



基于条件化有向图的工作流过程优化

谢玉凤 杨光信 史美林

(清华大学计算机科学与技术系 北京 100084)

摘 要 为改善工作流的性能和效率,一个智能化的 workflow 管理系统应该具有分析和优化 workflow 过程定义的能力. 该文首先给出影响 workflow 过程性能的主要因素,借助于定量分析的方法,讨论了这些因素对 workflow 性能和效率的影响. 以此为基础,基于条件化有向图这种过程定义模型提出了一组对 workflow 过程进行优化的方法.

关键词 workflow 管理系统, 业务流过程优化(BPR), workflow 过程优化(WPR), 计算机支持的协同工作(CSCW)
中图法分类号: TP311

Conditional Directed Graph-Based Workflow Process Re-Engineering

HSIEH Yu-Feng YANG Guang-Xin SHI Mei-Lin

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

Abstract Nowadays workflow management technology is increasingly being exploited by businesses in a variety of industries. A workflow process is a formalized business process, which is a procedure where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal. The execution of a workflow process is conducted by workflow management system. To reduce the cost of the companies and improve their performance, Business Process Re-engineering (BPR) is put forward, where Re-engineering means improving some main performance units, such as cost, quality, service and so on, by finding out the process bottleneck and re-designing the basic component of business process. Because a workflow process is usually defined according to the users' own requirements, there may be some faults in it or it is unsatisfactory. An intelligent workflow management system should have the ability to analyze and re-engineer workflow process to improve performance and efficiency. This paper gives requirements of Workflow Process and based on the conditional directed graph (CDG) model, these requirements are analyzed and improved and as a result the performance of Workflow Process is also improved. Unlike the previous works, where qualitative analysis is used to analyze workflow process and we can not compare two processes explicitly, we use quantitative analysis. For we start the analysis from a general perspective, i.e. the data dependencies between activities, the resulting optimizing methods are common ones. Dynamic optimization can be achieved based on these methods. For specific applications, some extensions also can be done based on them. We believe that a good optimizing system can do some special optimization according to the concrete business process besides the general ones. This paper also addresses the new concept, Workflow Process Re-engineering (WPR), which focuses on static analyzing of data flow and control flow of workflow definitions for process optimizing, so

收稿日期: 2000-04-28; 修改稿收到日期: 2000-10-15. 本课题得到国家自然科学基金(69773029)和国家“八六三”高技术研究发展计划(863-306-ZD-10-2B, 863-306-ZD-02-03-1)资助. 谢玉凤, 女, 1976年生, 硕士, 主要研究领域为计算机支持的协同工作(CSCW)、人工智能、数据库系统、计算机网络和工作流优化. 杨光信, 男, 1973年生, 博士, 现为贝尔实验室中国基础研究院高级研究员, 主要研究领域为计算机支持的协同工作(CSCW)、群件、工作流管理等. 史美林, 男, 1938年生, 教授, 博士生导师, 主要研究领域为计算机支持的协同工作(CSCW)和计算机网络.

that system performance and cooperation efficiency can be improved.

Keywords workflow management system, business process re-engineering (BPR), workflow process re-engineering (WPR), CSCW

1 引 言

为了减少企业成本和增加性能,人们提出 BPR (Business Process Re-engineering) 的概念. BPR 主要是检测和分析组织之间的工作流和过程^[1,2]来找到它的瓶颈. Re-engineering 是指对业务流过程的基本组织进行重新考虑和重新设计从而改善一些重要的性能指标,例如成本、质量、服务和速度^[3]. 整个建立过程中,模拟执行和分析业务流过程通常会模拟执行好多次以保证达到最优的工作流过程^[4]. BPR 只是局限在分析和优化,有些 BPR 软件工具甚至只有分析的方法而没有提出优化的方法.

