软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(11):3340−3354 [doi: 10.13328/j.cnki.jos.005483]

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

基于协作模式的工作流最优员工分配方法*

俞东进[1], 王娇娇[1], 柳诚飞[2]

[1](杭州电子科技大学 计算机学院,浙江 杭州 310018)

[2](Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne VIC3122, Australia)

通讯作者: 俞东进, E-mail:yudj@hdu.edu.cn

摘 要: 一个业务流程的执行,一般需要由多个员工共同协作完成.当员工完成流程中某项任务的能力已知时,员工之间的协作能力对于整个流程的执行性能就会有决定性的影响.通常,流程中执行活动的员工之间的协作能力越高,整个流程实例的运行效率就会越高.提出了一种基于协作模式的最优员工分配方法.该方法首先通过分析历史流程日志,计算不同员工在执行不同活动时彼此之间的协作能力;然后,从历史日志中挖掘出协作较好的员工分配方式(即协作水平较高的协作模式);最后使用编码的方式将这些模式与待分配流程快速匹配,选出可使流程协作水平达到最优的员工分配方式.实验结果说明,该方法能够快速、有效地实现流程协作最优的员工分配.

关键词: 协作模式;员工分配;实体;轨迹对齐;工作流

中图法分类号: TP311

中文引用格式: 俞东进,王娇娇,柳诚飞.基于协作模式的工作流最优员工分配方法.软件学报,2018,29(11):3340−3354. http://www.jos.org.cn/1000-9825/5483.htm

英文引用格式: Yu DJ, Wang JJ, Liu CF. Approach to optimal staff assignment in workflows based on collaboration patterns. Ruan Jian Xue Bao/Journal of Software, 2018,29(11):3340−3354 (in Chinese). http://www.jos.org.cn/1000-9825/5483.htm

Approach to Optimal Staff Assignment in Workflows Based on Collaboration Patterns

YU Dong-Jin[1], WANG Jiao-Jiao[1], LIU Cheng-Fei[2]

[1](School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China)

[2](Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne VIC3122, Australia)

Abstract: The execution of business process usually requires the collaboration of many staff members. When the capability of a staff member is determined, the collaboration becomes the dominating influence to the performance of process instances. In general, the better collaboration of the actors who perform collaborative tasks, the higher performance the workflow instance would achieve. In this paper, an approach to optimize staff assignment is proposed based on the collaboration patterns with high performance. Firstly, it computes the compatibility of actors when performing activities based on the historical logs. Afterwards, it mines collaboration patterns with high compatibility from process logs and matches them with the process for staff assignment by means of encoding. Experimental results demonstrate this approach can assign staff in a workflow for maximum collaboration efficiently.

Key words: collaboration pattern; staff assignment; entity; trace alignment; workflow

工作流中,人力资源的分配方式直接影响业务流程中任务执行的质量.特别地,在一个业务流程执行过程

* 基金项目: 国家自然科学基金(61472112, 61702144); 浙江省科技计划(2017C01010, 2015C01040)

Foundation item: National Natural Science Foundation of China (61472112, 61702144); Zhejiang Provincial Science and Technology Project (2017C01010, 2015C01040)

本文由面向智能制造的业务过程管理与服务技术专题特约编辑王建民教授、刘建勋教授推荐.

收稿时间: 2017-07-21; 修改时间: 2017-09-16; 采用时间: 2017-11-14; jos 在线出版时间: 2017-12-06

CNKI 网络优先出版: 2017-12-06 15:37:36, http://kns.cnki.net/kcms/detail/11.2560.TP.20171206.1537.029.html

中,相邻的两个任务之间往往需要大量的交互,而执行这些任务的员工(执行者)之间协作能力的高低,一般在很大程度上会影响整个流程的执行效率.比如在一个软件开发流程中,如果编码工程师能够与测试工程师很好地合作,整个软件的开发进度就有可能加快[1,2].

通常,工作流系统会从资源模型的角度(有能力执行某些活动的角色、员工)分析一个流程的执行情况.目前已有的资源分配方法大都是基于角色的分配方法.然而这种资源分配方法往往是“粗粒度”的[3],在某些情况下可能无法适用.例如,某制造业生产流程中的“生产计划”活动预定义由角色“生产计划设计者”执行,但是在实际分配中,可能需要将活动分配给该角色中的一个或多个更具有资格的员工而并非是该角色对应的所有员工.除此之外,也有一些人力资源分配方法关注适应度、紧急程度、符合度以及可用性等方面[4,5],但很少关注不同员工之间的协作水平.而现有的少数关于协作方面的研究,忽略了员工之间的协作能力会受到他们当前执行活动的影响.为了度量业务流程中员工之间的合作能力,Kumar 等人[6]提出了“协作度”这一概念.他们认为,有效的员工分配,能够使得流程整体协作度达到最大.事实上,我们认为,在员工分配时,不能仅根据员工之间的协作度大小进行分配,因为在员工之间的协作度会由于他们执行活动的不同而有所差别.不仅如此,这种仅通过计算员工之间的协作度进行员工分配的结果,就有可能会出现分配给某活动的员工并不具备执行该活动的权限.另一方面,这种员工之间的协作能力的度量显然也只能从历史流程实例中挖掘得到.事实上,从这些历史的日志中可以得到很多隐含的更富有实际意义的信息,如某活动可以很好地由某些员工执行(这些员工应该是属于角色对应的一部分更具有资格或者能力更强的员工)等.

为了解决以上问题,本文首先引入了实体(即活动和执行者的混合体)和实体协作度(即不同员工在执行不同任务之间的协作水平)的概念,然后在此基础上提出如何从流程运行的历史日志中挖掘出具有高协作能力的员工分配模式(即高协作水平的协作模式),最后,使用编码的方式将这些模式与待分配员工的流程快速匹配选出可使流程协作水平达到最优的员工分配方式.

本文的主要贡献在于:

(1) 提出了一种基于挖掘得到的高协作模式进行快速、有效的员工分配方法;

(2) 提出了实体协作度的概念,可用于度量不同员工在执行不同任务时彼此之间的协作水平;

(3) 构建了已分配员工的流程整体协作度计算模型;

(4) 提出了基于两种序列编码的匹配方式,用以快速获取基于高协作模式的最优员工分配方案.

本文第 1 节概述相关研究现状.第 2 节介绍相关的定义和问题描述.第 3 节详细介绍本文提出的员工分配方法框架、步骤以及算法.第 4 节基于合成数据集和真实的业务流程数据集进行实验评估与分析对比.第 5 节对全文的研究工作进行总结,并展望未来工作.

1 相关研究现状分析

近年来,已经有大量从资源管理的角度出发研究员工分配问题[7−9]的工作,特别是研究如何通过自动化实现合理的员工分配[10].如,Ly 等人[11]提出了从历史数据和组织模型中挖掘员工分配的规则作为决策树学习方法的输入进行员工分配;Liu 等人[12]提出了一种半自动的方法,使用机器学习算法从历史事件日志中学习哪些员工能够执行哪些活动;Xu 等人[13]提出了一种基于高斯领域的挖掘方法,可实现半自动化的员工分配.等等.这些方法都是使用机器学习中的学习方法对历史数据进行学习,然后根据学习的内容进行员工分配.另外,Cheng 等人[14]通过使用模糊数的理论解决了由于不同经验和能力的员工、任务的循环执行、优先级约束以及资源的限制等造成的不同的执行模式时实现最优的员工分配模型.除此之外,很多研究者关注员工分配的优化方法,例如:Liu 等人[15]提出了一种数据挖掘的方法找到事件日志的频繁模式,并根据预定义的关联规则生成资源分配的约束条件来解决员工分配问题,从而改善工作流资源管理的效率;Kumar 等人[6]提出了计算员工之间协作度的模型,用以员工分配;许荣斌等人[16]在此基础上提出了基于最大依赖度及最小冗余度的员工协作优化策略.然而,这类仅根据员工之间的协作度进行员工分配的方法忽略了员工之间的协作度在他们协作执行不同活动时会发生变化的情况.

另一方面,也有一些通过研究协作模式来提高流程的性能的工作,因为工作流中的一些好的协作模式往往代表着活动之间、员工之间具有良好的交互,其可以被用作分配员工时的一种经验参考.这方面的典型工作包括:Meddah 等人[17]提出使用流程挖掘的方法挖掘协作模式;Smirnov 等人[18]研究类似于协作模式的行为模式及其之间的关系;Verginadis 等人[19]论述了与协作相关的模式方法、模型和语言的研究结果,并提出使用已有的模式可以更好地为实现协作提供解决方案.此外,Diamantini 等人[20]提出一种使用图挖掘的理论知识在流程实例构成的图中挖掘协作模式的方法,但是这类基于子图挖掘理论进行模式挖掘的方法在时间复杂度上仍是一个挑战.然而,这些研究都仅仅只是挖掘出协作模式,并没有从协作能力的角度度量这些协作模式的协作水平.

