软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(7):1702−1716 [doi: 10.3724/SP.J.1001.2012.04222]

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

基于遗传算法的工作流个人工作列表资源调度*

邓铁清[1+], 任艮全[1,2], 刘英博[3]

[1](总后勤部 后勤科学研究所,北京 100071)

[2](清华大学 计算机科学与技术系,北京 100084)

[3](清华大学 软件学院,北京 100084)

Genetic Algorithm Based Approach to Personal Worklist Resource Scheduling

DENG Tie-Qing[1+], REN Gen-Quan[1,2], LIU Ying-Bo[3]

[1](Logistical Scientific Research Institute, Beijing 100071, China)

[2](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[3](School of Software, Tsinghua University, Beijing 100084, China)

+ Corresponding author: E-mail: dengtieqing@sohu.com

Deng TQ, Ren GQ, Liu YB. Genetic algorithm based approach to personal worklist resource scheduling. *Journal of Software*, 2012,23(7):1702−1716 (in Chinese). http://www.jos.org.cn/1000-9825/4222.htm

Abstract: In workflow management systems (WFMSs), appropriate consideration of applying scheduling techniques to manage actors' personal worklists is essential for successful implementation of workflow technology. Mainly, the attention of existing workflow scheduling has focused on the process perspective. As a result, issues associated with personal worklist's perspective, i.e., worklists that contain actors' to-do activity instances, have been largely neglected. Given this motivation, this paper for the first time, investigates issues in personal worklist scheduling under dynamic workflow environment. Towards these issues, the paper proposes a novel genetic algorithm to optimize the personal worklist management. This algorithm recommends for each actor a feasible worklist that will ensure the worklist's activity instances' successful execution while minimizing the total overtime costs for all personal worklists. Through comparing with other well-known workflow scheduling algorithms, the paper evaluates the effectiveness of the proposed genetic algorithm with a specific example and a simulation experiment.

Key words: workflow management system; resource scheduling; genetic algorithm; personal worklist scheduling

摘 要: 在工作流管理系统中,个人工作列表的优化调度具有重要意义.已有的相关研究主要关注工作流实例的调度,而关于个人工作列表调度的研究还较少.首先描述了工作流实例动态执行环境下个人工作列表调度问题,并提出了一个基于遗传算法的个人工作列表资源调度算法.该算法要为每一个执行人推荐一个可行工作列表,并在保证工作项联合执行成功率的同时最小化总体延误代价.最后,通过一个仿真实验将该遗传算法与其他 7 种基于分配规则的典型调度算法进行了比较.结果表明,所提出的基于遗传算法的个人工作列表资源调度算法比已有的其他典型调度算法具有更好的调度效果.

* 基金项目: 国家高技术研究发展计划(863)(2012AA040911)
收稿时间: 2012-02-28; 定稿时间: 2012-03-19

关键词: 工作流管理系统;资源调度;遗传算法;个人工作列表
中图法分类号: TP317 文献标识码: A

工作流实例运行时,工作流管理系统首先按照工作流模板中的活动顺序依次生成或激活相应的活动实例,然后为各个活动实例创建相应的工作项,最后将各个工作项分发给执行人(人力资源)执行[1].个人工作列表是分配给某个执行人执行的工作项列表,工作项分发时,需要在正确的时间将正确的工作项分发给正确的执行人,这一问题被称为个人工作列表资源调度问题[2].在实际的工作流管理系统中,一个工作流活动对应的工作项可能要分配给多个执行人(一对多关系),一个执行人可能要执行多个工作流活动对应的工作项(多对一关系),而且一个工作流模板中的某个活动可能同时对应多个激活的活动实例,使得在进行工作项分发时需要将多个工作项按一定规则分别分配给多个执行人.这些工作项对应的工作流活动可能相同,也可能不同.随着网格计算、云计算等各种信息技术的快速发展和广泛应用,协同工作流、网格工作流、云工作流等工作流技术也在不断发展,资源共享包括人力资源共享的程度不断提高,在工作项分发时,将多个工作项分配给多个执行人的现象越来越普遍,个人工作列表资源调度问题的重要性也在不断上升.

个人工作列表资源调度要解决的是多个执行人执行多个工作项时的资源调度问题,其面临的主要难点是工作流运行时的不确定性,例如执行轨迹的不确定性、执行时间的不确定性、优化目标的不确定性等等.在工作流运行时,不同的时间会有不同数量的流程按照不同的轨迹运行,因此,工作项的执行轨迹和创建时间等参数具有不确定性;其次,参与工作流执行的人员是高度自主的,每个人完成工作的方式和花费的成本都不相同,因此,工作项的执行时间和执行成功率等参数具有不确定性;再次,由于工作流技术应用涉及到各种不同的业务,而不同时期执行的工作流需要达到的目标也各不相同,因此,其具有目标多样性.这种不确定性和多目标性使得在将其他调度算法直接应用到个人工作列表资源调度问题时会遇到较大困难.事实上,在很多基于工作流的应用中,人们更多地是依赖自身的经验,而不是依靠系统来自动进行人员分配[2].

本文使用遗传算法来应对个人工作列表资源调度的不确定性.遗传算法(genetic algorithm)是目前公认的解决不确定性调度问题效果较好的启发式算法,其基本思想是仿效生物界中“物竞天择、适者生存”的演化法则,把问题参数编码为染色体,再利用选择、交叉和变异等运算来交换种群中染色体的信息,最终生成符合优化目标的染色体.遗传算法已经较好地应用于网络工作流调度、数据中心资源调度等与个人工作列表资源调度具有相似不确定性因素的领域.本文遵循“先猜测后解决”的思路,首先给出个人工作列表资源调度的基本问题模式,并根据已有研究成果假设待分配工作项的执行时间符合正态分布;然后提出基于遗传算法的个人工作列表资源调度算法,包括初始群体生成算法、适应度函数和终止条件以及 3 种遗传算子(选择、交叉和变异)的选择和实现方法等.该算法要为每个执行人推荐一个可行工作列表,并在保证联合执行成功率的同时最小化总体延误代价.已有的个人工作列表资源调度研究主要使用了基于分配规则的调度算法[2,3],为了验证遗传算法的优越性,最后通过实验来对比分析遗传算法和其他 7 种基于分配规则的调度算法在个人工作列表资源调度问题中的应用效果.

1 相关研究工作

工作流人力资源调度可分为工作流实例资源调度和个人工作列表资源调度两类[4,5].本节从个人工作列表资源调度、工作流实例资源调度和其他相关资源调度研究这 3 个方面分别介绍相关的研究工作.

1.1 个人工作列表资源调度

一些工作使用了简单的基于分配规则的调度算法[2,3]来解决个人工作列表资源调度问题.文献[6,7]针对公文生成过程中结构化工作流管理,提出了一个自适应工作流模型.在此基础上,文献[8]采用预定义规则和动态调度执行相结合的思路,提出了一种自适应的个人工作列表调度算法,使用了随机选择的分配规则(SIRO).Eder 等人在个人工作列表管理问题上的初步研究与本文提出的问题最为相似[2,3,9],他们提出了一种基于确定性模型

的调度方法.首先,使用执行时间的直方图来建模工作流实例,并且为每一个工作项生成一个执行时间;然后,基于这个执行时间,基于分配规则的调度算法(LDF 和 LSTF)来对一个个人工作列表进行排序,其主要优化目标是减少个人工作列表中延误工作的数量以及延迟度.但其调度算法建立在一个确定性的模型上,即每个工作项的执行时间均是事先确定的,这样的模型无法解决工作项执行时间的不确定性问题.而且该调度算法只针对某一个个人工作列表进行调度,取得局部的最优解,无法解决全局个人工作列表调度的问题.