以往,企业所要解决的最主要问题是提高整个工作过程的效率和使工作变得有成效.但是,随着计算机和网络的广泛应用,企业比以前面临的问题更多,例如安全性、稳定性和协调性等.要解决这些问题不仅要考虑组织和部门之间的关系,而且也要研究所使用的技术和数据共享的方法(尤其是在协调性方面).于是, workflow 管理系统^[5]应该尽量达到这些需求;而且,用户是根据自己的需求定义出工作流,这个工作流或许存在一些不如人意的地方,我们就必须提出一套优化方法来改善这个工作流过程.我们已经发现一些工作流的特征可以使用 BPR 的方法来进行优化^[6];所以 workflow 管理系统应该开发支持 BPR 需要的工作流技术^[7],也就是优化的工具.这里我们提出工作流过程优化的概念(WPR),它是一套符合工作流过程特性的优化方法;不仅是对工作流效率的改善,还对某些工作节点提供协同的工具使得企业的协同工作更加完善.

工作流过程是一个业务任务的形式化表示并且由 workflow 管理系统自动操作和执行.在不同的 workflow 管理系统里面,用来描述工作流过程的形式化模型是不一样的.广泛使用的工作流模型包括有向图^[8]、PetriNets^[9]、面向对象的模型^[10]、通用过程结构文法(GPSG)^[11]、工作流语言^[12]和条件化有向图(CDG)^[13].这里所提出的工作流优化方法是基于 W o w w ! 提出的条件有向图^[13],下面给出

W o w w ! 工作流模型中形式化描述的方法,在后面的方法描述中将会使用到.

W o w w ! 工作流模型将工作流 w ^[13]形式化为一个三元组 $\langle n, A, F \rangle$,它是实际业务过程数学描述:

(1) n 为工作流的名称.

(2) A 表示从业务过程中抽象出来的所有活动构成的集合.

(3) F 为 $A \times A \times C$ 的一个子集,它描述的是 w 中各活动之间的数据流动关系与控制流动关系.对于 C 中的每一个元素 c ,它是一个二元组 $\langle c, E \rangle$,其中 c 为一布尔表达式, E 为多个命名表达式构成的集合. c 描述的是在条件 c_c 得以满足时激活后续的哪一个环节并将什么数据对象传给它.

w 实际上是一个有向图,图中各结点表示工作流中的各个活动步骤,同常规的有向图的区别在于:与每一有向边相应分别有一个逻辑表达式和一个命名表达式集合.我们使用此种条件化的有向边来表示活动之间的控制关系与数据流动关系.

A 中的每一个元素 a 表示一个活动,它可由一个九元组 $\langle n, t, s, e, R, L, O, P, K \rangle$ 来表示,其中

n 为活动的名称;

t 为活动的类型;

s 为活动的开始条件;

e 为活动的终止条件;

R 为可以参与活动的所有用户的集合;

L 为活动的所有前驱活动(i)及对它所传进来的数据对象所做处理的方法(m)的二元组所组成的集合;

O 为活动所处理的数据对象;

P 为 R 中的用户可以用何种方式去访问 O 中的域;

K 表示所有后继活动(j)、激发这个后继活动条件(c)以及数据对象转换所调用函数(f)的三元组所组成的集合.

条件有向图使用有向有限图来描述工作流过程.节点表示工作流过程的实际或是抽象活动.边表示将激发的下一个活动节点.这个系统提供层次的方法直接在图上定义工作流过程. CDG 是面向数据.也就是说,每一个边只是将数据对象传给下一个

活动然后由数据对象来激发活动. CDG 有一个特点是, 每一个活动节点里根据它不同的输入边而会调用不同的处理函数.

本文的第 2 节给出影响工作流过程性能的几个主要因素和 WPR 应达成的目标, 然后在这个基础上给出优化的方法. 第 3 节将提出的优化方法应用到 CDG 工作流模型建立起来的工作流过程中并给出相应的算法. 第 4 节我们做出总结并提出未来的工作.

2 工作流过程性能定量分析方法

过去人们大都使用定性的分析方法来分析工作流过程的性能, 在这里我们对工作流过程进行定量分析, 以此为基础来判断工作流过程性能的优劣, 然后, 提出改进的方法. 根据 Loos 所提出来的工业工作流的需求^[14], 再结合工作流的特性, 得出下面 5 个影响工作流性能的主要因素:

(1) 复杂度 c . 复杂度用来表示工作流的复杂性. 一个工作流过程简单表示它的复杂度低. 并不是工作流过程中的活动数目(a)少, 就表示它的复杂度低, 而是等价活动类的个数(b)越多, 它的复杂度越高. 我们对工作流过程由上往下分析, 对于可以并行执行的活动我们称之为等价活动, 使之成为同一集合, 所得的集合个数就是 b .

