 图 5 中的两个流程模型的有序分片映射表如图 9(1)所示:F_5,F_8 为第一个 TPST 中的有序分片,F_1,F_4 为第二个 TPST 中的有序分片,3/5 表示分片 F_5 和分片 F_1 之间的相似度. 同理,无序分片映射表如图 9(2)所示.

在分片映射阶段,首先将流程模型对应的 TPST 结构分成一系列分片,通过式(1)来计算来自两个不同 TPST 的分片间的相似度,两个分片间的相似度越大,则它们越相似. 因为有序分片和无序分片在流程模型中代表不同结构,这里规定有序分片只能和有序分片映射,无序分片只能和无序分片映射.

分片映射的主要目的是找出最佳分片映射,最佳分片映射是指所有映射的分片的相似度加起来的和最大. 算法的输入为两个 Petri 网流程模型对应的 TPST:t_1 和 t_2,输出为最佳分片映射集合. 首先通过遍历两个流程模型对应的 TPST,找出两个流程模型所对应的有序分片和无序分片集合. 然后初始化分片映射表,该步骤的主要思路为遍历 t_1 中的有序(无序)分片集合 *order_F_*t_1,对其中的每个有序(无序)分片,考察其与第二个流程 t_2 中的有序(无序)分片集合 *order_F_*t_2 中的每个分片的相似度. 最后对初始化后的有序分片映射表和无序分片映射表分别使用匈牙利算法[7],找出最佳有序分片映射集合和最佳无序分片映射集合,两者的并集即为最佳分片映射集合.

图 9 是对图 5 中两个流程模型对应 TPST 的分片进行初始化映射的结果,根据 3.1 节中对两个流程模型的分片结果可知,两个流程均包含四个分片,两个有序分片和两个无序分片. 有序分片的初始化映射结果为图 9 中的(1),无序分片的初始化映射结果为(2). 对图 9 中的分片映射表使用匈牙利算法得到最佳有序分片映射集合为{(F_1,F_5),(F_4,F_8)},最佳无序分片映射集合为{(F_2,F_6),(F_3,F_7)},从而得到最佳分片映射集合为:M_F = {(F_1,F_5),(F_4,F_8),(F_2,F_6),(F_3,F_7)}.

3.2 节点映射

节点映射的目的是找出两个流程模型中相对应的节点,只考虑映射分片内节点的映射,减小了节点映射的选择空间,进而降低节点映射的时间. 该阶段主要依据上一步得到的最佳分片映射,根据2.2中节点映射规则,找出映射分片内的映射节点. 并将这些映射节点加入到节点映射集合 M.

以图5中的两个流程模型为例,由上步得到的最佳分片映射集合可知,F_1 和 F_5 映射,它们的映射节点如图7所示,连线的两个节点表示映射,即这两个分片的映射节点为{((P0:Sequence),(P0,Sequence)),(A,A),(B,B)},找出所有映射分片中的映射节点,求得节点映射集合为:M = {((P0:Sequence),(P0,Sequence)),((B1:XOR),(B1,XOR)),((B2:AND),(B2,AND)),((P4:Sequence),(P4,Sequence)),(A,A),(B,B),(C,C),(E,E),(F,F)}.

3.3 生成编辑操作序列

本节介绍编辑操作序列生成算法,算法输入为两个流程模型的TPST:t_1,t_2、节点映射集合 M 和分片映射集合 M_F,输出为可以实现两个流程模型转换的编辑操作序列 *editScript*. 算法伪码如算法1所示.

算法1 编辑操作序列生成算法

```
Input:  TPST t1,TPST t2,The node map:M,The Fragment map:MF
Output: The set of edit operation:editScript

1.  for each nodes of t1 in level-order do
2.      let x be the current node
3.      Parent(x)     //x节点的父节点
4.      if x not belong to M
5.          Add Del(x) to editScript;
6.  for each ndoes of t2 in level-order do
7.      let y be the current node
8.      Parent(y)     // y节点的父节点
9.      if y not belong to M
10.         Add Insert(x,parent(y),pos) to editScript;
11. for each Fragment pair(f1,f2) of MF
12.     parent_f1 = f1.root.parent // f1的父节点
13.     parent_f2 = f2.root.parent // f2的父节点
14.     if((parent_f1, parent_f2) not belong to M )
15.         Add Move (f1,parent_f2) to edit Script;
```