部分研究从工作项执行人的角度探讨工作流调度的问题.文献[2]设计的算法为流程参与者提供有效的信息,通知其将要执行的活动以及活动的可能完成时间,使流程参与者能够在工作到来之前提前进行安排.为了解决时间的不确定性问题,文献[10]提出了一种基于决策树的活动执行时间预测技术.该方法基于工作流日志中前序工作项的执行人、执行时间、等待时间等特征,预测当前活动的执行时间.文献[10]还提出了一种基于时序逻辑的关联工作项挖掘方法,该方法通过分析工作流日志中工作项执行开始时刻与结束时刻,发现与其可并行执行的关联工作项,并将其推荐给用户.文献[11]基于分治思想,假设每一活动的业务实例逗留时间服从负指数分布,给出了以资源的单位时间消耗成本最小化为目标的求解最优资源数量的调度方法,得到了一种基于工作流模型的资源优化配置方法.文献[12]提出活动实例在多个资源中的路径和一个产品在生产部门的不同机器中的路径具有很多的相似性,他们提到了一些简单的分配规则用于指导每个资源选择将要执行的工作,但是没有继续深入研究这个问题.文献[13]采用智能代理的方法,通过指定任务提前通知规则,智能代理之间能够互相通知,并将任务提前通知给执行人.

1.2 工作流实例资源调度

工作流实例资源调度可以分为两类:单流程实例资源调度和多流程实例资源调度.单流程实例资源调度主要对属于单个流程实例的多个活动实例进行资源分配.在工作流研究中,时间和调度是两个紧密交织在一起而又十分复杂的问题.在运行时,人们需要对工作流实例进行检查,以便动态地发现那些可能违反时间约束的工作流实例.如果存在这样的流程实例,就需要进行事前调度以防止时间约束被违反;而一旦时间约束被违反,还需要对流程实例进行事后调度以减少不必要的损失[14].有一些工作流调度算法试图为工作流和资源模型设置截至时间、资源成本等各种约束条件,然后通过自动化的手段判断约束是否满足,或者根据约束找到合适的资源配置,从而让工作流管理系统对其所管理的工作流实例进行合理的安排,以提高整体执行效率或者避免潜在的错误.文献[14]只考虑了单个活动实例的资源分配问题,目标是使某个活动实例完成得最早.文献[15]首先根据流程实例总体截止时间、各个活动实例的负载及其相互之间的依赖关系估算出各个活动实例的截止时间,然后将整个流程实例调度问题拆分为多任务调度问题.这两篇文章都致力于在满足流程实例总体最晚完成时间的要求下,最小化整个工作流实例的执行消耗.在此基础上,文献[16]提出的一种遗传算法试图在最小化单流程实例执行成本的同时,尽量缩短流程实例执行时间.文献[17]研究了工作流活动多实例的调度控制,对活动多实例的属性进行了统一的形式描述,提出了活动多实例控制体 Shell,用于控制活动多实例的分配和提交.

针对多流程实例资源调度的研究相对较少.文献[18]采用先到先服务(FIFO)、最近截止优先(EDF)、短任务优先(SJF)的策略对流程实例进行调度以满足时间约束.文献[19]尝试在工作流管理系统中减少延误工作流实例的数量,使用了“先猜测再解决”的解决思路.首先猜测实例的执行时间、执行路径等不确定性因素,然后使用遗传算法来解决生成的确定性调度问题.文献[20]使用多种基于分配规则的调度算法(FIFO,SIRO,EDD 和 SPT)进行多个工作流实例间的调度.文献[21]使用两步搜索形式的基因算法来调度多个工作流实例,以减少它们的总体完成时间.文献[22]提出了一种基于机器学习的半自动工作流人员分配方法,该方法针对工作流中某些关键活动的执行人需要在运行时进行动态指派的问题,通过学习相关工作流日志中该活动的人员分配模式,为流程实例负责人推荐这些关键活动的候选执行人.已有的相关研究主要关注了工作流实例资源调度问题,提出了一些工作流实例资源调度算法.与工作流实例资源调度相比,个人工作列表资源调度问题中待分配的工作项可能来自多个工作流模板,各个工作项的创建时间、执行时间等关键特性一般是不确定的,资源调度的优化目标也有较大区别,文献[19]详细讨论了这些问题.这些特性使得个人工作列表资源调度面临更多的不确定性,大部分工作流实例调度算法不能被直接应用到个人工作列表资源调度中.

1.3 其他相关资源调度研究

资源分配实际上是一个早已存在的问题:在操作系统领域中,CPU 资源在不同进程之间的分配就是一个资源分配问题.其研究的核心内容是:如何找到工作集合和人员集合之间的最佳匹配.传统人员分配方法基于人员与工作之间的匹配知识(如成本/效益矩阵等)来讨论分配策略与算法,从运筹优化的角度来研究这一问题[23].但是,个人工作列表资源调度的动态性、不确定性和多目标性,使得人们很难全面把握工作和人员之间的匹配知识,如果将传统调度算法直接应用到个人工作列表资源调度,将会遇到很大困难.与个人工作列表资源调度具有相似不确定性因素的调度是网格计算、云计算等网络计算环境下的资源调度,比较典型的有网格工作流调度和网络数据中心资源调度.

在面向网格的工作流中,资源是能够执行活动的计算实体[24].网格计算环境下有大量的计算、存储等资源,不仅处理的逻辑流程和流程关系更加复杂、涉及多个步骤、资源和过程等多种不确定因素,而且应用规模比一般应用大得多,往往需要实现各类网格资源的共享和集成,并在应用网格的支持下面向特定应用需求提供协同工作支持环境[23,25].文献[26]指出,工作流任务的自动分配需要考虑的因素比较多,包括待分配任务和候选资源的属性与状态,重要任务应该分配给高可靠性候选资源,低重要性任务可分配给低可靠性的候选资源,以免低可靠性资源一直处于饥饿状态.文献[25]重新定义了关键区间和相关任务集合等概念,通过相对门限值确定相关任务集合,有效地将执行时间较短的非关键路径任务排除在相关任务集合之外,同时提出了一种新的网格工作流资源分配算法,仿真结果表明了算法的正确性.文献[27]基于响应时间、花费、可靠性等多个服务质量参数,利用遗传算法来搜索资源空间(染色体编码),从而得到一组资源,并分配给工作流中的各项任务.

在面向 Web 应用的网络数据中心资源调度问题中,资源是设备所消耗的能源.服务器(以下简称为节点)资源利用率低,是数据中心能耗巨大的重要原因之一.目前,全球数据中心节点资源的平均利用率远低于期望值.如何提高资源利用率,成为数据中心控制能耗必须解决的问题[28].在网络数据中心资源调度问题中,需要根据不断变化的需求和环境条件,如请求的类型与数量、节点负载情况、资源利用率等不确定因素,对集群中的虚拟机进行动态配置(动态地调整节点运行数量及其上部署的虚拟机类型),在满足用户服务质量的前提下,提高集群的资源利用率.文献[28]提出了一种动态资源按需配置方法,基于遗传算法并行化搜索配置空间,能够根据不断变化的资源需求,以在线方式高效地重配置集群.文献[29]提出了一种针对大规模数据中心的应用动态放置方法,采用多目标优化算法,并通过 3 种算法(剩余配置算法、增量配置算法和重平衡配置算法)搜索近似最优配置.

2 问题概述

2.1 个人工作列表资源调度的基本概念

令 a_i 表示一个工作项,下面定义工作项的一些与时间相关的重要属性.