(2) 执行时间 t . 执行时间是指当这个活动可以开始执行的时候一直到它完成后提交给系统的整个时间. 在这里, 我们不考虑用户因为私人的关系, 而使活动执行延长的时间, 只考虑用户真正处理任务的时间. 每一个活动(a)的执行时间可以用 $a(t)$ 来表示. 我们希望得到一个较短的执行时间. 一个工作流过程的总体执行时间的计算方法为 $\sum b_{ia}(t)$, b_{ia} 为各个等价活动类别 b_i 里面所需处理时间最长的活动. 相同的复杂度并不表示有相同的执行时间.

(3) 控制复杂度 u . 表示系统对执行该工作流过程所需提供的控制和资源管理. 它由对象并发操作 o 和数据对象转换次数 e 决定. 对象的并发操作 o 表示如果同时使用同一个数据对象的用户越多, 表示要处理更多的同步操作和数据的一致性, 也就是控制复杂度愈高, 但是, 在处理得当的情况下, 可以增加系统的效率. 数据对象转换的次数 e 表示工作流执行时数据对象的传输转换. 一个工作流过程的数据转换次数越少, 系统所需提供的控制愈低.

(4) 稳定性 s . 一个系统的稳定性就是系统成

功执行的概率, 是由它的失败恢复率 r 和它的失败率 f 来决定: $s = r \times f + (1 - f)$. 为了提高系统的稳定性, 我们要尽量减低它的失败率或是提高它的失败恢复率. 失败率 f 受底层技术支持和控制复杂度 u 的影响, 底层技术支持无法从工作流管理系统中改进, 所以我们假设它是完美的.

(5) 企业策略 p . 这存在很大的人为因素. 相同的任务, 可以使用不同的工作流过程来实现. 是否为最好的工作流过程, 只能由使用的人来决定.

由上得出评价系统性能 v 的数学函数 $v = F_v(s, u, p, (\sum a_i(t)), c(b))$. v 的值和 $s, u, p, (\sum a_i(t))$ 是成正比的, 和 $c(b)$ 是成反比的. $s = r \times f(u) + (1 - f(u))$, 它的失败率 f 受控制复杂度 u 的影响. 这里, 我们假设系统的稳定性是 1, 也就是整个工作流管理系统提供的使用环境是完美的.

根据上述得出的结论, 我们提出 WPR (Workflow Process Re-engineering) 这个概念. WPR 是根据工作流过程的特性和需求, 所提出的一个分析和优化工作流过程的方法, 它是一个由上而下 (top-down) 的分析方法, 也就是由工作流的起始节点开始分析得出活动间的依赖关系. 由上面的数学函数可以得到, 我们只要改变其中的任何一个参数, 对系统的性能都有一定的影响. 我们以改善这些参数为目标, 然后使用现有的工具来支持工作流的协同及达到工作流的优化. 最重要的是, 在优化后用户的企图和执行结果, 在工作流过程里面不会被改变.

WPR 工具包含建立事务任务模型和优化工作流过程的工具. WPR 主要是对“过程”和“网络性能”的优化, 所以我们希望 WPR 在改善上述的需求时, 还能达到下面的目标:

(1) 在工作流管理系统中加强合作、协调和协同的能力.

(2) 突出团体或每个人之间任务的关系.

(3) 保证用户的原来意图和执行结果在经过优化后是不变的.

根据上述的目标和在简介中介绍的 BPR, 我们可以得出 BPR 和 WPR 的不同, BPR 主要检测和分析组织之间的工作流和过程来找到它的瓶颈, 然后提供对企业内部的重组或改变企业的策略方针而提高整个流程的效率. 在 WPR 中的优化是根据整个工作流过程的数据依赖关系对工作流进行重组; 它主要的实现方法是使用工作流模型提供的技术来改善工作流过程的协作和性能, 例如提供协作的机制.