2 问题和定义

2.1 基础概念

定义 1. 流程模型 $BPM=(A,C)$是由一系列的活动组成,它们通过获取一些资源(如执行者),按照活动之间的先后次序来执行,从而达到一个特定的目标,其中,$A=\{a_1,a_2,\ldots,a_i,\ldots,a_n\}$表示该流程中所有活动的集合,$C\subseteq(A\times A)$表示该流程中活动与活动之间的先后发生次序.

如图 1 所示,某制造业采购流程模型包括以下活动:采购申请(PP)、采购审批(PA)、供应商选择(SS)、价格谈判(QN)、订单签订(PS)、交货付款(DP),即 $A=\{PP,PA,SS,QN,PS,DP\}$,$C=\{\langle PP,PA\rangle,\langle PA,SS\rangle,\langle SS,QN\rangle,\langle QN,PS\rangle,\langle PS,DP\rangle\}$.它的每次执行都会得到一个对应的流程实例,这些流程实例会以日志的形式记录在流程管理系统中.

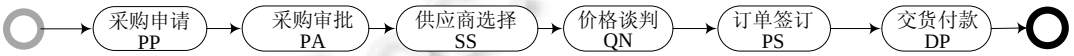


Fig.1 Purchasing process model in manufacturing

图 1 制造业采购流程模型

定义 2. 事件 $e=(\#_{startTime}(e),\#_{endTime}(e),\#_{activity}(e),\#_{resource}(e))$,其中,$\#_{startTime}(e),\#_{endTime}(e),\#_{activity}(e),\#_{resource}(e)$为事件 e 的属性,分别表示开始时间戳属性、结束时间戳属性、活动名称属性和资源属性.用 L 表示某一流程在运行时产生的事件日志,其中,$E_L=\{e_1,e_2,\ldots,e_i,\ldots,e_{n_L}\}$ 表示 L 中出现的所有事件的集合,$A_L=\{a_1,a_2,\ldots,a_i,\ldots,a_{n_L}\}$ ($A_L\subseteq A$)表示 L 中出现的所有活动的集合,$R_L=\{r_1,r_2,\ldots,r_i,\ldots,r_{n_L}\}$ 表示 L 中出现的所有员工(资源)的集合.

事件 e 是流程模型中活动的实例化,事件日志 L 是由 E_L 中一系列流程实例中的事件组成的.如表 1 所示,其中,第 1 行表示实例 ID 为 1,事件 ID 为 1 的事件有 6 个不同的属性:实例 ID 属性、事件 ID 属性、开始时间戳属性和结束时间戳属性(表示该事件开始于 2010/08/01 10:26:00,结束于 2010/08/02 14:06:00)、活动属性(该事件对应的是采购申请活动)和执行者属性(如 Sathish 完成该事件).

Table 1 An event log of the purchasing process in manufacturing

表 1 制造业采购流程的事件日志

实例 ID	事件 ID	属性			
		开始时间	结束时间	活动	执行者
1	1	2010/08/01 10:26:00	2010/08/02 14:06:00	采购申请/PP	Sathish/M1
1	2	2010/08/02 16:12:00	2010/08/03 14:18:00	采购审批/PA	Simona/M2
1	3	2010/08/03 09:24:00	2010/08/09 09:27:00	供应商选择/SS	Tammy/M3
1	4	2010/08/11 10:49:00	2010/08/12 11:21:00	价格谈判/QN	Deborah/M4
1	5	2010/08/12 12:27:00	2010/08/13 12:33:00	订单签订/PS	Maryse/M5
1	6	2010/08/14 11:11:00	2010/08/14 11:39:00	交货付款/DP	Cedric/M6
2	1	2010/08/05 10:20:00	2010/08/07 11:01:00	采购申请/PP	Sathish /M1
…	…	…	…	…	…

定义 3. 轨迹 $T_i=\langle e_{i,0};e_{i,1};\ldots;e_{i,j};\ldots;e_{i,T_{i.length}-1}\rangle$ 表示事件日志 L 中一个流程实例,由该流程实例中的所有事件按照发生的时间先后顺序组成的序列,其中,$e_{i,j}$ 表示第 i 个流程实例中第 j+1 个发生的事件,并在事件 $e_{i,j-1}$ 之后发生.使用事件的活动属性$\#_{activity}(e)$表示事件 e,轨迹又可记作 $T_i=\langle a_{i,0};a_{i,1};\ldots;a_{i,j};\ldots;a_{i,T_{i.length}-1}\rangle$.序列中元素之间的顺

序关系用$>_L$表示,如$e_{i,0}>_L e_{i,1}$.

如上述表 1 所示,相同实例 ID、不同事件 ID 的所有事件属于同一个流程实例,因此,实例 ID 为 1 的轨迹可表示为⟨采购申请;采购审批;供应商选择;价格谈判;订单签订;付款交货⟩,简单记作 T_1=⟨*PP*;*PA*;*SS*;*QN*;*PS*;*DP*⟩.

定义 4. “活动-执行者”实体(简称实体)$entity=\{(a_i|r_i)|a_i\in A_L,r_i\in R_L\}$表示事件日志 L 中执行者(员工)r_i 执行活动 a_i;实体轨迹 $ET_i=\langle a_{i,0}\,|\,r_{i,0};a_{i,1}\,|\,r_{i,1};...;a_{i,j}\,|\,r_{i,j};...;a_{i,T_{i.length}-1}\,|\,r_{i,T_{i.length}-1}\rangle$ 表示一个流程实例中的所有实体按照事件发生的时间顺序组成的序列.

如上述表 1 中的第 1 行所示,实例 ID 为 1、事件 ID 为 1 的事件就可以表示为:采购申请|*Sathish*(或 *PP*|*M*1),代表员工 *Sathish* 执行采购申请活动,该实例的实体轨迹可表示为 ET_1=⟨*PP*|*M*1;*PA*|*M*2;*SS*|*M*3;*QN*|*M*4;*PS*|*M*5;*DP*|*M*6⟩.

定义 5. 序列对齐是通过使用空格占位符和移动序列中元素的位置,尽可能地花费最小的代价实现序列中相同元素对齐的一种方式,从而发现相同的子序列,如,两条序列 *GCATTCA* 和 *GATTACA* 可能会得到如下的一种对齐:

G C A T T - C A

G - A T T A C A

实体轨迹对齐根据序列对齐的思想将实体轨迹序列实现对齐,从而快速发现相同的子实体序列.比如,两条实体轨迹 ET_1=⟨*PP*|*M*1;*PA*|*M*2;*SS*|*M*3;*QN*|*M*4;*PS*|*M*5;*DP*|*M*6⟩和 ET_2=⟨*PP*|*M*1;*PA*|*M*2;*SS*|*M*3;*QN*|*M*4;*SS*|*M*3;*QN*|*M*4;*PS*|*M*5;*DP*|*M*6⟩,可以得到如下的对齐:

PP|*M*1 *PA*|*M*2 *SS*|*M*3 *QN*|*M*4 - - *PS*|*M*5 *DP*|*M*6

PP|*M*1 *PA*|*M*2 *SS*|*M*3 *QN*|*M*4 *SS*|*M*3 *QN*|*M*4 *PS*|*M*5 *DP*|*M*6

如某个流程的事件日志 L 中有 n 个流程实例,即可以得到 n 条实体轨迹.根据上述的对齐思想,可以得到对齐的实体轨迹矩阵:

$$AM_{n\times m}=(a_{i,j}\,|\,r_{i,j})_{n\times m}=\begin{bmatrix} a_{0,0}\,|\,r_{0,0} & a_{0,1}\,|\,r_{0,1} & \cdots & a_{0,m-1}\,|\,r_{0,m-1} \\ a_{1,0}\,|\,r_{1,0} & a_{1,1}\,|\,r_{1,1} & \cdots & a_{1,m-1}\,|\,r_{1,m-1} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n-1,0}\,|\,r_{n-1,0} & a_{n-1,1}\,|\,r_{n-1,1} & \cdots & a_{n-1,m-1}\,|\,r_{n-1,m-1} \end{bmatrix}.$$

其中,$a_{i,j}|r_{i,j}$ 表示对齐后的第 i 条实体轨迹的第 j 列的元素,该元素为某一实体或者“-”,同一列的所有非“-”实体对应的活动相同.