(1) 执行时间 d^{a_i} :实际工作流应用中存在的大量人为和环境等不确定性因素,给工作项调度增加了很大难度.文献[15]指出,如果能够对活动的执行时间进行预测,并且预测的误差能够控制在 30%以内,则应用调度算法可以有效地减少工作流执行过程产生的延迟任务.文献[10]针对活动执行时间的不确定性,提出了一种基于决策树的活动执行时间预测技术,从预测结果来看,一般工作项的执行时间比较接近正态分布.

本文假设工作项 a_i 的执行时间符合正态分布,即 $d^{a_i} \sim N(\mu_{a_i},\sigma_{a_i}^2)$,其中, μ_{a_i} 是正态分布的期望值, $\sigma_{a_i}^2$ 是其标准差.由正态分布的 3σ定律可知,正态分布样本有 99.73%的概率会在区间$[\mu-3\sigma,\mu+3\sigma]$中,故定义工作项 a_i 的最小执行时间为 $d_{\min}^{a_i}=\mu_{a_i}-3\sigma_{a_i}$,最大执行时间为 $d_{\max}^{a_i}=\mu_{a_i}+3\sigma_{a_i}$.

(2) 开始时间 s^{a_i} :工作项 a_i 的开始时间,即工作项对应的活动实例在系统中被激活的时间.工作项开始时间取决于其创建时间前一个工作项的完成时间,也是不确定的.

(3) 截止时间 e^{a_i} :工作项 a_i 的截止时间,在工作流模型中一般由用户按照一定的规则手工定义.

(4) 执行成功率 r^{a_i} :工作项 a_i 可以在截止日期之前完成的概率.对于工作项 a_i,只有当其完成时间($s^{a_i}+d^{a_i}$)

早于截止时间 e^{a_i} 时才会被认为是成功执行的.如图 1 所示,工作项的完成时间落在区间 $i_1=(s^{a_i}+d^{a_i}_{\min},s^{a_i}+d^{a_i}_{\max})$ 上,只有当工作完成时间落在区间 $i_2=(s^{a_i}+e^{a_i})$ 上时才能成功执行.故工作项 a_i 的执行成功率应该等于随机变量($s^{a_i}+d^{a_i}$)处于时间区间 $i_3=(s^{a_i}+d^{a_i}_{\min},e^{a_i})$ 上的概率总和.

由此可见,当 $e^{a_i}<s^{a_i}+d^{a_i}_{\min}$ 时, r^{a_i} =0;当 $s^{a_i}+d^{a_i}_{\min}<e^{a_i}<s^{a_i}+d^{a_i}_{\max}$ 时,0< r^{a_i} <1;当 $e^{a_i}>s^{a_i}+d^{a_i}_{\max}$ 时, r^{a_i} =1.

(5) 延误代价 c^{a_i} :由于工作项 a_i 的完成时间超过截止时间所带来的惩罚代价.工作项 a_i 的延误会导致其他工作项的延误,或者需要调用更多的资源来完成,从而造成一定的损失.延误代价的大小可能由多个参数决定,例如工作项 a_i 的重要度或紧急度.

在上述属性中,执行时间、开始时间和截止时间使用相同的时间单位,如小时或分钟.执行成功率的单位为百分比,不同工作项的延误代价的计量单位也有可能不同,本文假设其是一致的.

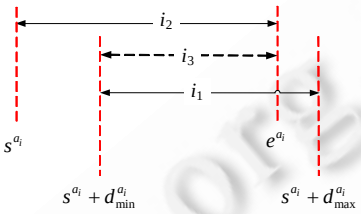


Fig.1 An activity instance's successful execution ratio

图 1 工作项执行成功率示意图

2.2 典型案例和问题分析

图 2 展示了一个在实际工作流管理系统中出现的个人工作列表管理的例子.

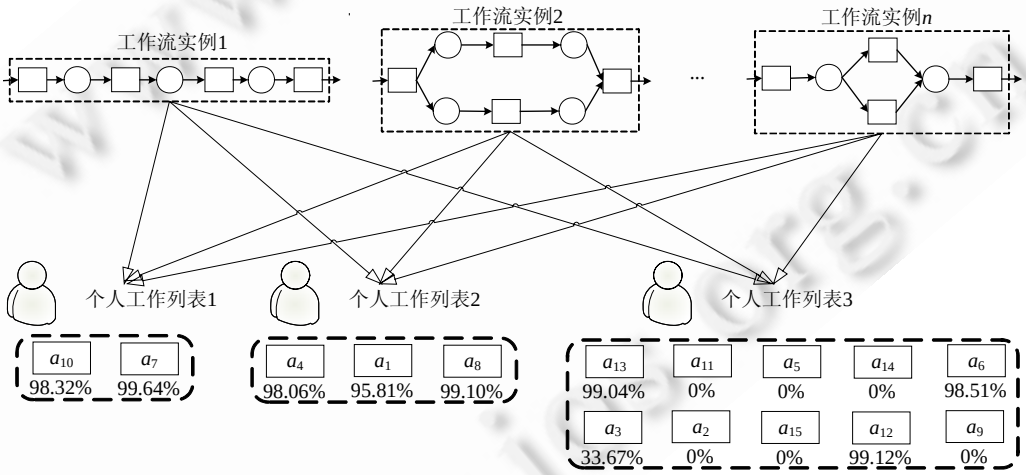


Fig.2 Example scene of personal worklist management

图 2 个人工作列表资源调度示例

在这个例子中,一共存在 3 个执行人,都是某个装备管理财务部门的会计,他们可以执行 3 种不同的工作项:查价、查账和开发票.示例中,多个工作流实例共产生了 15 个活动实例,从而生成了 15 个工作项,并被分配到 3 个执行人的个人工作列表中,在分配过程中未应用调度算法.图中工作项下面的数字代表它的执行成功率.由此可以看出,当未采用调度算法时,个人工作列表资源调度存在着如下问题:

(1) 从工作项的角度来看,每个工作项的执行时间和执行路径对于工作项的执行成功率有着重要影响;

(2) 从单个个人工作列表的角度来看,个人工作列表中的工作项数量和工作项的相关属性对工作列表的完成情况有较大影响.以个人工作列表 3 为例,a_3 加入该工作列表时,其执行成功率从 99.72%降到了 33.67%,主要是由于之前的工作项严重推迟了工作项 a_3 的开始时间,从而使其执行成功率骤降.同时,个人工作列表 3 中由于工作项负荷非常重,使其中的大多数工作项由于执行时间冲突而无法完成;

(3) 从多个个人工作列表的角度来看,工作负荷的平均分配是比较重要的一个需求.示例中,个人工作列表 1 和列表 2 总共只包含 5 个工作项,而个人工作列表 3 包含 10 个工作项,这样不平衡的分配,使得资源 3 有较差的稳定性:工作负荷过重,使其负责的多个工作项难以在截止时间前完成.

2.3 个人工作列表资源调度问题描述

上述示例表明,在个人工作列表管理时,有必要使用一定的调度技术,以更好地安排待执行工作项,从而使资源调度更为合理.在此,首先给出个人工作列表资源调度的问题定义:

令$\{a_1,a_2,\ldots,a_n\}(n>1)$表示一定数量的待执行工作项,$N$ 表示可以执行这些工作项的资源(执行者)的个数,我们期望一个调度算法能够给出一个调度安排,其中有 N 个可行工作列表(用工作列表 S 表示)和对所有资源的一个不可行工作列表(用工作列表 U 表示).相对于每一个资源存在一个可行工作列表,其中包括的工作项是可以在各自的截止时间内完成的;对于所有资源的不可行工作列表,包含着那些不可能在截止时间内完成、将会延误的工作项.