因为一个优化方法不可能只影响一个参数, 所

以下我们对优化方法影响最多的参数进行分类, 并分析它对 workflow 结构的影响.

(1) 复杂度: 可以对 workflow 结构进行重组, 以达到降低复杂度的目的.

(2) 执行时间(t): 可以将 workflow 过程中的冗余元素去除.

(3) 控制复杂度(u): 简化 workflow 过程中的输入输出流的判断条件.

(4) 工作流特性优化: 可以提供协作的机制让多个活动同时运行, 或是向下兼容为 BPR 的优化做准备.

这里我们给出通用性的优化方法, 而不是对于特定某个领域的优化方法, 所以, 具体的应用和动态的优化方法都可以在这个基础上再做进一步的扩展.

3 工作流过程优化方法

在确定影响 workflow 过程性能的几个主要因素和初步了解 workflow 过程实现的步骤之后, 我们具体将第二节所提出的方法应用到 *Workflow!* 工作流模型并给出相应的算法.

3.1 将外部动作改成内部动作

定义 1. 协商节点——如果有两个节点处理相同的数据对象而且对它有相同的操作权限, 他们可以互相传送数据对象进行通信并且所传送的数据对象是由相同的属性构成, 我们称这两个节点为协商节点.

A 节点和 B 节点为协商节点的充分必要条件:

(1) $\exists \langle B, M_b \rangle \in A.L$ and $\langle A, M_a \rangle \in B.L$;

(2) $A.O = B.O$ and $A.P = B.P$;

(3) $\langle B, c_b, f_b \rangle \in A.K$, $\langle A, m_a \rangle \in B.L \Rightarrow A.O = m_a(f_b(A.O))$, 可以得出 m_a 为 f_b 的逆变化;

(4) $\langle A, c_a, f_a \rangle \in B.K$, $\langle B, m_b \rangle \in A.L \Rightarrow B.O = m_b(f_a(B.O))$, 可以得出 m_b 为 f_a 的逆变化.

在找到一组协商节点时, 我们可以提供一个协商的工具让他们直接进行协商. 也就是说, 我们将外部活动改成内部活动.

如果我们找到 N 个协商节点, N_1, \dots, N_n , 他们的前继节点为 I_1, \dots, I_k , 他们的后继节点为 O_1, \dots, O_m , 我们可以将他们合并成一个节点并且为他们提供某种协商技术, 如图 1 所示.

在这个情况下, 我们将多个非独立的节点(它依赖于之前的执行结果)合并为一个节点, 由第 2 节的

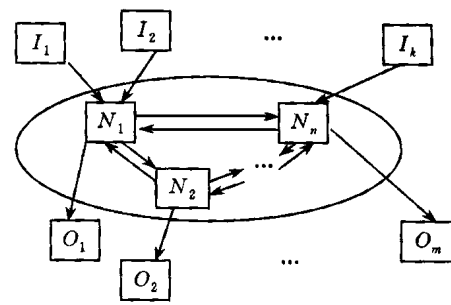


图 1 多个协商节点合并为一个节点示意图
介绍知, 如果优化前的等价活动类的个数为 B_p , 那么优化后的等价活动类的个数就变为 $B_p - (n - 1)$. 由上可知, 如此不仅减少了系统的复杂性 c 还减少了数据的转换次数 e . 因为增加对某个数据的并发操作数目可提高控制复杂度 u , 但是, 我们已经假设系统的稳定性是 1, 所以这里并不影响系统的性能, 且在这样的优化还体现出工作流的协作机制.

为了保证用户的意图不变, 我们在合并之后必须遵守下面两条规定:

(1) 如果它的前继节点原来是属于 N_m , 我们设定 N_m 为协商活动的初始者.

(2) 如果是在 N_k 退出结束整个协商活动, 我们就只触发原来 N_k 所能触发的节点.