编辑操作序列生成算法分为3步:(1)删除节点操作:层序遍历 t_1,如果 t_1 中的当前节点 x 不在映射集合 M 中,则在 *editScript* 中加入 Del(x)(第1-5行);(2)插入节点操作:层序遍历 t_2,如果 t_2 中的当前节点 y 不在节点映射集合 M 中,则在 *editScript* 中加入 *Insert*(y,parent(y),*pos*)(第6-10行);(3)移动分片操作,遍历 M_F 中的分片映射对(f_1,f_2),如果分片映射对(f_1,f_2)满足 f_1 和 f_2 的根节点的父节点映射节点 M 中,说明两个分片的位置一样的,不需要进行移动;否则要进行移动分片操作,在 *editScript* 中加入 *Move*(f_1,parent(f_2. root),*pos*),其中 f_2. *root* 代表分片 f_2 的根节点,parent(f_2. *root*)代表根节点的父节点,即分片的父节点(第11-15行).

产生分片移动操作的原因是在进行分片映射时没有考虑分片的位置,可能两个位置相差很远的分片也可以映射在一起. 比如,分片(F_4,F_8)这两个分片映射,但它们的位置不同. 如果不将其映射在一起,则要将 F_4 中的节点全部删除,然后再将 F_8 中的所有节点插入到 t_1 中的对应位置,这样会带来更多的操作,所以将其映射,在最后进行判断两个映射分片的位置是否相同,相同则无须另外的操作,如果不同,只需将分片进行移动即可.

以图5中的两个流程模型为例,最后得到的编辑操作序列为:*editScript*(Process 1->Process 2) = {Del(I),Del(J),Insert(D,B1,0),Move(F4,B2,0)}.

4 实验结果与分析

本节通过实验评估算法的性能,主要从准确率和效率两个方面考虑. 在准确率方面,由于没有统一的评估标准,通过人工修改数据集并记录对应的修改操作序列作为两个流程模型的基准差别序列. 将算法得出的差别操作序列和基准差别序列进行对比,以此来评估算法的准确性,并考察流程模型复杂度对算法准确率的影响. 在效率评估实验中,设计了一系列实验考察流程模型某一元素(库所、变迁、边)数量变化以及分阶段考察流程模型复杂度对算法效率的影响.

本文所有实验基于如下环境:处理器 Inter(R)Xeon(R) CPU E5-2637,3.5GHz,8GB 内存,Window7 64 位操作系统,JDK1.7.

4.1 数据介绍

表1 数据集情况

min/max/average place	7/175/35.496
min/max/average task	5/168/34.9832
min/max/average edge	12/367/74.8488

实验数据集包含两个部分:(1)基于 IBM 公司的一组真实数据[6]改造而来的,选择其中的10个流程,对每个流程进行人工修改得到与之相关的9个流程,最终建立一个包含100个流程的流程库,表1展示了这100个流程的库所、变迁、边数的最大值、最小值、平均值信息;(2)模拟新建了四组单一结构流程库,分别包含 Sequence,XOR,AND,AND+XOR 四种结构. 其中每组包含6个流程模型:1个原始流程和5个变种流程,变种流程是在原始流程上进行插入、删除、移动节点等修改得到的. 原始流程和变体流程包含的任务节点数依次

递减,分别为 160、140、120、100、80、60.

4.2　准确率评估

评估准确率的实验中,使用数据集的第 2 部分,分别使用每个流程库中的原始流程(original)和变种流程(variant1,variant2,…,variant5)进行差别检测,通过算法得到的差别结果和基准差别序列进行对比,来测试流程结构和流程中任务节点数对算法准确性的影响. 图 10 展示了算法对顺序结构、选择结构、并行结构、选择并行结构流程模型进行差别检测的准确率结果. 对于顺序结构流程库,算法的准确率是 100%,而其他三种结构的准确率低于顺序结构.