需要注意的是,在可行工作列表中,所有工作项的执行成功率是不确定的.由于工作项被顺序安排,每一个在可行工作列表中的工作项的开始时间会被更新,从而使其执行成功率也会被更新.例如,对于可行工作列表 S 中的工作项 $a_j(i\leqslant j\leqslant n)$,按照开始时间对 S 中的工作项进行排序后,a_j 之前的工作项为 a_i,其执行时间期望为 μ_{a_i},截止时间为 e^{a_i},所以工作项 a_i 将会在其截止时间或之前完成,完成时间为 $\min\{s^{a_i}+\mu_{a_i},e^{a_i}\}$.相应地,工作项 a_j 的开始时间则由其本身的开始时间或前一个工作项 a_i 的完成时间决定,即 $s^{a_j}=\max\{s^{a_j},\min\{s^{a_i}+\mu_{a_i},e^{a_i}\}\}$.于是,$a_j$ 的执行成功率应该根据它更新的开始时间重新计算.由于开始时间、截止时间、执行成功率等参数对执行时间存在着依赖性,因此,这些参数与执行时间一样具有一定程度的不确定性.

下面定义两个用于评价个人工作列表调度算法优劣的参数:

定义 1(可行工作列表中所有工作项的联合执行成功率 r^s). 给定调度方案,令$\{a_1,a_2,\ldots,a_n\}(n>1)$表示其中所有可行工作列表中的 n 个工作项,则 r^s 是所有 n 个工作项的执行成功率 r^{a_j} 的乘积:

$$r^s=\prod_{i=1}^{n}r^{a_j}.$$

定义 2(不可行工作列表中所有工作项的总体延误代价 c^U). 给定调度方案,其中不可行工作列表中总共有 m 个工作项:$\{a_1,a_2,\ldots,a_m\}(m>1)$,$c^U$ 是所有 m 个工作项延误代价的总和:

$$c^U=\sum_{i=1}^{m}(c^{a_i}).$$

对多种调度算法而言,在满足可行工作列表中所有工作项的联合执行成功率 $r^s\geqslant\delta$(δ是系统内用户设定的安全阈值)的前提下,不可行工作列表中所有工作项的总体延误代价 c^U 最小的调度方案所对应的调度算法性能最好.安全阈值δ代表着用户对于个人工作列表的成功率要求,需要同时考虑到系统的需求(性能、吞吐量、服务质量等)和资源的情况(效率、费用、工作负荷以及项目经验等).这个要求事实上是系统管理员在系统稳定性和延误代价之间的平衡.显然,较高的联合执行成功率会带来更高的工作项延误代价.在给定一个安全阈值的条件下,好的调度算法要尽可能地减小总体延误代价.

因此,个人工作列表资源调度问题可以描述如下:给定工作项集合 $A=\{a_1,a_2,\ldots,a_n\}(n>1)$,资源数目 $m(m>0)$和需要的工作项联合执行成功率 r,调度算法需要给出各个资源的可行工作列表$\{W_1,W_2,\ldots,W_m\}$和整体的不可行工作列表 U,使得调度方案在满足所有可行工作列表中工作项的联合执行成功率(r^s,见定义 1)阈值的情况下,具有最小的总体延误代价(c^U,见定义 2).

3 基于遗传算法的资源调度

3.1 遗传算法概述

遗传算法由 Holland 提出[30],是一种鲁棒的空间搜索技术,可以在线性时间内使用进化的原理从一个较大的搜索空间中获得一个可行的解[31].遗传算法中,搜索空间中的一个解被表示成一个个体(被看作染色体).这一算法需要维护一定数量的个体,这些个体的集合被称为群体.每一代群体之间不断进化,将过去搜索出的最好假设和在新区域中搜索出的最好假设加以结合形成新一代群体[32].群体中一个个体的质量由适应度函数所决定,适应度值描述了一个个体相对于群体中其他个体的质量.

典型遗传算法的基本步骤如图 3 所示,应用其来解决个人工作列表资源调度问题.首先,需要对个人工作列表资源调度问题进行编码,随机生成初始的第一代群体.然后,遗传算法将使用 3 种不同的遗传操作算子生成新的一代群体:选择、交叉和变异.在生成新群体之后,使用适应度函数来评价群体中每个个体,并且选取新群体中适应度最高的个体.如果这个个体假设满足终止条件,则遗传算法终止,生成的这个最好个体为问题的解决方案;否则,重复上述步骤,直至符合终止条件.

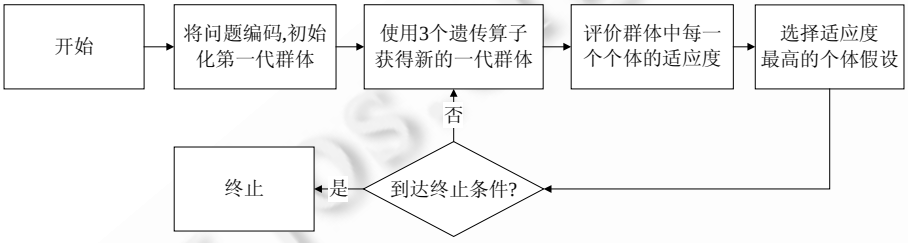


Fig.3 Overall steps of genetic algorithm

图 3 遗传算法的基本步骤

3.2 问题编码和初始群体生成

在遗传算法运算之前,需要针对问题设计染色体,包括基因字串的长度以及基因代表的含义.即,对要搜索空间的可行解以编码的形式来呈现.一般的编码方式采用二进制编码,此外也有整数、实数、文字等编码方式.在本文提出的基于遗传算法的资源调度算法中,一个个体的编码用于表示个人工作列表调度问题的一个可能解.这个编码字符串由所有工作项的工作安排组成,每一个工作安排由两部分组成:工作项实例以及它被安排到的工作列表序号.以图 4 为例,“a_1(2)”表示工作项 a_1 被分配到个人工作列表 2 执行.

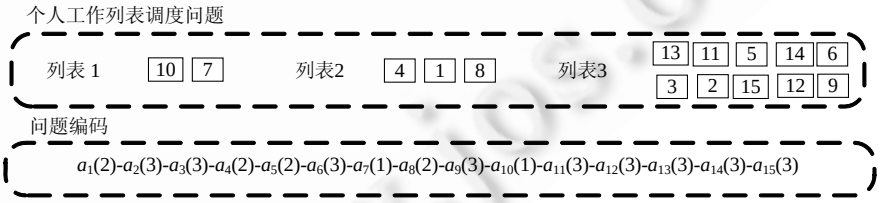


Fig.4 Personal worklist scheduling problem encoding

图 4 个人工作列表管理问题编码

初始化分配算法应用启发式的随机分配方法,将所有工作项随机地分配到可行工作列表中或者进入到不可行工作列表中.基于此问题的背景,工作项的可行工作列表定义如下:

定义 3(工作项 a 的可行工作列表 W). 已知工作列表 W 已经包含 n 个工作项$\{a_1,a_2,\ldots,a_n\}$(n>0),它们的开始时间遵循($s^{a_1} \leqslant s^{a_2} \leqslant s^{a_3} \leqslant \ldots \leqslant s^{a_n}$).如图 5 所示,工作列表 W 必须同时满足下面两个条件才可以成为工作项 a

的可行工作列表:

(1) 工作项 a 的开始时间不能晚于 W 中前 k 个工作项的最晚完成时间:

$$s^a \geqslant \max\{(s^{a_1}+d_{\max}^{a_1}),(s^{a_2}+d_{\max}^{a_2}),\ldots,(s^{a_k}+d_{\max}^{a_k})\}.$$

(2) 工作项 a 的最晚完成时间必须早于 W 中第 k+1 个工作项 a_{k+1} 的开始时间:

$$s^a+d_{\max}^a \leqslant s^{a_{k+1}}.$$

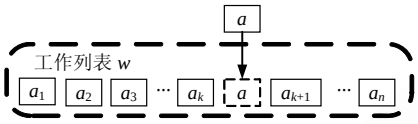


Fig.5 Example feasible worklist w for activity instance a

图 5 工作项 a 的可行工作列表 w

在不考虑可行工作列表中联合执行成功率的情况下,初始化分配算法首先初始化一个调度分配方案,使得尽可能多的工作项进入到可行工作列表中,然后再逐步进行优化.算法说明如下:

算法 1. 启发式随机分配算法.