定义 6. 参考活动序列 $R_{as}=\langle a_{\cdot,0};a_{\cdot,1};\ldots;a_{\cdot,j};\ldots;a_{\cdot,m-1}\rangle(a_{\cdot,j}\in A_L)$表示对齐的实体轨迹矩阵 $AM_{n\times m}$ 中最长的一条活动序列,即,将矩阵的每一列中所有非“-”实体对应的同一个活动作为该列的活动,最后得到的 m 个活动组成的活动序列.待分配活动流程 $aP=\langle a_1;a_2;\ldots;a_i\rangle(a_i\in A_L)$表示针对某个流程模型正在实例化的一个流程实例,即,一个待分配员工的活动序列(仅有活动信息).

根据上述定义,对于上述对齐的两条实体轨迹 ET_1 和 ET_2,我们可以得到参考活动序列为⟨*PP*;*PA*;*SS*;*QN*;SS_2;QN_2;*PS*;*DP*⟩.

定义 7. 活动协作度 $coop_{a_1,a_2}$ 用来度量事件日志 L 中两个有协作关系的活动 $a_1(a_1\in A_L)$和 $a_2(a_2\in A_L)$之间的关系,大小为 0-1 范围内的某个连续值,反映了流程中的两个连续的活动在执行时的协作程度:

$$coop_{a_1,a_2}=\frac{1}{1+e^{-k(c_{a_1,a_2}-\bar{c})}}(a_1\neq a_2,a_1>_L a_2) \tag{1}$$

其中,c_{a_1,a_2} 表示两个连续的活动 $a_1>_L a_2$ 在事件日志 L 中出现的频繁度,$\bar{c}$ 表示事件日志 L 中所有类型的两个连续活动在日志中出现的平均频繁度,k 是一个用来调优的参数.

定义 8. 活动序列协作度 $Coop_{as}$ 表示一个活动序列 $as=\langle a_k;a_{k+1};\ldots;a_j;\ldots;a_l\rangle(0\leqslant k<l\leqslant m,a_j\in A_L)$中所有有协作关系的连续活动之间总的协作程度:

$$Coop_{as}=\sum_{\forall(a_1,a_2)\in as}coop_{a_1,a_2} \tag{2}$$

定义 9. 实体协作度 $compatibility_{a_1|r_1,a_2|r_2}$ 用来度量事件日志 L 中两个员工 $r_1(r_1\in R_L)$和 $r_2(r_2\in R_L)$在协作完成两个活动 $a_1(a_1\in A_L)$和 $a_2(a_2\in A_L)$时的协作能力,即实体 $a_1|r_1$ 和 $a_2|r_2$ 之间的协作能力,大小为 0-1 范围内的某个连续值.

$$compatibility_{a_1|r_1,a_2|r_2}=\frac{1}{1+e^{-k(\overline{t}_{a_1|,a_2|}-\overline{t}_{a_1|r_1,a_2|r_2})}}(a_1\neq a_2,r_1\neq r_2;a_1\mid r_2>_L a_2\mid r_2) \tag{3}$$

其中,$\overline{t}_{a_1|r_1,a_2|r_2}$ 表示事件日志 L 中员工 r_1,r_2 执行活动 a_1,a_2 的平均所需时间,$\overline{t}_{a_1|,a_2|}$ 表示事件日志 L 中协作执行该两个活动 a_1,a_2 的平均所需时间,k 是一个用来调优的参数.

定义 10. 实体序列协作度 $Compatibility_{es}$ 表示事件日志 L 中一个已分配员工的活动序列,即实体序列 $es=\langle a_k|r_k;a_{k+1}|r_{k+1};\ldots;a_j|r_j;\ldots;a_l|r_l\rangle(0\leqslant k<l\leqslant m-1,a_j\in A_L,r_j\in R_L)$的整体协作能力.

$$Compatibility_{es}=\sum_{\forall(a_1|r_1,a_2|r_2)\in es}(coop_{a_1,a_2}\times compatibility_{a_1|r_1,a_2|r_2}) \tag{4}$$

定义 11. 高协作模式用一个二元组集合$\langle es,cpt\rangle$表示,其中,$es=\langle a_k|r_k;a_{k+1}|r_{k+1};\ldots;a_j|r_j;\ldots;a_l|r_l\rangle(0\leqslant k<l\leqslant m-1,a_j\in A_L,r_j\in R_L)$表示事件日志 L 中协作能力较高的实体序列,$cpt=Compatibility_{es}$ 表示该实体序列的协作度值.高协作模式代表历史流程实例中的员工在协作完成活动时的协作能力比较高的一种方式.

定义 12. 基于最大协作度的员工分配(最优员工分配)策略用一个三元组集合表示,其中,

- $aP=\langle a_1;a_2;\ldots;a_i\rangle(a_i\in A_L)$表示由用户提供的待分配活动流程;
- $eP=\langle a_1|r_1;a_2|r_2;\ldots;a_i|r_i\rangle(a_i\in A_L,r_i\in R_L)$表示针对该活动流程中的活动分别进行员工分配后使得整体协作度最大的实体序列;
- $cpt=Compatibility_{eP}$ 表示相应的分配策略所对应的协作度值.

2.2 问题描述

对制造业采购流程系统来说,一旦采购经办人成功发起了采购申请,即正确填写了采购的相关信息,采购经理就需要对此次采购请求进行审批;如果审批通过,采购人员需要根据采购单和报价选择相应的供应商;选定合适的供应商后,财务部人员需要与供应商进行价格谈判;待双方达成一致意见后,业务人员需要与供应商签订订单;订单签订成功后,供应商可以开始按照订单生产;最后交货时,需要财务人员付款.显然,整个采购流程需要由来自不同部门的员工共同协作才能完成.在这个过程中,若采购经办人员非常了解采购经理审批的依据,那么他在填写采购申请单时就可以避免错误,从而快速实现采购申请的批准.同样地,如果采购人员与财务人员的协作水平较好,那么他就可以在选择供应商的时候尽可能地选择财务人员更能认可的供应商.这样就可以极大地提高整个采购流程的效率.因此,本文从协作的角度研究员工分配问题:一方面,为了最大程度地提高流程执行的性能,文中提出了一种使流程整体协作度最大的员工分配方法;另一方面,历史流程执行日志中隐含地存在一些协作较好的员工执行活动的模式,这种分配方式应该被发现并运用在优化员工分配问题中,所以本文提出了一种方法用来挖掘协作水平较高的协作模式.基于这些协作模式,以整体协作度最大为目标,实现员工最优分配.

目前,从协作的角度实现员工最优分配的研究是基于员工之间的协作能力来进行分配的,因为忽略了员工之间的协作水平会由于执行不同的活动而影响,这种分配方法会出现不合理的分配.比如,某事件日志有员工集合$\{r_1,r_2,r_3,r_4\}$和活动集合$\{a_1,a_2,a_3,a_4\}$,根据日志统计:员工 r_1 和 r_2 协作执行活动 a_1 和 a_2(即 a_1:r_1,a_2:r_2)的协作度为 0.6,协作执行活动 a_3 和 a_4(即 a_3:r_1,a_4:r_2)的协作度为 0.2,员工 r_3 和 r_4 协作执行活动 a_3 和 a_4(即 a_3:r_3,a_4:r_4)的协作度为 0.3.然后,在此基础上对活动序列$\langle a_3;a_4\rangle$分配员工,如果仅根据员工之间的协作度来进行最大协作度分配就会出现 a_3:r_1,a_4:r_2 的不合理分配(因为 r_1 和 r_2 之间的平均协作度为 0.4,r_3 和 r_4 之间的平均协作度为 0.3,实际上 r_3 和 r_4 在协作执行 a_3 和 a_4 时的协作度更高一些).

为了解决这种不合理分配问题,本文首先提出了活动协作度和实体协作度的计算,对员工之间协作执行不同活动的协作水平进行准确度量.根据这种度量方式,挖掘历史日志中存在的具有高协作水平的员工分配模式,然后参考这些模式,将待分配员工的活动流程实现合理有效的流程最大化协作的员工分配.下面用一个简单的例子说明本文是如何实现协作最优员工分配以及最优分配的意义(a_1(Time)表示员工完成活动 a_1 所需的时间).

根据图 2、表 2 和表 3,可以分别计算出不同员工分配方法分配后的实体序列的协作度.假如待分配活动流程为 4 个活动组成的序列$\langle a_1;a_2;a_3;a_4\rangle$.