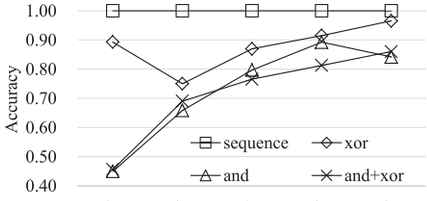


图10　四种不同结构准确率结果

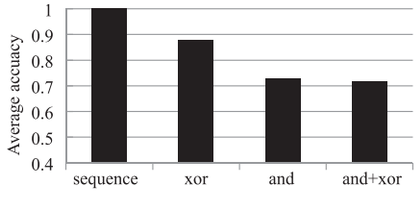


图11　四种不同结构的平均准确率

原因是在分片映射时,顺序结构的流程只有一个分片,因此一定能找到最佳分片映射,进而找到最佳的节点映射,使得其准确率为 100%;而其他三种结构包含多个分片,对于一个流程中的一个分片,在另一个流程中可能会存在多个分片与其相似度相同,使用匈牙利算法找最佳分片映射时存在多个最佳分片映射,无法保证得到最佳节点映射,使得其准确度下降. 如图 12 所示, Process 3 和 Process 4 的最佳的节点映射对为 $M=\{(A,A),(B,B),(D,D),(E,E),(F,F),(G,G),(H,H),(I,I),(P_0,P_5),(P_1,P_8),(P_2,P_9),(P_3,P_6),(P_4,P_7),(B_1,B_4),(B_2,B_3)\}$. 在无序分片映射时, Process 1 中的 F_1 和 F_2 分片(如左图虚线框所示)可以和 Process 2 中的 F_3 和 F_4(由右图虚线框所示)进行映射,由式(1)可知,$Sim(F_1,F_3)=Sim(F_1,F_4)=Sim(F_2,F_3)=Sim(F_2,F_4)=1/3$,最佳分片映射集合有两个 $M_1=\{(F_1,F_3),(F_2,F_4)\}$,$M_2=\{(F_1,F_4),(F_2,F_3)\}$,$M_1$ 所对应的节点映射为 $\{(B_1,B_3),(B_2,B_4)\}$,M_2 对应的节点映射为 $\{(B_1,B_4),(B_2,B_3)\}$,但本文所提算法对相似度相同的分片没有制定进一步的映射策略,因此对于 Process 3 中的非任务节点 B_1、B_2,哪个非任务节点与其映射是未知的.

由图 10 可以看出,除顺序结构外,其他三种结构,随着任务节点的减少,准确率呈增加趋势,原因是随着任务节点数减少,流程的分片集合会变小,进而减小了得到次优分片映射集合的可能. . 对于选择结构(xor)中 variant2 和并行结构中(and)的 variant5 的准确率下降,原因是在改造过程中,增加了如图 12 中 Process 3 中的 F_1 这样的分片,即分片中的节点全为非任务节点,进而增加了出现多个最佳分片映射的可能.

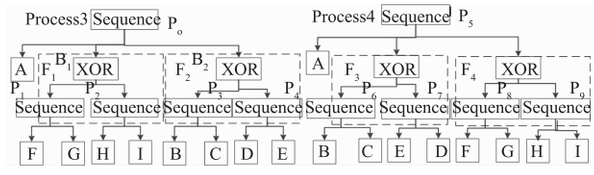


图12　两个Petri网流程样例的TPST

综上,可以看出,虽然随着流程结构的复杂度变高,差别检测算法准确率会随之降低. 但算法在一般场景下保持较高的准确率,由图 11 可知,算法对各种结构的平均准确率在 70% 以上.

4.3　效率评估

本节对算法的效率进行评估,首先在数据集第 1 部分中找出一些复杂度相当的流程进行一对一比较,考察流程的库所、变迁、边的数量变化对算法执行总时间的影响,并结合算法策略对实验进行分析.

图 13 表示待比较流程的库所数与算法耗时的关系. 可以看出,随着比较流程的库所数的增加,算法的执行时间增加. 这是因为随着库所数量的增加,流程模型所对应的分片的也会相应的增加,在初始化分片映射阶段,分片间的相似度计算次数会增加,所以差别检测总时间会增加.