(1) 从所有 k 个还未被分配的工作项列表$\{a_1,a_2,\ldots,a_k\}$中选取工作项 a_i;

(2) 在所有工作列表$\{W_1,W_2,\ldots,W_m\}$中寻找工作项 a_i 的可行工作列表,无时间冲突条件参见定义 3;

(3) 如果 a_i 有一个或者多个可行工作列表,那么随机选择一个将 a_i 加入到该可行工作列表中;否则,将 a_i 加入到不可行工作列表中;

(4) 如果还有工作项未被分配,则跳到步骤 1.

3.3 适应度函数和终止条件

在遗传算法中,适应度函数用来度量群体中个体的质量.由于个人工作列表资源调度的目标是在满足系统用户要求的联合执行成功率的条件下减小总体延误代价,适应度函数可分为两个:时间适应度函数和延误代价适应度函数.时间适应度函数描述了系统用户对于时间方面的成功率需求,使用所有可行工作列表的联合执行成功率(定义 1 中的 r^S)来表示适应度函数的时间适应度;另一方面,当某个群体中有一个或多个个体的时间适应度满足了成功率安全阈值时,延误代价适应度函数倾向于选择具有最小总体延误代价的个体作为最优解.所以,延误代价适应度函数使用不可行工作列表中所有工作项的总体延误代价(定义 2 中的 c^U)来计算.综上所述,对于个体 I,完整的适应度函数 $F(I)$可定义为

$$F(I)=\begin{cases} r^S, & \text{if } r^S < \text{safe threshold} \\ c^U, & \text{otherwise} \end{cases}.$$

基于这样的适应度函数,可以定义遗传算法的终止条件为:在一代群体生成后,当群体中至少包含一个个体,其适应度值(时间适应度部分)大于安全阈值时,遗传算法停止生成下一代群体,在此群体中可能会出现多个满足成功率条件的个体;否则,算法将会在进化指定数目的群体之后自动停止,例如 200 代.

3.4 遗传算子的定义

遗传算法中,3 种遗传算子(选择、交叉和变异)的主要目的是希望能够通过遗传算子使得当前个体获得更好的下一代个体[25,32].所以,遗传算法中的 3 个遗传算子尝试着将尽可能多的工作项放置到可行工作列表中,从而获得更好的后代个体.

- 选择算子用来保留群体中最好的一些个体以进行后续的进化.选择过程使用轮盘赌选择法选择最好的个体.具体来说,个体的轮盘赌选择基于适应度函数对于每一个个体的评价值,适应度越大的个体越容易被选择进入到下一代群体中.
- 交叉算子是遗传算法中起核心作用的遗传操作.所谓交叉,是指两个父代个体的部分结构加以替换重

组而生成新个体的操作,用于结合并且重新安排当前群体中的两个个体.这一算法的交叉算子通过交换两个个体中相应工作项被安排的工作列表的序号来获得更好的后代,并选择交叉概率为 0.6,即选择60%的个体进行交叉操作.详细来说,交叉算子采用如下实现方式:

(1) 按照轮盘赌选择方法从当前群体中随机选择两个双亲个体;

(2) 分别将双亲个体中执行成功率最小的一个工作项转移到不可行工作列表;

(3) 从一个个体中随机选取一段处于可行工作列表中的工作项,在另一个个体中找到这些工作项,对于每个都处于可行工作列表中的工作项,交换该工作项在两个个体中的工作列表序号;

(4) 在新群体中用新生成的个体替换被选择的个体.

- 变异算子是对群体中个体串的某些基因座的基因值作变动,用于提供给子代个体一些其双亲个体都不具备的特性,从而帮助遗传算法尝试之前没有考虑到的新的更好的基因材料.在本文的遗传算法里,变异算子将现有个体中每一个工作项分配的工作列表替换成一个随机选择的可行工作列表.我们选用基本变异算子,并选定变异概率为 0.1.具体的实现方法如下:

(1) 按照事先设定的变异概率从新的群体中按照轮盘赌法则选择需要变异的个体;

(2) 对每个需要变异的个体,随机选择多个工作项,针对每个工作项,寻找其可行工作列表:如果它有多个可行工作列表,则随机选择一个列表替换其原来分配的列表;否则,保持原来分配的列表;

(3) 在新群体中用新生成的个体替换被选择的个体.

每一代群体通过选择算子和交叉算子,再经过变异算子作用在生成的群体上之后,最终生成新一代群体作为下一次进化的基础.由于涉及到轮盘赌、选择、变异、交叉等算子,并需要很多轮随机化搜索,因此,遗传算法要比基于分配规则的调度算法复杂得多.但随着分布式计算、网格计算、云计算等技术的迅速发展,实际应用中,基于遗传算法的调度不会对工作项的执行产生较大影响.

应用遗传算法的主要过程见算法 2.

算法 2.

/ //*K*:待分配工作项数量;*m*:可行工作列表个数,即执行人个数

//*char*[·] *workitem*:工作项

//*List*⟨*workitem*⟩ *intialtowlist*:待分配的工作项列表,有 *K* 个工作项

//*List*⟨*workitem*⟩ *wlist*:单个个人可行工作列表,

//*List*⟨*wlist*⟩ *clist*:所有可行工作列表,共有 *m* 个

//*List*⟨*workitem*⟩ *infwlist*:不可行工作列表,有 1 个

//*generations*:迭代次数;*population*:群体大小

//*List*⟨*double*⟩ *fitValuelist*:可行工作列表的执行成功率,适应度值

//*List*⟨*double*⟩ *tocost*:不可行工作列表总体延误代价,适应度值

输入:*intialtowlist*[*K*]:前端接收到待分配工作项列表,有 *K* 个;

safethvalue:可行工作列表中所有工作项的联合执行成功率终止条件,由用户输入;

输出:

clist:所有可行工作列表,共有 *m* 个;

infwlist:不可行工作列表,只有 1 个.