- 第 1 种情况:随机员工分配

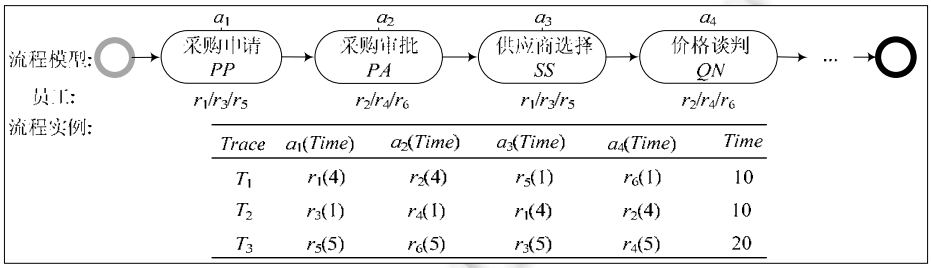
活动 a_1:员工 r_5;活动 a_2:员工 r_6;活动 a_3:员工 r_3;活动 a_4:员工 r_4;即

$$es=\langle a_1|r_5;a_2|r_6;a_3|r_3;a_4|r_4\rangle, Compatibility_{es}=0.5\times0.034+0.5\times0.034+0.5\times0.034=0.052;$$

- 第 2 种情况:基于最大协作度的员工分配

活动 a_1:员工 r_1;活动 a_2:员工 r_2;活动 a_3:员工 r_5;活动 a_4:员工 r_6;即

$$es=\langle a_1|r_1;a_2|r_2;a_3|r_5;a_4|r_6\rangle, Compatibility_{es}=0.5\times0.209+0.5\times0.841+0.5\times0.991=1.020.$$



Trace	$a_1(Time)$	$a_2(Time)$	$a_3(Time)$	$a_4(Time)$	Time
T_1	$r_1(4)$	$r_2(4)$	$r_5(1)$	$r_6(1)$	10
T_2	$r_3(1)$	$r_4(1)$	$r_1(4)$	$r_2(4)$	10
T_3	$r_5(5)$	$r_6(5)$	$r_3(5)$	$r_4(5)$	20

Fig.2 Example of execution traces of purchasing process in manufacturing

图 2 制造业采购流程的执行轨迹示例

Table 2 Compatibility table of two activities ($coop_{a_1,a_2}$)

表 2 活动-活动协作度表($coop_{a_1,a_2}$)

活动→活动	协作度
$a_1\rightarrow a_2$	0.5
$A_2\rightarrow a_3$	0.5
$A_3\rightarrow a_4$	0.5

Table 3 Compatibility table of two entities ($compatibility_{a_1|r_1,a_2|r_2}$)

表 3 实体-实体协作度表($compatibility_{a_1|r_1,a_2|r_2}$)

实体→实体	协作度	实体→实体	协作度
$a_1\|r_1\rightarrow a_2\|r_2$	0.209	$a_2\|r_4\rightarrow a_3\|r_1$	0.841
$a_1\|r_3\rightarrow a_2\|r_4$	0.991	$a_2\|r_2\rightarrow a_3\|r_3$	0.841
$a_1\|r_5\rightarrow a_2\|r_6$	0.034	$a_3\|r_1\rightarrow a_4\|r_2$	0.209
$a_2\|r_6\rightarrow a_3\|r_3$	0.034	$a_3\|r_5\rightarrow a_4\|r_6$	0.991
$a_2\|r_2\rightarrow a_3\|r_5$	0.841	$a_3\|r_3\rightarrow a_4\|r_4$	0.034
…	…	…	…

从这个示例中可以看出,这两种不同的分配方法得到的整体协作度差别很大,说明了最优员工分配研究对流程执行性能的重要性.

3 协作最优的员工分配方法

基于最大协作度(协作最优)的员工分配方法 FOSA(fast optimal staff assignment)总体框架如图 3 所示,其整体过程可概述如下.

(1) 首先,预处理历史流程执行日志得到标准事件日志,并将一条条的流程实例转换成实体轨迹;然后,再使用 Needleman-Wunsch 算法[21]对齐实体轨迹得到对齐矩阵.

(2) 其次,根据对齐的实体轨迹矩阵和协作度计算方法可以得到活动-活动、实体-实体的协作度表(见表 2 和表 3);然后,基于 Apriori 算法思想实现高协作模式挖掘(后文算法 1),以得到代表高协作能力的员工分配模式表.

(3) 最后,根据步骤(1)、步骤(2)中得到的活动-活动协作度表、实体-实体协作度表以及高协作模式表,使用本文提出的协作最优员工分配算法(后文算法 2)可以实现快速、有效的员工分配.

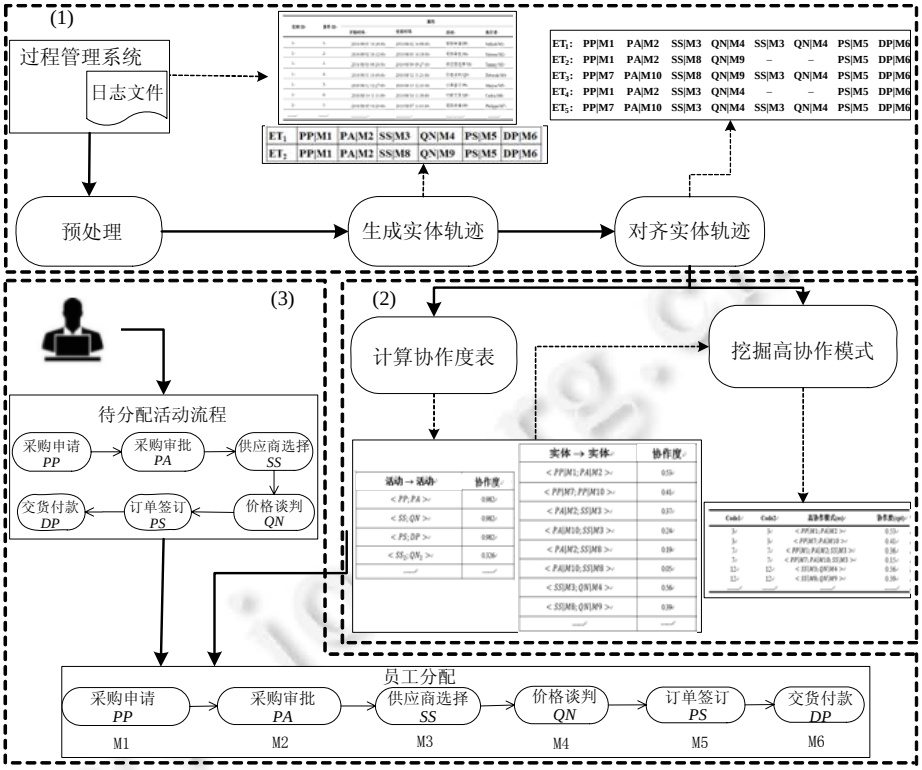


Fig.3 Framework of staff assignment approach for maximizing compatibility

图 3 基于最大协作度的员工分配方法框架

3.1 预处理日志文件和对齐实体轨迹

流程管理系统记录的流程执行日志文件中往往存在着由不当操作造成的噪声数据和大量的冗余数据.本文通过对日志中关键属性值的提取、时间戳属性的标准化、异常值的剔除等一些基本的预处理操作可以得到标准的事件日志.然后再根据上文的定义,将事件日志转换成一条条的实体轨迹.

本文使用经典的 Needleman-Wunsch 算法[21]实现实体轨迹对齐.

3.2 计算协作度和挖掘高协作模式

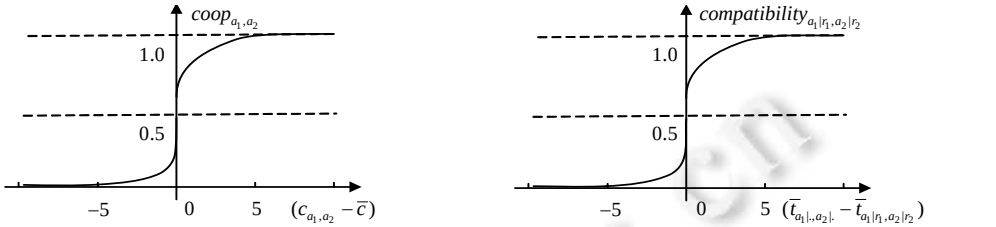
为了度量分配员工后的整个流程的协作度(即实体序列协作度),根据公式(4),本文提出一种方法来计算需要协作完成的活动-活动之间的协作度以及在协作完成活动时的实体-实体之间的协作度.另外,从一些复杂的(有循环、分支结构)或者未知的流程模型运行日志中生成的实体轨迹个体之间差异较大,为了能够挖掘得到过去的最佳实践的员工分配,基于对齐的实体轨迹,本文提出了挖掘高协作模式的算法.