第(1)条是为了保证优化前后发起协商的群组是一样的. 第(2)条是为了保证协商之后, 优化前后所触发的活动是一样的. 如果所使用的工作流管理系统不支持一个节点有多个用户共同协作, 我们可以参考 Georg^[15]提出的方法, 它主要是将多媒体会议系统的技术嵌入工作流管理系统.

3.2 突显活动间实际世界的关系

定义 2. 相似节点——如果两个节点有相同的用户集合和处理相同的数据对象, 而且对于相同的数据对象域有相同的权限. 对于相同的前驱节点和后继节点有相同的处理方式, 我们称这两个节点为相似节点.

A 节点和 B 节点为相似节点的充分必要条件:

(1) $A.R = B.R$ and $A.O = B.O$ and $A.P = B.P$ and $A.S = B.S$;

(2) $\forall \langle C, m_{ac} \rangle \in A.L$, $\langle C, m_{bc} \rangle \in B.L \Rightarrow \exists \langle A, c_a, f_a \rangle, \langle B, c_b, f_b \rangle \in C.K$ and $c_a = c_b \Rightarrow m_{ac}(f_a(C.O)) = m_{bc}(f_b(C.O))$;

(3) $\forall \langle C, c_a, f_a \rangle \in A.K$, $\langle C, c_b, f_b \rangle \in B.K$ and $c_a = c_b \Rightarrow \exists \langle A, m_{ca} \rangle, \langle B, m_{cb} \rangle \in C.L \Rightarrow m_{ca}(f_a(A.O)) = m_{cb}(f_b(B.O))$.

在合并相似的节点之后, 所得到的性能优化和

3.1 节中的结果是类似的, 这里不再赘述. 但从商业的角度来看, 我们知道每一个部门或是组织他们所负责的任务通常是固定的. 而根据上面的定义我们知道相似节点属于同一部门或是组织, 所以他们实行的任务在很大程度上是相同的. 为了让系统能更清楚地明白组织或部门之间的关系, 我们将相似节点合并起来. 这样我们可以使用 BPR 的工具来对这个优化后的工作流进行企业策略的优化. 这里我们主要分析“谁”在这个活动内工作, 而不是这个活动做些什么“事”. 这一个步骤可以看成是为了做 BPR 优化而做的准备.

在前面我们已经提到 CDG 定义的工作流里面, 不同的输入边在节点内的执行动作是不同的, 所以这里我们合并相似节点以后, 还是可以区分出原来各自节点所要执行的事务.

应该注意的是, 图 2 中, 在合并活动 N_1 和活动 N_2 以后, 会出现两个重复的边 L_1 和 L_2 , 我们可以将其中的一个删除. 还有, 在做合并优化时, 应该不让各个节点的判断语句过于庞大, 我们做优化的目的是为了简化工作流过程, 而不是使它变得更加复杂化.

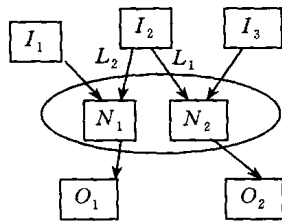


图 2 相似节点合并示意图

3.3 冗余元素的删除

在用户自定义的工作流过程中, 一定存在冗余的活动节点, 而且我们应该从工作流过程中将它移去来减少工作流过程执行的时间. 移去对系统不起任何影响的活动节点, 所达到的效果是显而易见的, 就像是编译优化中移去冗余赋值语句, 都可以减少它的执行时间. 不同的应用中会有不同的定义, 这里我们介绍两种常见的冗余活动节点.

(1) 只有输出边没有输入边的节点. 这种节点如果不是初始节点, 那它就永远不会被激发, 这种节点是冗余的.

A 为冗余节点的充分必要条件:

- ① $A.L = \emptyset$;
- ② $A.t! = \text{'initial'}$.

(2) 重复工作的节点. 如果存在一个节点, 它一定到达一个和它相似的节点, 且在到达这个相似节

点之前, 它所处理的数据对象都不会被改变. 我们知道由上面相似节点的定义, 这两个节点会处理同样的数据对象且有同样的修改权限, 因为它所处理的数据在后来的节点还会再重复一次, 所以这个节点在整个活动中是多余的.