1: 按照算法 1 产生一个个体,并按照指定的群体大小初始化第一代群体

2: int *generations*=0; //迭代的次数初始为 0

3: *fitValuelist*=*calfitvalue*(*clist*);

//根据适应度函数计算当前群体各个个体的联合执行成功率,结果存在 *fitValuelist* 中

4: while (*fitValuelist*[·]<=*safethvalue*) //当没有染色体符合终止条件,循环执行

5: *selection*(·); //根据适应值的大小选择优良的父代,结果分别保存在 *clist* 和 *infwlist* 中

6: *crossover*(·); //在新个体中随机配对,在配对个体中随机选择交叉位,结果保存在 *clist* 和 *infwlist* 中
7: *mutation*(); //在新个体中随机选择变异点,结果保存在 *clist* 和 *infwlist* 中
8: //开始计算适应度函数值
fitValuelist=calfitvalue(*clist*); //重新计算各个个体的联合执行成功率
9: if (*generations*==200)
10: break; //如果超过 200 次迭代,停止迭代
11: end if;
12: *generations*++; //迭代次数加 1
13: end while;
14: *tocost=calcost*(*infwlist*); //计算总体延误代价,结果存在 *tocost* 中
15: 从群体中选择满足成功率阈值要求的、具有最小延误代价的个体

4 实验与比较

4.1 实验设计

虽然大部分工作流实例调度算法不能直接应用于个人工作列表资源调度,但一些简单的基于分配规则的调度算法可以直接应用于个人工作列表资源调度,文献[5]中的个人工作列表资源调度算法也采用了基于分配规则的调度算法(LDF 和 LSTF).为了对比分析遗传算法的优越性,我们选择实现了 7 种典型的基于分配规则的调度算法.本文涉及的遗传算法的人口大小为 50,即 50 种可能的调度方案,使用 200 代来优化这些调度.变异操作按照文献[21]提出的方法进行定义.采用了通用的遗传算法参数,交叉率为 0.6,变异概率为 0.1[25,32].

基于分配规则的调度算法主要采用分配规则(dispatching rules)指导工作流管理系统选择工作项.工作流管理系统优先从工作项等待队列中选取最符合这个分配规则的一个工作项进行调度,将这个工作项分发到资源的工作列表中,或者从可行工作列表中收回.很多经典的工作流管理系统采用了这种调度策略.在我们实现的基于分配规则的调度算法中,首先使用算法 1 中描述的启发式随机分配算法,在不考虑可行工作列表中工作项联合执行成功率的情况下,将所有工作项进行分配.初始分配完成之后,如果当前分配方案的可行工作列表中工作项联合执行成功率小于需求值,则按照相应分配规则选取可行工作列表中最符合该规则的工作项,将其从可行工作列表中移除,插入到不可行工作列表中.重复这个操作,直到可行工作列表中工作项联合执行成功率满足要求.本实验选择了 7 种基于分配规则的典型调度算法,它们分别是:

(1) FIFO(first-in-first-out,先入先出规则[18,20,32]):优先从可行工作列表中移除到达时间最晚的工作项;
(2) SIRO(service in random order,随机选择规则[18,20,32]):从可行工作列表中随机地移除一个工作项;
(3) SCF(small cost first,最小代价优先[18,20,32]):优先移除具有最小延误代价的工作项;
(4) LEDF(longest expected duration first,最长期望执行时间优先[18,20,32]):优先从可行工作列表中移除具有最长的执行时间期望值的工作项;
(5) SRF(smallest ratio first,最低成功率优先[32]):优先移除具有最小执行成功率的工作项;
(6) LDF(latest deadline first,最晚截止时间优先[2]):优先移除具有最晚截止时间的工作项;
(7) LSTF(lowest slack time first,最小富裕时间优先[2]):富裕时间是指工作项的执行时间期望值和工作项拥有的执行时间(截止时间与开始时间相减)的差值.该算法优先移除具有最小富裕时间的工作项.

4.2 应用示例

下面使用图 2 中的实例来说明遗传算法实验.表 1 分别给出了图 2 中 15 个活动实例的 6 个相关参数值:执行时间(包括期望值和标准偏差)、开始时间、截止时间、执行成功率、延误代价.其中,执行时间期望值、执行时间标准偏差、开始时间和截止时间的单位是小时,执行成功率的单位是百分比,延误代价的单位是基础的代价单位.另外,执行时间期望值的取值范围为 0~150,开始时间的取值范围为 0~60.为方便将精力集中于遗传算法

和其他 7 种算法的比较,将初始执行成功率取值为 95%~100%范围内的随机数.这些参数值是参照多个制造企业的工作流日志和相关文档获得的.

Table 1 15 activity instances' six related variables in the example of Fig.2

表 1 图 2 实例中 15 个工作项的 6 个相关参数

工作项	执行时间		开始时间	截止时间	执行成功率(%)	延误代价
	期望值	标准偏差				
a_1	12.79	0.43	96.43	109.97	95.81	18.70
a_2	10.37	0.35	86.17	97.41	94.19	15.87
a_3	25.47	0.84	84.92	112.89	96.72	17.79
a_4	17.17	0.56	22.81	41.14	97.06	13.40
a_5	33.84	1.10	24.49	61.40	97.60	12.30
a_6	22.70	0.78	65.07	89.49	96.51	15.90
a_7	18.88	0.64	129.45	150.13	96.64	10.86
a_8	32.24	1.09	113.54	148.42	94.10	17.76
a_9	22.67	0.77	122.15	146.90	96.53	19.64
a_{10}	26.76	0.92	46.06	74.79	98.32	15.86
a_{11}	58.12	1.20	19.99	82.33	97.14	12.90
a_{12}	21.32	0.73	122.01	145.11	94.12	16.07
a_{13}	55.48	1.81	18.04	77.87	96.04	16.76
a_{14}	32.21	1.07	24.94	59.94	93.42	19.06
a_{15}	5.40	0.18	92.94	98.80	96.42	15.33

假设实例中的联合执行成功率的安全阈值设定为 75%,初始的个人工作列表是空的,则共有从 a_1~a_{15} 的 15 个活动实例被加在个人工作列表中.

图 6 给出了在加入了前 7 个活动实例之后的 8 种算法的调度结果(GA,SRF,FIFO,SIRO,SCF,LEDF,LDF,LSTF).图 6 中,每一个个人工作列表(列表 1、列表 2、列表 3)有其对应的一个可行工作列表,3 个工作列表有一个唯一的不可行工作列表 U.在调度之后,遗传算法能够将所有 7 个工作项分配到它们的可行工作列表,但是其他 7 种算法都不得不移除两个或多个工作项到不可行工作列表,因此,遗传算法的延误代价为 0.与此相比,另外 7 种算法的延误代价从 33.66~68.32.

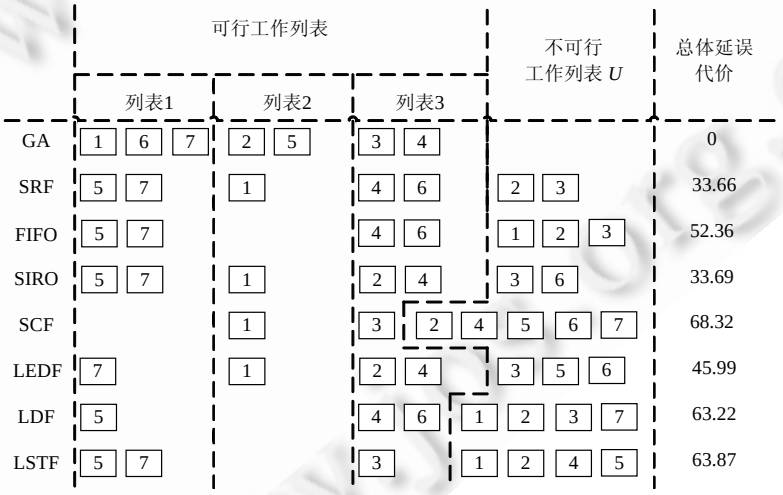


Fig.6 Example scheduling results after adding the first seven activity instances

图 6 前 7 个工作项的调度结果

图 7 显示了基于上述 15 个工作项的 8 种算法的调度结果.遗传算法中,不可行工作列表 U 有最少的工作项(4 个),因此,遗传算法获得了最好的调度结果和最少的延误代价:66.93.