3.2.1 计算协作度

为了度量两个活动之间的协作度 $coop_{a_1,a_2}$ (公式(1)),本文通过在基础 *Sigmoid* 函数中加入参数 k,将某两个连续的活动 $a_1>_L a_2$ 在事件日志 L 的轨迹中出现的频繁度 c_{a_1,a_2} 与所有不同类型的两个连续活动出现的平均频繁度 $\bar{c}$ 之间的差值作为一个变量映射到 0-1 之间.根据这个协作度值,可以形象化地表示协作水平的高低:如两个活动之间的协作度值为 0,则表示这两个活动之间没有任何的依赖关系和交互;值为 1,则表示这两个活动之间具有很强的协作关系.同理,为了度量两个员工 r_1,r_2 在分别执行两个活动 $a_1>_L a_2$ 时的协作能力,即两实体 $a_1|r_1>_L$

$a_2|r_2$之间的协作度 $compatibility_{a_1|r_1,a_2|r_2}$ (公式(3)),将事件日志 L 中协作执行活动 a_1,a_2 的平均所需时间 $\overline{t}_{a_1|.,a_2|.}$ 与员工 r_1,r_2 执行活动 a_1,a_2 的平均所需时间 $\overline{t}_{a_1|r_1,a_2|r_2}$ 的差值作为变量,使用变形的 *Sigmoid* 函数将其映射到 0-1 之间.如两个实体之间的协作度值为 0,则表示协作执行这两个活动的员工之间交互能力很差;值为 1,则表示他们之间的协作能力很强、执行活动的效率很高.

图 4 分别表示 $coop_{a_1,a_2}$ 和 $compatibility_{a_1|r_1,a_2|r_2}$ 的变化曲线(参数 k=1 时).



(a) 活动协作度 $coop_{a_1,a_2}$ 与 $(c_{a_1,a_2}-\overline{c})$ 的关系　　(b) 实体协作度 $compatibility_{a_1|r_1,a_2|r_2}$ 与 $(\overline{t}_{a_1|.,a_2|.}-\overline{t}_{a_1|r_1,a_2|r_2})$ 的关系

Fig.4　Variation curves of activity compatibility and entity compatibility

图 4　活动协作度和实体协作度的变化曲线

从图 4(a)中可以发现:频繁度 c_{a_1,a_2} 比均值 $\overline{c}$ 越大,则活动协作度 $coop_{a_1,a_2}$ 的值越接近 1,说明越频繁发生的两个连续活动的协作度就越高.同样地,从图(b)中可以发现:平均时间 $\overline{t}_{a_1|r_1,a_2|r_2}$ 与所有实体平均时间 $\overline{t}_{a_1|.,a_2|.}$ 的差值越小,则实体协作度 $compatibility_{a_1|r_1,a_2|r_2}$ 越接近于 1,说明执行时间越短的两个实体的协作度越高.另外,通过图 4 可以发现:在 $(c_{a_1,a_2}-\overline{c})$ 与 $(\overline{t}_{a_1|.,a_2|.}-\overline{t}_{a_1|r_1,a_2|r_2})$ 的四分之一和四分之三的位置时,$coop_{a_1,a_2}$ 与 $compatibility_{a_1|r_1,a_2|r_2}$ 的变化最明显.为了使得协作度能更准确地度量协作能力,协作度计算公式就要敏感地响应变量的变化.因此,本文使用 c_{a_1,a_2} 与 $\overline{c}$ 、 $\overline{t}_{a_1|.,a_2|.}$ 与 $\overline{t}_{a_1|r_1,a_2|r_2}$ 的差值序列的四分之一处的值 q_1、四分之三处的值 q_3,通过 k=10/(q_3−q_1)来计算 k 的值.

为了快速计算实体序列协作模式,确定公式中的参数 k 后,需要将事件日志 L 中所有具有协作关系的活动的协作度以及所有协作执行的实体的协作度进行计算并存储,最后可以得到一个所有类型的活动-活动的协作度表和一个所有类型的实体-实体的协作度表(与第 2.2 节类似).

3.2.2　挖掘高协作模式

为了挖掘历史日志中协作度较高的员工分配,本节基于对齐的实体轨迹实现了算法 1.该算法首先计算所有的活动序列及其协作度(第 2 行~第 7 行)、所有的实体序列及其协作度(第 8 行~第 10 行);然后,通过最小协作支持度阈值过滤得到具有高协作度的活动序列;再在这些活动序列中通过最小协作置信度阈值过滤得到具有高协作度的子实体序列即高协作模式(第 11 行~第 17 行).

算法 1. 挖掘高协作模式.

输入:$AM_{n\times m}$　//n 条实体轨迹对齐得到的矩阵;

　　CT_{op}　//所有不同类型的活动-活动之间的协作度表;

　　CT_{ep}　//所有不同类型的实体-实体之间的协作度表;

　　min_*cpt_sup*　//最小协作支持度阈值;

　　min_*cpt_conf*　//最小协作置信度阈值;

输出:$HP\langle es,cpt\rangle$　//高协作模式及其协作度.

01:　初始化 *colIndexList*[*n*],*actSeqMap*⟨*actSeq*,*cpt*⟩,*entitySeqMap*⟨*entitySeq*,*cpt*⟩

02:　FOR i=1 to n DO

03:　　*colIndexList*[i]←矩阵 $AM_{n\times m}$ 中第 i 行所有非“–”的列下标

```
04:  END FOR
     //存储所有不同类型的活动序列及其协作度的映射
05:  FOR i=1 to n DO
       //根据列下标 colIndexList 得到所有连续的活动序列及其协作度
06:    actSeqMap⟨actSeq,cpt⟩←getActivitySeq_cpt(colIndexList[i],CT_op)
07:  END FOR
     //存储所有不同类型的实体序列及其协作度的映射
08:  FOR each ⟨actSeq,cpt⟩ of actSeqMap DO
       //根据 actSeqMap 得到所有连续的实体序列及其协作度
09:    entitySeqMap⟨entitySeq,cpt⟩←getEntitySeq_cpt(actSeq,CT_ep)
10:  END FOR
11:  FOR each ⟨actSeq,cpt⟩ of actSeqMap DO
12:    IF actSeqMap.cpt≥min_cpt_sup THEN
13:      FOR each ⟨entitySeq,cpt⟩ of entitySeqMap corresponding to actSeq DO
14:        IF entitySeqMap.cpt≥min_cpt_conf THEN
15:          HP⟨es,cpt⟩←⟨entitySeq,cpt⟩
16:      END FOR
17:  END FOR
```

基于算法1得到的高协作模式⟨*es*,*cpt*⟩(见定义11),本文提出了如图5所示的两种编码方式(以上文提到的制造业采购流程为例).

预处理	
对齐的实体轨迹	ET_1: PP|M1 PA|M2 SS|M3 QN|M4 SS|M3 QN|M4 PS|M5 DP|M6 ET_2: PP|M1 PA|M2 SS|M8 QN|M9 – – PS|M5 DP|M6 ET_3: PP|M7 PA|M10 SS|M8 QN|M9 SS|M3 QN|M4 PS|M5 DP|M6 ET_4: PP|M1 PA|M2 SS|M3 QN|M4 – – PS|M5 DP|M6 ET_5: PP|M7 PA|M10 SS|M3 QN|M4 SS|M3 QN|M4 PS|M5 DP|M6
参考活动序列	<*PP PA SS QN* SS_2 QN_2 *PS DP*>
活动编码	$\mathbf{2^0}$ $\mathbf{2^1}$ $\mathbf{2^2}$ $\mathbf{2^3}$ $\mathbf{2^4}$ $\mathbf{2^5}$ $\mathbf{2^6}$ $\mathbf{2^7}$

二进制编码	
待分配活动流程	<*PP*;*PA*;*SS*;*QN*;*PS*;*DP*>
二进制编码值	$\mathbf{2^0 + 2^1 + 2^2 + 2^3 + 2^6 + 2^7 = 207}$
二进制编码	1 1 1 1 0 0 1 1

(a) 二进制编码

伴随二进制编码	
高协作模式	<*SS*|*M3*;*QN*|*M4*;*PS*|*M5*;*DP*|*M6*>
二进制编码	0 0 1 1 0 0 1 1
伴随二进制编码	0 0 1 1 1 1 1 1

(b) 伴随二进制编码

Fig.5 Two kinds of encoding: binary encoding and adjoint binary encoding

图5 两种不同的编码方式:二进制编码和伴随二进制编码

(a) 二进制编码:首先,根据实体轨迹对齐得到的参考活动序列(如图5中的⟨*PP*;*PA*;*SS*;*QN*;SS_2;*QN*;SS_2;QN_2;*PS*;*DP*⟩)对流程中的所有活动按位置进行编码(如第1个活动*PP*的编码为2^0,第2个活动*PA*的编码为2^1,…,以此类推);然后,根据活动编码可将待分配活动流程⟨*PP*;*PA*;*SS*;*QN*;*PS*;*DP*⟩进行编码,得到一个编码序列11110011和编码值207(如图5所示).对高协作模式进行二进制编码,即根据其对应的活动序列进行编码,如高协作模式对应的活动序列为⟨*SS*;*QN*;*PS*;*DP*⟩,我们可以得到其二进制编码序列

为 00110011.