A 为 B 的重复工作节点的充分必要条件:

- ① $Similar(A, B) = true$ and $GoThrough(A, B) = true$;
- ② $Value(A.O) = Value(B.O)$;
- ③ $A.R = B.R$.

函数 $Similar(A, B)$ 判断 A, B 是否为相似节点.

函数 $GoThrough(A, B)$ 判断 A 是否一定可以到达 B.

函数 $Value(A.O)$ 表示活动 A 开始时, 数据对象 O 中每一个属性的值.

以上三个函数都可以很容易地实现出来, 所以这里不再详细写出.

以上介绍的两种活动节点都可以直接从工作流中移去, 而不会改变它的执行结果. 要判断一个节点对于这个工作流过程是否是多余的, 除了根据它是否会被触发 (第一类节点), 还可以根据它所处理的任务是否有存在意义 (第二类节点) 来判断它是否可以移去. 在移去这些多余元素的时候, 应该注意会不会使得下面所需要执行的数据变得不一样.

3.4 工作流结构的重组

定义 3. 独立节点——指这个节点的触发条件是不依赖于它的父节点 N_p 的执行结果, 而且它所处理的数据对象并不会被它的父节点所改变. 这样的节点我们称之为独立于父节点 N_p 的节点.

A 节点独立于 B 节点的充分必要条件:

- ① $\exists \langle B, m \rangle \in A.L$ and $\neg \exists \langle A, m \rangle \in B.L$;
- ② $\langle B, m_b \rangle \in A.L, \langle A, c_a, f_a \rangle \in B.K \Rightarrow A.S(A.O) = A.S(m_b(f_a(B.O)))$;
- ③ $Value(A.O) = Value(m_b(f_a(B.O)))$.

如果一个节点独立于它的父节点, 我们可以让这个节点和它的父节点同时执行. 也就是说, 我们改变它在工作流中的执行位置. 这里将串行活动改成并行活动, 使得可并行执行的活动变多, 减少等价活动的类别数目 (b), 从而降低系统的复杂度 (c) 并且节省工作流的整个处理时间 (t), 从而提高它的性能. 但是我们必须加入一些特殊的规定使得在异常结束的情形下, 它的结果数据和未优化前的数据是一样的.

下面我们给出一个例子, 来说明它的改变方式.

图 3 是表示用户定义的工作流过程, 这里等价活动的类别数目为 $4(\{N_1, N_2\}, \{N_3, N_4\}, \{N_5\}, \{N_6\})$. 如果节点 N_5 独立于节点 N_3 , 但是依赖于节点 N_4 的执行结果, 我们可以将工作流过程重组, 如图 4 所示. 节点 N_3 和节点 N_5 是并行执行的, 但是节点 N_5 还是在节点 N_4 的后面执行, 这里等价活动的类别个数还是 $4(\{N_1, N_2\}, \{N_3, N_5\}, \{N_4\}, \{N_6\})$. 所以只有当活动是由 N_1 触发的时候, 它的执行时间才有改善, 但整个系统的执行时间我们还是当作不变的. 如果图 3 中的节点 N_5 独立于节点 N_3 和节点 N_4 , 我们可以将工作流过程重组成图 5 的结构. 这样节点 N_3 , 节点 N_4 和节点 N_5 都可以并行执行, 它的等价活动的类别个数为 $3(\{N_1, N_2\}, \{N_3, N_4, N_5\}, \{N_6\})$. 这样整个系统的执行时间就得到了改善.

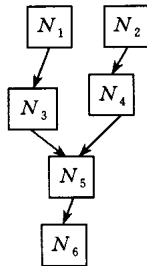


图 3 用户定义工作流

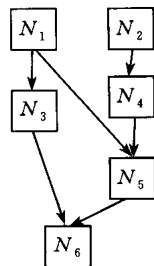


图 4 优化后(1)

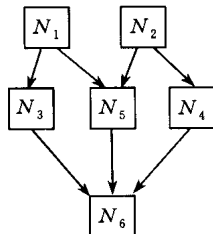


图 5 优化后(2)

在节点 N_6 , 我们必须保证节点 N_5 和节点 N_3 或是节点 N_4 完全执行完成后, 才执行节点 N_6 . 在这里必须改变一些工作流相关的执行规则. 更值得提出来注意的是, 在图 4 中, 如果节点 N_3 发出了一

个异常处理, 我们要判断节点 N_5 是否已经执行完成, 如果已经执行完成, 我们将它复原回去, 然后再做异常处理的动作. 这样做是为了保证优化后工作流的处理结果和原来的是相同的.