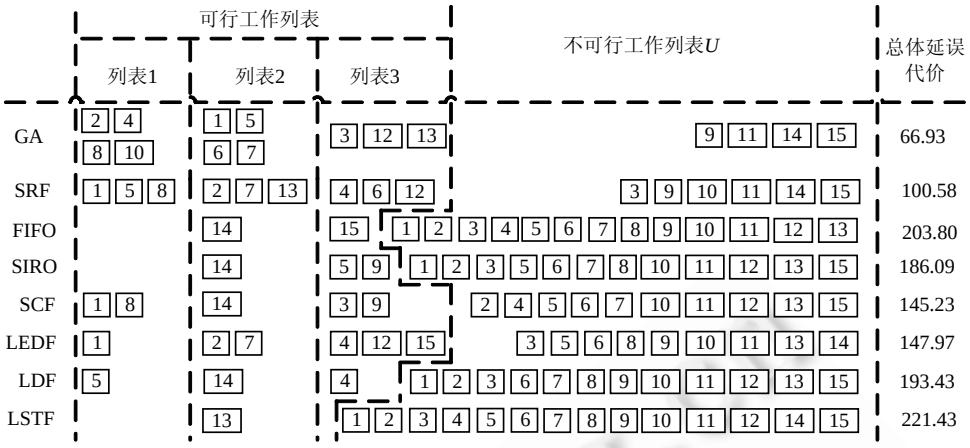


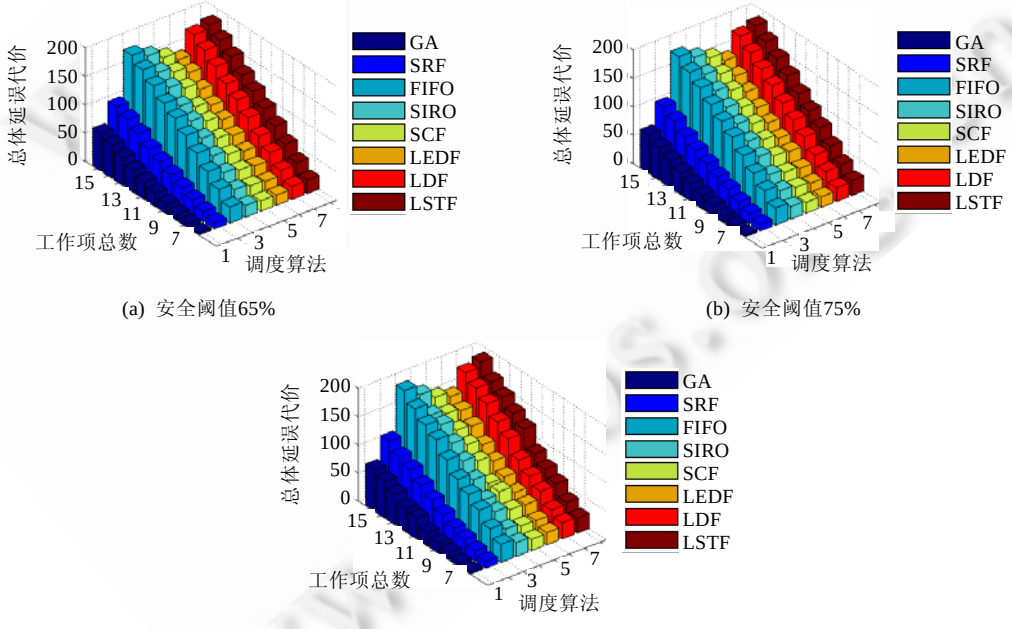
Fig.7 Example scheduling results after adding the first 15 activity instances

图 7 包含 15 个工作项的 8 种算法的调度结果

4.3 实验结果和评估

本实验对应 3 个可执行个人工作列表,共包含 10 个步骤,从 6 个工作项开始,每个步骤增加 1 个工作项.因此,从步骤 1~步骤 10,工作项总数分别为 6~15 个.每一个步骤都实现了 8 种调度算法,在满足给定的成功率安全阈值的前提下,计算了不可行工作列表 U 的总体延误代价.本实验共设置了 3 个安全阈值:65%,75%和 85%.为充分对比不同算法的性能,对应每个安全阈值的实验执行次数都超过了 1 000 次,因此,该实验模拟执行次数超过了 3 000 次.为公平起见,每步实验的总体延误代价取值为所有 1 000 个实验的延误代价的平均值.

图 8 分别展示了对应于安全阈值设定为 65%,75%和 85%时的实验结果对比分析三维图.



(a) 安全阈值65% (b) 安全阈值75%

(c) 安全阈值 85%

Fig.8 Simulation experiment results of eight scheduling algorithms

图 8 8 种调度算法在不同阈值下的实验结果对比

在图 8 的三维坐标图中,x 轴代表这 8 种算法,y 轴代表插入的工作项数量,z 轴代表不可行工作列表 U 中的工作项延误代价.

实验结果表明,对 65%,75%和 85%这 3 个联合执行成功率安全阈值而言,遗传算法的延误代价要小于其他 7 种算法;除此之外,当工作列表由于插入工作项而变大时,遗传算法和其他算法之间延误代价的差距会不断拉大;而且,遗传算法的性能会随着安全阈值的增大而提升,这意味着,当执行者需要更高的稳定性时,遗传算法比其他算法更能降低延误成本.因此可以断定,遗传算法比其他 7 种算法更能减小延误代价.工作负荷越大,安全阈值的要求越高,遗传算法的优势就越明显.从算法性能上看,遗传算法的运行时间要明显高于其他 7 种算法,但相对于本实验中各个工作项的执行时间而言可以忽略不计.如果遗传算法的运行时间大于工作项的平均执行时间,在一个工作项执行完成后,其后续工作项的调度方案可能还未生成,遗传算法会失去其应用优越性.我们可从两个方面来减少遗传算法的运行时间:一方面,可对遗传算法进行优化.另一方面,可采用并行技术实施遗传算法.

5 结论与展望

个人工作列表资源调度算法的应用,可以对业务流程管理起到巨大的提升作用.虽然已经存在的工作流调度引擎提供了一些复杂算法来描述和分析工作流过程,但对于个人工作列表的研究还处于起步阶段.本文提出的优化个人工作列表资源调度的启发式遗传算法,综合考虑了工作项创建时间、执行时间、执行成功率等要素的不确定性,以及多个个人工作列表的综合调度问题,可向每一个执行者明确地推荐工作项列表,其联合执行成功率能够满足一般的联合执行成功率要求,同时能够减少所有工作列表的总体延误代价.本文使用了一个具体实例来展示这种遗传算法,并通过与基于分配规则的多种调度算法相比较证明了这一调度算法的优越性.

本文提出的调度算法能够应用到工作流执行阶段,以提升个人工作列表中工作项执行的性能.另外,随着网格工作流、云工作流等工作流技术的不断发展和深入应用,遗传算法可以较好地使用并行技术,显著降低搜索所需要的时间消耗[32];传统的网络数据中心也逐渐承担起资源管理中心和流程管理中心的职责.本文的算法也可以用于解决网格工作流中多个任务的资源调度问题,以及完成网络数据中心的资源调度工作.

但是本文的工作还存在一个较大缺陷,即实验使用的数据均为模拟地随机生成的数据,与真实工作流管理系统中工作项的数据特点可能有所差别.未来我们将对从真实工作流管理系统中收集的工作项日志进行分析,获得其关键属性的分布特性,然后再据此生成调度算法所使用的实验数据.同时,要提供一个相对成熟与方便的实验环境,能够更方便、更快捷地进行实验的扩展,以更好地获得算法对比实验结果.另外,还要进一步优化个人工作列表调度算法,提升运行效率和吞吐量等其他调度性能.