(b) 伴随二进制编码:将高协作模式的二进制编码(如 00110011)中相邻的两个 1 中的 0 全部取反,即可得到它的伴随二进制编码(即 00111111).

基于事件日志挖掘出来的每个高协作模式,可分别得到它的二进制编码值 *code*1 和伴随二进制编码值 *code*2.每个协作模式用一个四元组⟨*code*1,*code*2,*es*,*cpt*⟩表示,挖掘得到的所有协作模式存在一个表中,即高协作模式表,见表 4.

Table 4 Example of high collaboration patterns

表 4 高协作模式的示例

*Code*1	*Code*2	高协作模式(*es*)	协作度(*cpt*)
3	3	⟨*PP*|*M*1;*PA*|*M*2⟩	0.53
3	3	⟨*PP*|*M*7;*PA*|*M*10⟩	0.41
7	7	⟨*PP*|*M*1;*PA*|*M*2;*SS*|*M*3⟩	0.36
7	7	⟨*PP*|*M*7;*PA*|*M*10;*SS*|*M*3⟩	0.15
12	12	⟨*SS*|*M*3;*QN*|*M*4⟩	0.56
12	12	⟨*SS*|*M*8;*QN*|*M*4⟩	0.39
…	…	…	…

最后,根据一定的匹配规则(如图 5 所示),可以快速确定可与待分配活动流程匹配的高协作模式.这些高协作模式将作为员工分配的候选方案.

3.3 员工分配算法

为了找到能使待分配活动流程达到最大协作度的最优员工分配方案,基于第 3.2 节中得到的员工分配的候选方案(候选方案中的高协作模式都是子实体序列,只能作为部分活动的候选分配方案),本文定义了如公式(5)所示的目标函数:

$$\text{Minimize}\sum\nolimits_{\forall(a_1|r_1,a_2|r_2)\in entityProList}(1-coop_{a_1,a_2}\times compatibility_{a_1|r_1,a_2|r_2}) \tag{5}$$

其中,*entityProList* 表示已经分配员工的活动流程(即实体序列).根据该函数,可从候选方案中选择使得整体协作度最大的一种分配方案.具体实现如算法 2 所示.

- 首先,对待分配流程进行二进制编码并记录当前未分配的活动序列的编码(第 1 行、第 2 行);
- 然后,根据图 6 的匹配规则快速得到能作为候选的员工方案的高协作模式(第 4 行~第 8 行);
- 最后,以整体协作度最大为目标,在已有候选分配方案的部分活动的基础上,通过遍历协作度表 CT_{ap} 和 CT_{ep} 对没有候选分配方案的活动子序列选择一种协作度最大的分配方案(第 9 行~第 16 行).

如图 6 所示,高协作模式 1 能与待分配的活动流程成功匹配,说明它可以作为候选分配方案对 *SS*(供应商选择)、*QN*(价格谈判)、*PS*(订单签订)、*DP*(交货付款)活动进行分配.

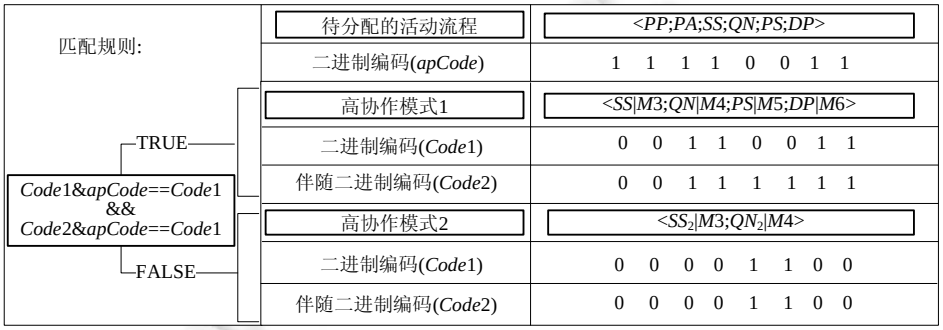


Fig.6 Matching rule between high collaboration patterns and activity process

图 6 高协作模式和活动流程的匹配规则

算法 2. 协作最优员工分配.

输入:$M_{es}\langle code1,code2,HP\rangle$ //高协作模式表;

$aP=\langle a_1;a_2;\ldots;a_i\rangle(a_i\in A_L)$ //待分配活动流程(活动序列);

$CT_{op}\langle activity1,activity2,cpt\rangle$ //所有类型的活动-活动协作度表;

$CT_{ep}\langle entity1,entity2,cpt\rangle$ //所有类型的实体-实体协作度表;

输出:$entityProList=\langle a_1|r_1;a_2|r_2;\ldots;a_i|r_i\rangle(a_i\in A_L,r_i\in R_L)$ //分配员工的活动流程(实体序列).

01: $apCode\leftarrow enCode(aP)$ //将待分配活动流程进行二进制编码

02: $cuCode\leftarrow apCode$ //记录当前待分配活动流程的编码

03: 根据高协作模式表 M_{es} 中相同的编码值 $code1$ 按照 cpt 的值进行降序排列

04: FOR each element $\langle code1,code2,HP\rangle$ of M_{es} DO

05: IF $code2\&apCode==code1\&\&code1\&cuCode==code1$ THEN

//根据 M_{es} 中匹配成功的模式更新分配序列 $entityProList$

06: $entityProList\leftarrow\langle code1,code2,HP\rangle$ of M_{es}

07: $cuCode\leftarrow cuCode\&(\sim code1)$ //更新待分配活动流程中的未分配活动序列

08: END FOR

09: $segProList\langle startIdx,endIdx\rangle\leftarrow deCode(cuCode)$ //存储未分配的活动子序列

10: FOR each element $\langle startIdx,endIdx\rangle$ of $segProList$ DO

11: $foreIdx\leftarrow startIdx-1$; $tailIdx\leftarrow endIdx+1$

12: $s_1\leftarrow entityProList\langle startIdx\rangle$; $s_2\leftarrow entityProList\langle endIdx\rangle$

13: $s_1'\leftarrow searchMaxCptFrom\langle CT_{ap},CT_{ep},s_1\rangle$

14: $s_2'\leftarrow searchMaxCptTo\langle CT_{ap},CT_{ep},s_2\rangle$

15: $entityProList\leftarrow searchMaxCpt(CT_{ap},CT_{ep},\langle foreIdx,tailIdx\rangle,s_1',s_2')$

16: END FOR

4 实验与验证

为了验证本文提出的员工分配方法 FOSA 的可行性和有效性,分别在两个不同的数据集(合成数据集和真实数据集)上,根据实体之间的协作度对比,使用不同的分配算法对所有可能出现的待分配活动流程进行员工分配,最后对比分配后的协作度以及分配所需的时间.对比的算法主要有贪婪启发式分配算法 GHWA(greedy heuristic work assignment)、随机分配算法 RWA(random work assignment)以及暴力求解算法 BFWA(brute forth work assignment).

另外,为了验证本文基于实体间协作度进行员工分配的效果比仅考虑员工之间协作度进行分配的效果更好,在第 4.3 节中使用真实数据集,分别基于实体之间的协作度计算(CMEA)和基于员工之间的协作度(CMA)[6]计算,并使用 BFWA 算法(消除分配算法不同而带来的影响)分配员工,然后对比分配后的效果.上述算法均使用 Java 语言开发实现,实验运行环境是 Windows 7+Intel Xeon E5-2620+2.00GHz 主频+64GB 内存.

4.1 合成数据集

本节使用 CPNTools (http://cpntools.org/)工具,根据着色 Petri 网合成了一个软件开发流程(如图 7 所示)的日志数据集,包含 10 000 条流程实例、106 500 个活动、21 种不同的实体.

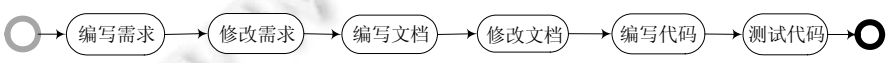


Fig.7 Software development process model

图 7 软件开发流程模型

该流程中的某些复杂活动如编写代码(WriteCode)经常由多个员工共同协作完成,针对这种情况,本文将完成同一个活动的员工看作一个整体,使用实体(编码|员工 1,员工 2,员工 3)表示,然后,在此基础上计算协作度.在该数据集上,将本文提出的 FOSA 算法与其他 3 种算法(RWA,GHWA,BFWA)的实验结果进行对比,结果见表 5.该表统计了在计算实体间协作度的基础上,使用不同的分配算法对所有可能出现的待分配活动流程(即活动个数范围为 2~6)进行协作最优员工分配,最后得到分配方案对应的协作度以及分配所需时间的平均值.从表 5 中可以看出,综合考虑协作度和算法运行时间,本文提出的 FOSA 算法是最优的.