3.5 工作流定义的简化

为了方便管理和简化工作流模型, 我们可以加入一些控制节点以达到这个目的. 在加入一个新的节点时, 我们应该注意不要去增加系统的负担和延长处理的时间. 在一些商业用的工作流里面, 不同的节点里面常常会出现相同的判断语句去决定下一个激发的节点. 如图 6 所示.

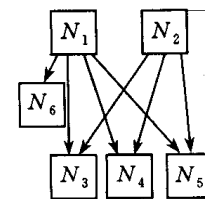


图 6 用户定义工作流

A 节点之前可加入控制节点的充分必要条件:

$$(1) \langle B, m_b \rangle, \langle C, m_c \rangle \in A.L, \langle A, c_b, f_b \rangle \in B.K, \langle A, c_c, f_c \rangle \in C.K \Rightarrow Value(m_b(f_b(B.O))) = Value(m_c(f_c(C.O)))$$

图 6 中, 节点 N_1 和节点 N_2 有相同的子节点 (节点 N_3 , 节点 N_4 和节点 N_5), 而节点 N_6 只是被节点 N_1 触发. 对于节点 N_3 , 我们加入一个新的控制节点 C_1 , 同理, 对于节点 N_4, N_5 各自加入控制节点 C_2, C_3 , 如图 7 所示. 根据上面的定义, C_1, C_2 和 C_3 为相似节点并且可以加以合并变为一个新的控制节点 C , 并将多余的边去掉, 如图 8 所示. 这个节点 C 起着控制的作用, 只是由系统感知和执行. 所以

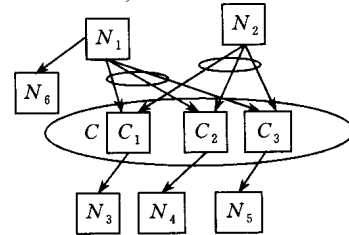


图 7 简化中

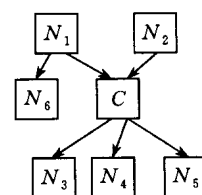


图 8 简化后

加入这样的节点不会增加整个系统的负载和延长流程的执行时间。

当我们要改变它的分枝条件的时候, 我们只需要在节点 C 的地方进行修改。这不仅简化 workflow, 而且使整个 workflow 的关系更为清楚。

3.6 一致性的判断

为了保证优化后 workflow 的工作结果和之前是一样的, 我们利用 workflow 管理系统自动模拟执行优化前和优化后的 workflow 过程, 然后, 比较两者每一步骤的执行结果和触发下一个活动的初始条件是否相同。我们并不是严格要求每一个步骤的结果都是一样的, 例如我们之前提到的: 将独立的工作流节点并行处理的情况下, 独立节点执行完成后, 结果数据可能是不一样的, 因为它的父节点可能还没有执行完成, 这里我们要严格要求的是: 他们触发下一个活动的起始条件是一样的。在异常发生的情况下, 我们就必须要求他们的执行结果必须是一样的。在满足以上的规定后, 我们就认为这样的优化是可以接受的, 并且不改变用户的原始意图。

4 总 结

以前我们都是使用定性的方法来分析 workflow 过程, 这样无法明确地比较 workflow 过程的优劣, 所以我们提出影响 workflow 过程的主要因素并使用这些因素对其进行定量分析。我们通过分析 workflow 过程定义各活动间的数据依赖关系, 得到上面对于流程一般性的优化方法。使用这些方法, 一方面可以减少 workflow 执行的时间, 提高整个 workflow 的效率, 另一方面, 通过加入实时同步协作工具使业务处理者之间的交互更加灵活。