图 14 表示待比较流程的变迁数与算法耗时的关系. 可以看出随着比较流程的变迁数的增加,算法的执行时间增加. 原因主要有两点:(1)若变迁的增加不产生新的支路,如图 17case1 所示,不会增加流程的分片数量,不增加初始化分片映射表时计算分片间相似度的次数;但在生成编辑操作序列时,遍历的节点数也会增加,所以执行总时间会增加. (2)若变迁的增加若导致了新的支路产生,如图 17 case2 所示,那么流程对应 TPST 中的分片会增加. 在初始化分片映射表时,分片间相似度计算的次数会增加,同样生成编辑操作序列的时间也会增加. 此外,增加同样的变迁数,case2 增加的时间要比 case1 多. 记含有 168 个变迁的流程与含有 99 个变迁的流程的差别检测的耗时为 $time_1$,与含有 88 个流程差别检测的耗时为 $time_2$. 从图 14 中可以看出,由 $time_1$ 到 $time_2$ 算法耗时增加显著. 这主要因为上述的原

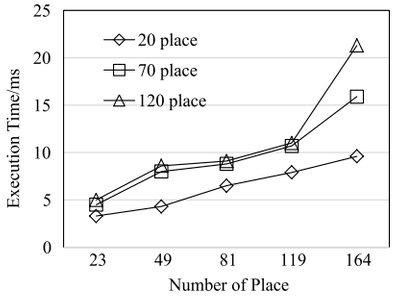


图13 耗时随库所数变化的关系

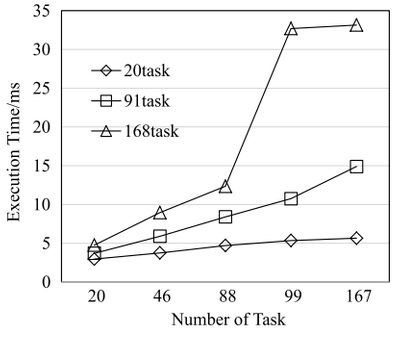


图14 耗时随变迁数变化的关系

因(2)造成的. 而含有 21 和 91 个变迁的流程模型的算法耗时曲线没有突然的变化,主要是因为这两个流程的结构与含有 168 个变迁的模型相比较为简单,这两个流程中的分片数量很少,即使与其比较的流程分片数增加,在初始化分片映射时,分片映射相似度计算次数也不会增加很多,因而时间变化比较平缓.

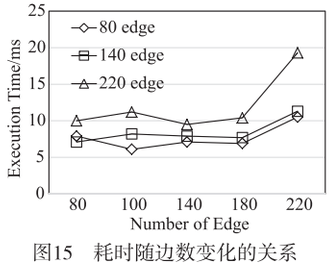


图15 耗时随边数变化的关系

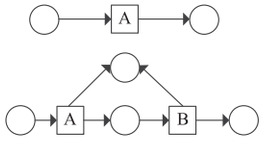


图16 增加边数

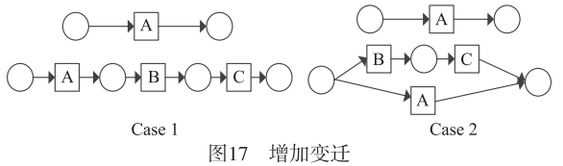


图17 增加变迁

边数的增加有两种情况,一种是会导致任务节点的增加,如图 17case1 所示,增加变迁 B 前后两条边;另一种是不会导致任务节点的增加,如图 16 所示,在变迁 A、B 之间增加了两条边. 对于第一种情况,由于增加了任务节点数,使得执行时间会增加;而对于第二种情况,随着边数的增加,流程对应的 TPST 并没有什么改变,所以执行时间不变. 由图 15 可以明显看出,随边数的增加,执行时间会增加. 这是由上述两种情况引起的,而对于其中执行时间显著增加的情况是由第二种情况导致的.

