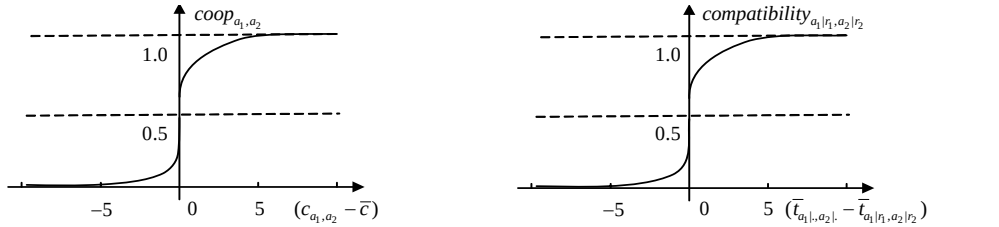
$a_2|r_2$之间的协作度 $compatibility_{a_1|r_1,a_2|r_2}$ (公式(3)),将事件日志 L 中协作执行活动 a_1,a_2 的平均所需时间 $\overline{t}_{a_1|.,a_2|.}$ 与员工 r_1,r_2 执行活动 a_1,a_2 的平均所需时间 $\overline{t}_{a_1|r_1,a_2|r_2}$ 的差值作为变量,使用变形的 *Sigmoid* 函数将其映射到 0-1 之间.如两个实体之间的协作度值为 0,则表示协作执行这两个活动的员工之间交互能力很差;值为 1,则表示他们之间的协作能力很强、执行活动的效率很高.

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: $actSeqMap\langle actSeq,cpt\rangle \leftarrow getActivitySeq\_cpt(colIndexList[i],CT_{op})$
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 $\langle actSeq,cpt\rangle$ of *actSeqMap* DO
//根据 *actSeqMap* 得到所有连续的实体序列及其协作度
09: $entitySeqMap\langle entitySeq,cpt\rangle \leftarrow getEntitySeq\_cpt(actSeq,CT_{ep})$
10: END FOR
11: FOR each $\langle actSeq,cpt\rangle$ of *actSeqMap* DO
12: IF $actSeqMap.cpt \geqslant \min\_cpt\_sup$ THEN
13: FOR each $\langle entitySeq,cpt\rangle$ of *entitySeqMap* corresponding to *actSeq* DO
14: IF $entitySeqMap.cpt \geqslant \min\_cpt\_conf$ THEN
15: $HP\langle es,cpt\rangle \leftarrow \langle entitySeq,cpt\rangle$
16: END FOR
17: END FOR

基于算法1得到的高协作模式$\langle es,cpt\rangle$(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$\langle PP;PA;SS;QN;SS_2;QN;SS_2;QN_2;PS;DP\rangle$)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$\langle PP;PA;SS;QN;PS;DP\rangle$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$\langle SS;QN;PS;DP\rangle$,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:

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

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.

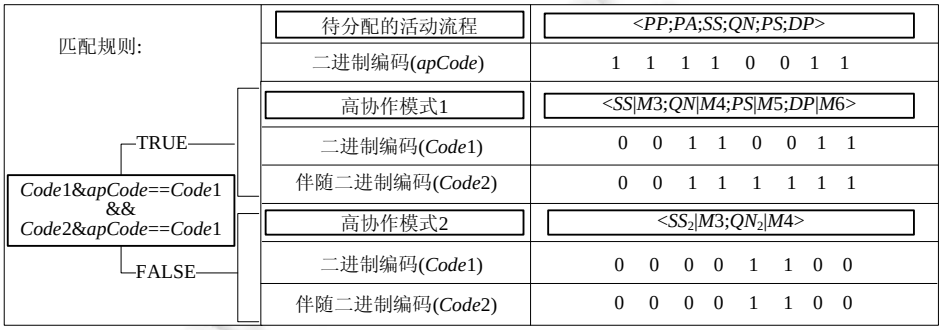


Fig.6 Matching rule between high collaboration patterns and activity process

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.

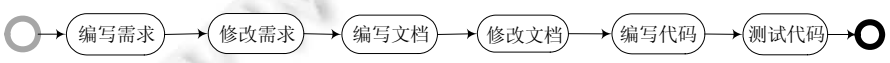


Fig.7 Software development process model

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).