Table 5 Comparison results of four staff assignment methods based on synthetic dataset

表 5 基于合成数据集的 4 种员工分配算法结果对比

活动个数	平均协作度				算法平均运行时间(ms)			
	RWA	GHWA	BFWA	FOSA	RWA	GHWA	BFWA	FOSA
2	0.08	0.73	0.73	0.73	0.00	0.02	0.293 7	0.018
3	0.18	1.54	1.57	1.57	0.00	0.02	0.136 9	0.027
4	0.28	2.45	2.45	2.45	0.00	0.04	0.227 0	0.051
5	0.33	3.25	3.27	3.27	0.00	0.04	0.908 8	0.199
6	0.37	4.40	4.41	4.41	0.00	0.06	3.792 4	0.867

4.2 真实数据集

本文使用的真实数据集是由 Disco(https://fluxicon.com/disco/)提供的一个复杂采购流程(如图 8 所示,与图 1 不同)日志.该日志包含 99 条实例、24 种不同的活动、27 名员工,活动和员工构成的实体类型为 145 种(其中,每种类型的活动都可由 2 个~14 个员工执行).与第 4.1 节中进行相同的实验,结果见表 6(由于对比的 BFWA 算法的时间随着分配活动个数的增加会变得非常大,因此只统计了活动个数为 2~10 的待分配活动流程).

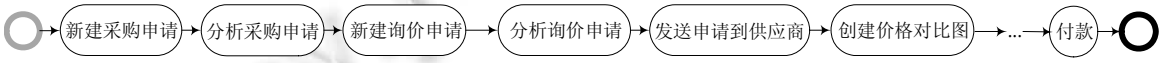


Fig.8 Complex purchasing process model

图 8 复杂采购流程模型

Table 6 Comparison results of four staff assignment methods based on real dataset

表 6 基于真实数据集的 4 种员工分配算法结果对比

活动个数	平均协作度				算法平均运行时间(ms)			
	RWA	GHWA	BFWA	FOSA	RWA	GHWA	BFWA	FOSA
2	0.01	0.22	0.22	0.22	0.000	0.421	0.134	0.625
3	0.02	0.62	0.64	0.64	0.001	0.171	0.605	0.566
4	0.09	0.95	0.95	0.95	0.000	0.198	1.086	0.853
5	0.05	1.32	1.42	1.42	0.000	0.112	3.803	0.408
6	0.21	2.18	2.21	2.21	0.000	0.191	27.315	0.444
7	0.22	2.82	2.93	2.93	0.001	0.072	172.843	0.458
8	0.17	3.12	3.19	3.19	0.001	0.067	767.944	0.860
9	0.31	3.38	3.55	3.53	0.001	0.085	2 558.186	1.574
10	0.25	3.72	3.84	3.84	0.001	0.086	2 806.143	5.247

从表 6 中可以发现,FOSA 算法的执行结果非常接近 BFWA 算法的最优分配结果(理论上最优),而时间上却有绝对的优势,因此可以说明本文的方法能够实现快速而有效的员工分配.为了更好地对比算法效果,本文使用堆积柱状图展示随着待分配活动流程中活动个数的增加,FOSA 算法与 RWA,GHWA 算法对每一个待分配流程个体分配的结果(协作度)与 BFWA 算法所得的理论上最大的协作度之间的差值分布情况,如图 9 所示.我们首先将协作度差值划分为 5 类,分别为 0,0~1,1~2,2~3,3~7,然后统计使用不同的分配方法将待分配活动流程个体分配结果的协作度与理论上能取得的最大协作度(BFWA 算法)的差值分布.从图中可以看出,使用本文提出的 FOSA 算法进行员工分配时的协作度几乎能够达到理论的最优值;而另外两种分配算法随着活动个数的增加,分配结果与理论最优的分配方案的协作度差值越来越大.

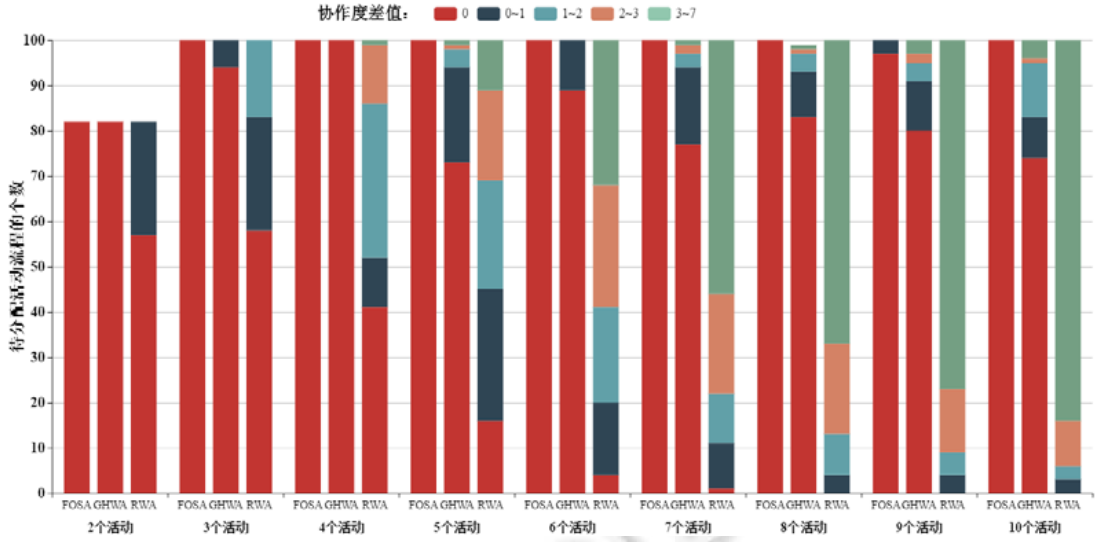


Fig.9 Comparison results of three staff assignment methods for activity processes with different number of activities

图 9 使用 3 种不同的算法对不同活动个数的活动流程分配员工的结果比较

4.3 有效性验证

为了验证本文提出的根据实体之间的协作度(CMEA)进行员工分配的结果比根据员工之间的协作度(CMA)进行分配的结果更合理,本文在真实数据集上分别基于这两种不同的协作度计算方法,对所有可能出现的待分配活动流程(按流程中的活动个数分类)使用 BFWA 算法进行员工分配,然后比较每条待分配活动流程在两种不同的员工分配结果时,整个流程的执行时间(参考历史日志中员工执行活动的平均时间),如图 10 所示.

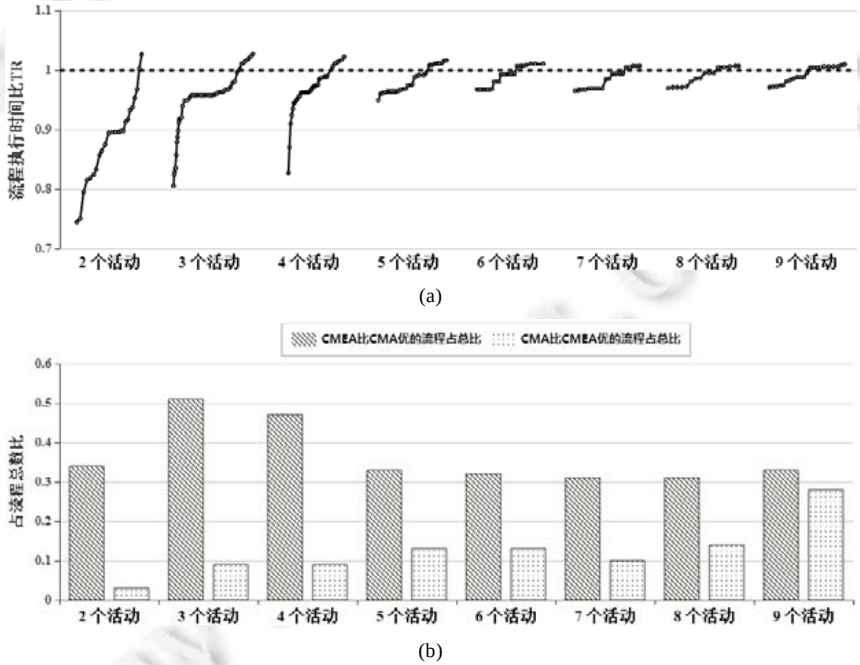


Fig.10 Comparison results of staff assignment between CMEA and CMA

图 10 基于实体协作度(CMEA)和基于员工协作度(CMA)的员工分配结果比较

图 10(a)中的折线图表示不同活动个数的所有待分配活动流程,分别基于 CMEA+BFWA 得到的员工分配方案对应的流程执行时间与基于 CMA+BFWA 的分配方案对应的流程执行时间的比值(记作“流程执行时间比”,用 *TR*=*Time*(*CMEA*+*BFWA*)/*Time*(*CMA*+*BFWA*)表示)的分布情况.图中的每个折线图上有很多个数值点,每个数值点代表一条待分配活动流程,横坐标值代表流程中的活动个数,纵坐标值代表两种不同方案得到的流程执行时间比,图中的加粗黑色虚线代表 *TR* 等于 1,即两种分配方案的效果一样.从图中可以发现:不管活动个数为多少,*TR* 小于 1(即 CMEA+BFWA 的流程执行时间比 CMA+BFWA 流程执行时间少)的数值点个数比 *TR* 大于 1(即 CMEA+BFWA 的流程执行时间比 CMA+BFWA 流程执行时间多)的数值点要多.这说明大多数情况下,基于实体协作度 CMEA 进行员工分配后的流程执行时间比基于员工协作度 CMA 进行员工分配后的流程执行时间要少,进一步说明了基于实体协作度进行员工分配的效果更好.