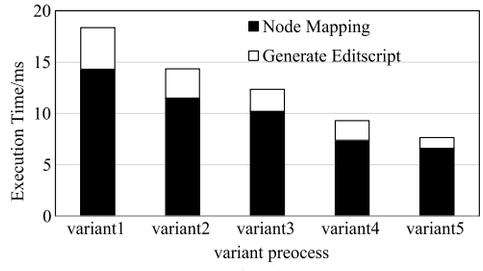


图18 顺序结构耗时

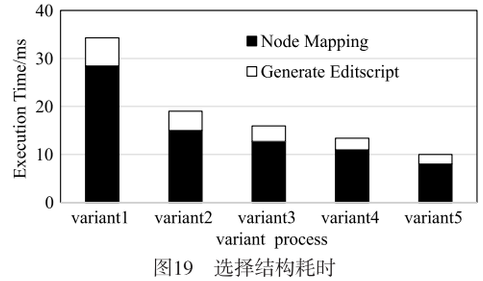


图19 选择结构耗时

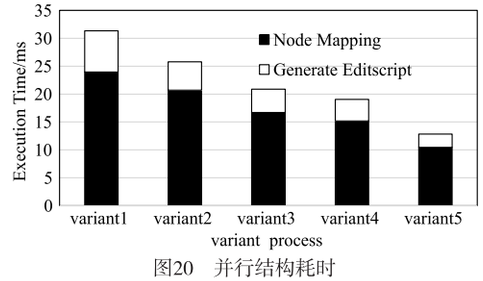


图20 并行结构耗时

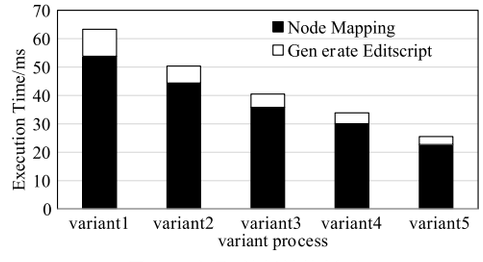


图21 选择+并行结构耗时

图 18、图 19、图 20 与图 21 所示的实验中,使用数据集第 2 部分中四种结构流程库中原始流程与其 5 个变种流程进行比较,并观察算法各个阶段执行时间的变化. 由图 18 到图 21 可以明显看出,随着流程结构复杂度的增加,算法执行总时间增加,其中生成编辑操作序列阶段时间增加不太显著,而节点映射阶段时间有显著的增加. 节点映射阶段时间增加是因为随着比较流程的复杂度越高,流程的分片就会增多,导致初始化分片映射表时,需要计算的分片间相似度次数增加,进而时间也相应的增加. 生成编辑操作序列时间也有相应的增加,但没有节点映射阶段变化显著. 这是由于生成编辑操作序列需要遍历流程中的节点,而各组流程中原始流程和变种流程的

任务节点节点数是相同的,随着流程结构的复杂度增加,非任务节点也会随之增加,但非任务节点与任务节点相比在节点总数中占的比例较小.

接下来对图 18 到图 21 四个实验结果进行分析. 以图 18 为例(其余三个实验结果与之类似),可以看出当结构不变,随着流程的任务结点数减少,节点映射和生成编辑操作序列这两个阶段时间都有减少. 原因是任务节点的减少,使得节点映射阶段比较的次数减少. 此外,在分片映射的初始化分片映射表阶段,分片中节点减少使得计算分片间的相似度的耗时减少,所以分片映射阶段的时间减少. 同样,任务节点的减少使得总的节点减少,生成编辑操作序列阶段的时间也会减少.

综上,可以看出差别检测的总时间均随着库所、变迁、边数目增加而增加,尤其当库所、变迁、边数目的改变导致流程结构变化时,时间变化的较为明显;同时,随着流程结构复杂度的增加,差别检测的总时间增加. 但就本文实验的结果结合流程差别检测在实际应用中的时间需求来看,本文提出的流程间差别检测算法可以满足实际应用的性能需求.

5　相关工作

流程模型间的差别检测用于计算两个给定流程模型间的不同部分. 目前为止对于该内容的研究工作不多. 下面具体介绍与本文相关的研究工作.

Dijkman R 等人将相似流程间频繁出现的差别进行了分类[7],并提出了一种方法来诊断 EPC 建模流程之间的差别,该诊断返回的是两个流程模型间差别的准确位置以及差别的类型[8]. Liu 等人[9]对流程模型在结构方面的差别进行了具体的定义,并通过实验证明这些差别在实际流程中都是存在的. Yan 等人[10]提出检测基于行为的流程模型差别检测算法,可以高效检测两个流程模型在行为方面的差别,与之前方法相比,在效率方面有了很大的提高.