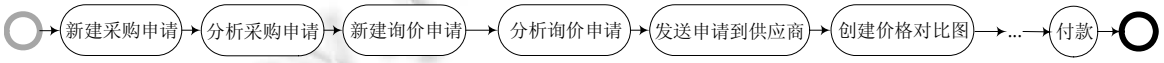


Fig.8 Complex purchasing process model

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.

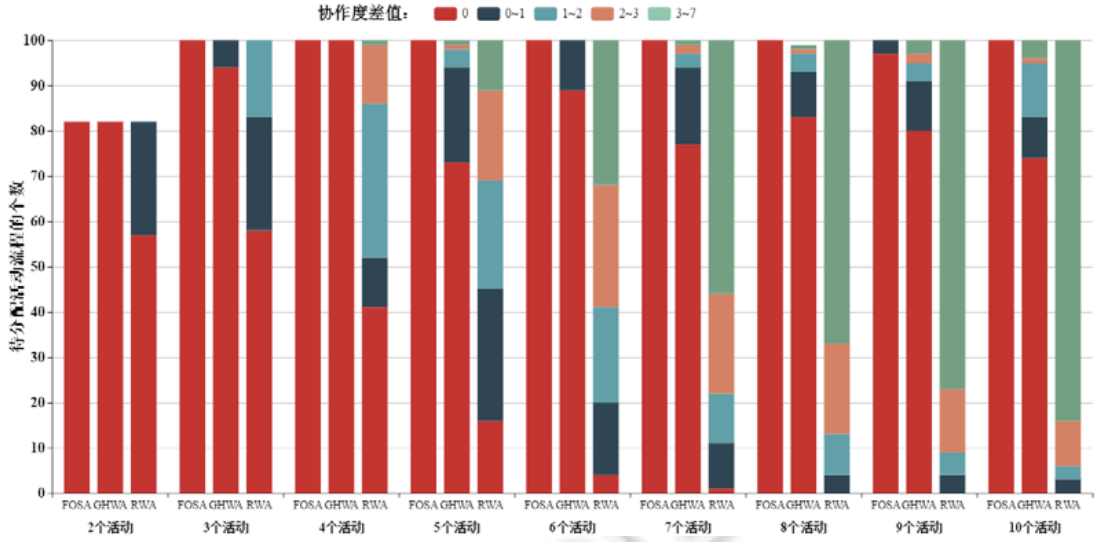


Fig.9 Comparison results of three staff assignment methods for activity processes with different number of activities

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.

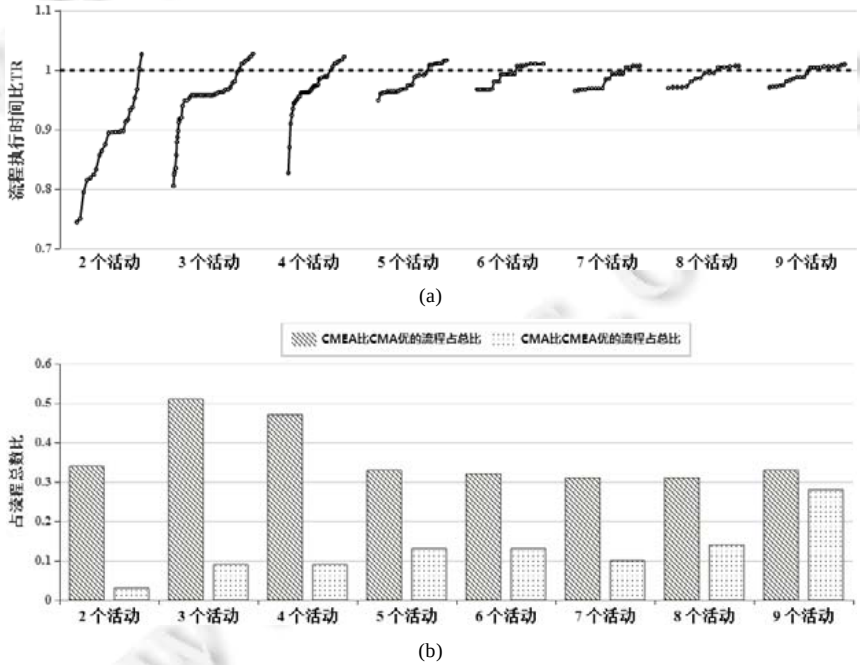


Fig.10 Comparison results of staff assignment between CMEA and CMA

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.