图 10(b)中的柱状图是对图 10(a)中的流程执行时间比 *TR* 的进一步分析,它表示不同活动个数的所有待分配活动流程中执行时间比 *TR* 小于 1(即 CMEA 比 CMA 优)、*TR* 大于 1(即 CMA 比 CMEA 优)的流程个数占流程总数的比值分布情况.从图中可以发现:不管待分配流程中的活动个数是多少,CMEA 比 CMA 优的流程占总比的值都要比 CMA 比 CMEA 优的流程占总比的值大.这进一步说明了使用实体协作度计算进行员工分配比考虑员工之间的协作度的分配效果要好.

5 结论和未来工作

本文提出了一种通过参考历史日志中的高协作模式实现最大协作度的员工分配方法.特别地,在计算员工与员工之间的协作度时也考虑了活动本身对其协作水平的影响.本文提出了两种编码方式,用于快速匹配待分配的活动流程和挖掘得到的高协作模式,并将匹配成功的模式作为员工分配的候选方案,然后从中选出使得整个流程的协作度达到最大的一种员工分配方案.大量的对比实验,验证了该方法的可行性和有效性.

由于对比实验选择了暴力求解算法,这使得实验中待分配活动流程中的流程个数必须较少,否则无法运行.未来我们准备基于其他算法、使用活动个数更多的流程来验证本文方法.此外,由于挖掘得到的高协作模式的质量高低会影响后续员工分配方案的选择和效率,所以需要进一步通过实验对比来选择最合适的协作模式挖掘算法.最后,我们也将基于 A*算法的思想,通过综合考虑各个因素研究工作流最优员工分配问题.

References:

[1] Noll J, Beecham S, Richardson I. Global software development and collaboration: Barriers and solutions. Association for Computing Machinery, 2010,1(3):66−78. [doi: 10.1145/1835428.1835445]

[2] Magdaleno AM. A maturity model to promote collaboration in business processes. Journal of Business Process Integration & Management, 2009,4(2):111−123. [doi: 10.1504/IJBPIM.2009.027779]

[3] Liu TY, Cheng YL, Ni ZH. Mining event logs to support workflow resource allocation. Knowledge-Based Systems, 2012,35(15): 320−331. [doi: 10.1016/j.knosys.2012.05.010]

[4] Aalst WVD. Process mining. Communications of the ACM, 2012,55(8):76−83. [doi: 10.1145/2240236.2240257]

[5] Bose RPJC, Aalst WVD. Trace alignment in process mining: Opportunities for process diagnostics. In: Hull R, Mendling J, Tai S, eds. Proc. of the 2010 Int'l Conf. on Business Process Management. Berlin: Springer-Verlag, 2010. 227−242. [doi: 10.1007/978-3-642-15618-2_17]

[6] Kumar A, Dijkman RM, Song M. Optimal resource assignment in workflows for maximizing cooperation. In: Daniel F, Wang J, Weber B, eds. Proc. of the 2013 Int'l Conf. on Business Process Management. Berlin: Springer-Verlag, 2013. 235−250. [doi: 10.1007/978-3-642-40176-3_20]

[7] Xie Y, Chien CF, Tang RZ. A method for estimating the cycle time of business processes with many-to-many relationships among the resources and activities based on individual worklists. Computers & Industrial Engineering, 2013,65(2):194−206. [doi: 10.1016/j.cie.2013.02.015]

[8] Xie Y, Chien CF, Tang RZ. A dynamic task assignment approach based on individual worklists for minimizing the cycle time of business processes. Computers & Industrial Engineering, 2015,99:401−414. [doi: 10.1016/j.cie.2015.11.023]

[9] Bessai K, Charoy F. Business process tasks-assignment and resource allocation in crowdsourcing context. In: Proc. of the 2016 IEEE Int'l Conf. on Collaboration and Internet Computing. New York: IEEE, 2016. 11−18. [doi: 10.1109/CIC.2016.016]

[10] Reijers HA, Jansenvullers MH, Muehlen MZ, Appl W. Workflow management systems+swarm intelligence=dynamic task assignment for emergency management applications. In: Alonso G, Dadam P, Rosemann M, eds. Proc. of the 2007 Int'l Conf. on Business Process Management. Berlin: Springer-Verlag, 2007. 125−140. [doi: 10.1007/978-3-540-75183-0_10]

[11] Ly LT, Rinderle S, Dadam P, Reichert M. Mining staff assignment rules from event-based data. In: Bussler CJ, Haller A, eds. Proc. of the 2005 Business Process Management Workshops. Berlin: Springer-Verlag, 2005. 177−190. [doi: 10.1007/11678564_16]

[12] Liu YB, Wang JM, Yang Y, Sun JG. A semi-automatic approach for workflow staff assignment. Computers in Industry, 2008,59(5): 463−476. [doi: 10.1016/j.compind.2007.12.002]

[13] Xu RB, Liu X, Xie Y, Yuan D, Yang Y. A Gaussian fields based mining method for semi-automating staff assignment in workflow application. In: Proc. of the 2014 Int'l Conf. on Software and System Process. New York: ACM Press, 2014. 178−182. [doi: 10.1145/2600821.2600843]

[14] Cheng H, Chu XN. Task assignment with multiskilled employees and multiple modes for product development projects. Journal of Advanced Manufacturing Technology, 2012,61(1-4):391−403. [doi: 10.1007/s00170-011-3686-7]

[15] Liu TY, Cheng YL, Ni ZH. Mining event logs to support workflow resource allocation. Knowledge-Based Systems, 2012,35(15): 320−331. [doi: 10.1016/j.knosys.2012.05.010]

[16] Xu RB, Bao GH, Yang PQ, Wang XM, Xie Y. Staff collective optimization strategy based on maximal dependency and minimal redundancy. Journal of Computer Integrated Manufacturing Systems, 2017,23(5):1014−1019 (in Chinese with English abstract). [doi: 10.13196/j.cims.2017.05.012]

[17] Meddah I, Khaled B. Discovering patterns using process mining. Journal of Rough Sets & Data Analysis, 2016,3(4):21−31. [doi: 10.4018/IJRSDA.2016100102]

[18] Smirnov S, Weidlich M, Mendling J, Weske M. Action patterns in business process model repositories. Computers in Industry, 2009,63(2):115−129. [doi: 10.1016/j.compind.2011.11.001]

[19] Verginadis Y, Papageorgiou N, Apostolou D, Mentzas G. A review of patterns in collaborative work. In: Proc. of the 16th ACM Int'l Conf. on Supporting Group Work (Group 2010). New York: ACM Press, 2010. 283−292. [doi: 10.1145/1880071.1880118]

[20] Diamantini C, Genga L, Potena D, Storti E. Discovering behavioral patterns in knowledge-intensive collaborative processes. In: Appice A, Ceci M, Loglisci C, Manco G, Masciari E, Ras Z, eds. Proc. of the New Frontiers in Mining Complex Patterns. Berlin: Springer-Verlag, 2015. 149−163. [doi: 10.1007/978-3-319-17876-9_10]

[21] Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 1970,48(3):443−453. [doi: 10.1016/0022-2836(70)90057-4]

附中文参考文献:

[16] 许荣斌,鲍广华,杨培全,汪欣梅,谢莹.基于最大依赖度及最小冗余度的员工协作优化策略.计算机集成制造系统,2017,23(5): 1014−1019. [doi: 10.13196/j.cims.2017.05.012]

俞东进(1969－),男,浙江平湖人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程理论和方法,业务过程管理,行业大数据.

王娇娇(1992－),女,博士生,CCF 学生会员,主要研究领域为业务过程管理,数据挖掘,大数据可视化.

柳诚飞(1961－),男,博士,教授,博士生导师,主要研究领域为 XML,数据库系统,工作流.