本文从分析活动间的数据依赖关系即一般性的角度入手, 给出了具有通用性的 workflow 优化方法。具体的应用和动态的优化方法都可以在这个基础上再做进一步的扩展。而一个好的优化系统应该能够结合业务流程的具体情况, 并根据这些具体情况进行一些特定的优化。已经有人介绍优化 workflow 过程的技术, 还有一些人已经提出在某个特定领域内具体的优化算法^[16]。

我们相信除了这里所介绍的通用方法外, 还会有其它的方法, 未来可以对此做更进一步的研究。

参 考 文 献

- 1 Thomas M M. Re-engineering rationale. *Journal of Management in Engineering*, 1995, 11(4): 11- 13
- 2 Seidmann A, Sundararajan A. The effects of task and information asymmetry on business process redesign. *International Journal of Production Economics*, 1997, 50(2-3): 117- 128
- 3 Hammer M. Re-engineering work: Don't automate, obliterate. *Harvard Business Review*, 1990, 68(4): 104- 112
- 4 Bardley P, Browne J, Jackson S, Jagdev H. Business Process Re-engineering (BPR)—A study of the software tools currently available. *Computers in Industry*, 1995, 25(3): 309- 330
- 5 Shi MeiLin, Yang Guang-Xin, Xiang Yong, Wu Shang-Guang. WfMS: Workflow management systems. *Chinese Journal of Computers*, 1999, 22(3): 325- 334 (in Chinese)
(史美林, 杨光信, 向 勇, 伍尚广. WfMS: 工作流管理系统. *计算机学报*, 1999, 22(3): 325- 334)
- 6 Hutchison A. CSCW as opportunity for business process re-engineering. *IFIP Transactions A-Computer Science and Technology*, 1994, A-54: 309- 318
- 7 Swenson K D, Irwin K. Workflow technology: Tradeoffs for business process re-engineering. In: *Proc Conference on Organizational Computing Systems*, Milpitas, 1995. 22- 29
- 8 Du Weimin, Davis J, Shan Ming-Chien. Flexible specification of workflow compensation scopes. In: *Proc ACM SIGGROUP Conference on Supporting Group Work*, Phoenix, 1997. 309- 316
- 9 Ellis C A, Nutt G J. Modeling and enactment of workflow systems. In: Ajmone Marsan ed. *App and Theory of PetriNets*. Berlin: Springer-Verlag, 1993. 1- 16
- 10 Peter C L, Walter H D. Object-oriented protocol hierarchies for distributed workflow systems. *Theory and Practice of Object Systems*, 1995, 1(4): 281- 300
- 11 Natalie S G, Daniele S P, Remo P. Generalized process structure grammars (GPSG) for flexible representations of work. In: *Proc ACM Conference on Computer Supported Cooperative Work*, Boston, 1996. 180- 189
- 12 Charles K A, Scott C B, Stephen J M. A Web-based enterprise workflow system. In: *Proc ACM SIGGROUP Conference on Supporting Group Work*, Phoenix, 1997. 214- 220
- 13 Shi MeiLin, Yang Guang-Xin, Xiang Yong, Wu Shang-Guang. Wwww!: A Web-based workflow management system. *Journal of Software*, 1999, 10(11): 1148- 1155 (in Chinese)
(史美林, 杨光信, 向 勇, 伍尚广. 一个基于 Web 的工作流管理系统—Wwww!. *软件学报*, 1999, 10(11): 1148- 1155)
- 14 Loos P, Allweyer T. Application of production planning and scheduling in the process industries. *Computers in Industry*, 1998, 36(3): 199- 208
- 15 Georg S, Michael W, Gerhard P *et al*. Integrating synchronous multimedia collaboration into workflow management. In: *Proc ACM SIGGROUP Conference on Supporting Group Work*, Phoenix, 1997. 281- 290
- 16 Zenic A, Schal T. Analyzing and redesigning a remote sensing business process for rapid estimates of agriculture in Europe. In: *Proc Conference on Organizational Computing Systems*, Milpitas, 1995. 116- 126