Küster J M 等人[11]解决了在没有变化日志的情况如何计算两个给定流程的差别. 算法将流程被分解成单入口单出口(SESE)的多个片段. 检测两个流程中片段内和片段间的差别. Cao J 等人[12]将流程模型转换成过程结构树(PST),通过比较 PST 间的差别来找到流程间的差别,其中比较两个 PST 采用的是最大公共子树的方法. 与其相比,本文采用分片思想,可以得到更加准确的节点映射集合,进而增加算法的准确率.

6　总结与未来工作

本文采用基于任务节点的过程结构树来表示流程,将流程间的差别检测转化为两棵树之间的差别检测. 算法采用分治思想,首先将流程模型对应的 TPST 划分成一系列分片,采用匈牙利算法找出最佳分片映射集合,提高了非任务节点准确映射准确率,基于最佳分片映射进行映射分片内节点映射,获得最佳节点映射集合,同时减少了节点映射时的选择空间,提高节点映射效率. 最后生成一个近似最小代价编辑操作序列作为两个流程间的差别. 在实验环节,采用真实的数据集和模拟数据集,从准确性和效率两方面对所提算法进行了评估. 总的来说,实验结果表明本文提出的方法在准确性和性能上能够满足实际应用.

参考文献

[1] WEBER B, RINDERLE S, REICHERT M. Change Patterns and Change Support Features in Process-Aware Information Systems[M]. Berlin Heidelberg Germany: Springer, 2007, 574 – 588.

[2] LA ROSA M, DUMAS M, UBA R. Merging business process models[A]. On the Move to Meaningful Internet Systems: Confederated International Conferences [C]. Greece: DBLP, 2010. 96 – 113.

[3] POLYVYANYY A, VANHATALO J, VÖLZER H. Simplified computation and generalization of the refined process structure tree[J]. International Workshop on Web Services and Formal Methods, 2010, 6551(3): 25 – 41.

[4] VANHATALO J, VÖLZER H, LEYMANN F. Faster and more focused control-flow analysis for business process models through SESE decomposition[J]. Lecture Notes in Computer Science, 2007, 4749: 43 – 55.

[5] ZHANG K, STATMAN R, SHASHA D. On the editing distance between unordered labeled trees[J]. Information Processing Letters, 1992, 42(3): 133 – 139.

[6] KUHN H W. The Hungarian method for the assignment problem[J]. Naval Research Logistics, 2005, 52(1): 7 – 21.

[7] DIJKMAN R. Diagnosing differences between business process models[A]. International Conference on Business Process Management [C]. Germany: Springer-Verlag, 2008. 261 – 277.

[8] DIJKMAN R. A classification of differences between similar business processes[A]. Enterprise Distributed Object Computing Conference[C]. IEEE, 2007. 37.

[9] LIU K, YAN Z, WANG Y, et al. Efficient syntactic process difference detection using flexible feature matching[A]. Asia Pacific Business Process Management Conferences[C]. Germany: Springer, 2014. 103 – 116.

[10] YAN Z, WANG Y, WEN L, et al. Efficient behavioral difference detection between business process models [A]. Otm Conferences[C]. Germany Berlin Heidelberg: Springer, 2014. 220 – 236.

[11] Küster J M, GERTH C, FÖRSTER A, et al. Detecting and

resolving process model differences in the absence of a change log[A]. International Conference on Business Process Management[C]. USA:ACM,2008. 244 – 260.

[12] CAO J,YAO Y,WANG Y. Mining change operations for workflow platform as a service[J]. World Wide Web, 2015,18(4):1071 – 1092.

作者简介

曹 斌 男. 1985 年 5 月出生,山西孟县人,博士、讲师、CCF 会员,主要研究领域为业务流程管理、大数据.

E-mail: bincao@ zjut. edu. cn

安卫士 男. 1990 年 4 月出生,河南郸城县人,硕士生、CCF 学生会员,主要研究领域为业务流程管理.

王佳星 女,1990 年 11 月出生,浙江海宁人,博士生、CCF 学生会员,主要研究领域为工作流.

范 菁(通讯作者) 女,1969 年 12 月出生,浙江宁波人,博士、教授、博士生导师、CCF 高级会员,主要研究领域为服务计算、虚拟现实.