References:

[1] Russell N, ter Hofstede AHM, Edmond D, van der Aalst WMP. Workflow resource patterns. Technical Report, Eindhoven: Eindhoven University of Technology, 2005. http://ww.workflowpatterns.com/documentation/documents/Resource%20Patterns%20BETA%20TR.pdf

[2] Eder J, Pichler H, Gruber W, Ninaus M. Personal schedules for workflow systems. In: van der Aalst WMP, ter Hofstede AHM, Weske M, eds. Proc. of the Conf. on Business Process Management. LNCS 2678, Heidelberg: Springer-Verlag, 2003. 216−231.

[3] Eder J, Eichner H, Pichler H. A probabilistic approach to reduce the number of deadline violations and the tardiness of workflows. In: Meersman R, Tari Z, Herrero P, *et al*., eds. Proc. of the Conf. on the Move to Meaningful Internet Systems 2006 (OTM 2006) Workshops. LNCS 4277, Heidelberg: Springer-Verlag, 2006. 5−7. [doi: 10.1007/11915034_3]

[4] Rinderle S, van der Aalst WMP. Life-Cycle support for staff assignment rules in process-aware information systems. Technical Report, Beta Working Paper Series, No.213, Eindhoven: Eindhoven University of Technology, 2007.

[5] Wang K, Deng TQ, Zhou DJ. Research on agent of collaboration workflow. System Engineering and Electronics, 1999,21(8):48−51 (in Chinese with English abstract).

[6] Wang K, Deng TQ, Zhou DJ. Research on official document workflow system. In: Siriruchatapong P, Lim Z, Barthes JP, eds. Proc. of the 2nd Int'l Workshop on CSCW in Design. Beijing: Int'l Academic Publishers, 1997. 202−208.

[7] Wang K, Deng TQ, Zhou DJ. Structural workflow management on official documents management. In: Bian JN, Liu MY, eds. Proc. of the 9th Chinese Conf. on Design Automation of Digital Systems and Concurrent Engineering. Beijing: Publishing House of Electronics Industry, 1999. 81−86 (in Chinese).

[8] Deng TQ, Wang K. Design and implementation of official document management workflow system. In: Wu QY, Qian YL, eds. Proc. of the 4th Chinese Conf. on Intelligent Computer Interfaces and Intelligent Applications. Beijing: Publishing House of Electronics Industry, 1999. 108−113 (in Chinese).

[9] Eder J, Pichler H, Vielgut S. Avoidance of deadline-violations for inter-organizational business processes. In: Vasilecas O, Eder J, Caplinskas A, eds. Proc. of the 7th Int'l Baltic Conf. on Databases and Information Systems. Paris: IEEE Computer Society, 2006. 33−40. [doi: 10.1109/DBIS.2006.1678471]

[10] Liu YB. Research on workflow runtime intelligent staff assignment technology [Ph.D. Thesis]. Beijing: Tsinghua University, 2008 (in Chinese with English abstract).

[11] Liu S, Fan YS. Method for analyzing staying-time of instances in workflow models with resources constraints. Chinese Journal of Electronics, 2005,33(10):1867−1871 (in Chinese with English abstract).

[12] Dumas M, van der Aalst WMP, ter Hofstede AHM. Process-Aware Information Systems: Bridging People and Software through Process Technology. New York: John Wiley & Sons, Inc., 2005.

[13] Liu JX, Zhang SS, Cao J, Hu JM. An agent enhanced framework to support pre-dispatching of tasks in workflow management systems. In: Han YB, Tai T, Wikarski D, eds. Proc. of the 1st Int'l Conf. on Engineering and Deployment of Cooperative Information Systems. LNCS 2480, Heidelberg: Springer-Verlag, 2002. 80−89. [doi: 10.1007/3-540-45785-2_6]

[14] Sakellariou R, Zhao H, Tsiakkouri E, Dikaiakos M. Scheduling workflows with budget constraints. In: Gorlatch S, Danelutto M, eds. Proc. of the CoreGRID Integration Workshop 2005 on Integrated Research in Grid Computing. Heidelberg: Springer-Verlag, 2007. 189−202.

[15] Jia Y, Buyya R, Chen KT. Cost-Based scheduling of scientific workflow applications on utility grids. In: Stockinger H, Buyya R, Perrott R, eds. Proc. of the 1st Int'l Conf. on e-Science and Grid Computing. Melbourne: Computer Society, 2005. 147−154. [doi: 10.1109/E-SCIENCE.2005.26]

[16] Jia Y, Buyya R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Scientific Programming, 2006,14(3-4):217−230.

[17] Sun RZ, Shi ML. Schedule of activity instances in workflow management system. Journal of Software, 2005,16(3):400−406 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/400.htm [doi: 10.1360/jos160400]

[18] Kafeza E, Karlapalem K. Temporally constrained workflows. In: Chi L, Hui K, Lee DL, eds. Proc. of the 5th Int'l Computer Science Conf. on Internet Applications. LNCS 1749, Heidelberg: Springer-Verlag, 1999. 149−154. [doi: 10.1007/978-3-540-46652-9_24]

[19] Baggio G, Wainer J, Ellis C. Applying scheduling techniques to minimize the number of late jobs in workflow systems. In: Haddad HM, Papadopoulos GA, eds. Proc. of the 2004 ACM Symp. on Applied Computing. New York: ACM Press, 2004. 1396−1403. [doi: 10.1145/967900.968180]

[20] Tramontina GB, Wainer J. Modeling the behavior of dispatching rules in workflow systems: A statistical approach. In: Fuks H, Lukosch S, Salgado AC, eds. Proc. of the 11th Int'l Workshop on Groupware: Design, Implementation, and Use (CRIWG 2005). LNCS 3706, Heidelberg: Springer-Verlag, 2005. 208−215. [doi: 10.1007/11560296_16]

[21] Eder J, Pichler H. Duration histograms for workflow systems. In: Rolland C, Brinkkemper S, Saeki M, eds. Proc. of the Engineering Information Systems in the Internet Context. Massachusetts: Kluwer Academic Publishers, 2002. 239−253.

[22] Liu YB, Wang JM, Yang Y, Sun JG. A semi-automated approach to workflow staff assignment. Computers in Industry, 2008,59(5): 411−419. [doi: 10.1016/j.compind.2008.02.003]

[23] Chen JJ, Yang Y. Taxonomy of grid workflow verification and validation. Concurrency and Computation: Practice and Experience, 2007,20(4):347−360. [doi: 10.1002/cpe.v20:4]

[24] Chen JJ, Yang Y. Adaptive selection of necessary and sufficient checkpoints for dynamic verification of temporal constraints in grid workflow systems. ACM Trans. on Autonomous and Adaptive Systems (TAAS), 2007,2(2):1−25. [doi: 10.1145/1242060.1242063]

[25] Yu J, Tian GZ, Cao YD, Sun XH. A resource allocating algorithm in arid workflow based on critical regions reliability. Journal of Computer Research and Development, 2009,46(11):1821−1829 (in Chinese with English abstract).

[26] Xiao ZJ, He QM, Chen Q. A multilevel model of task assignment in fuzzy situations of workflow. Journal of Computer Research and Development, 2007,44(2):302−309 (in Chinese with English abstract). [doi: 10.1360/crad20070217]

[27] Wang Y, Hu CM, Du ZX. QoS-Awared grid workflow schedule. Journal of Software, 2006,17(11):2341−2351 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/2341.htm [doi: 10.1360/jos172341]

[28] Mi HB, Wang HM, Yin G, Shi DX, Zhou YF, Yuan L. Resource on-demand reconfiguration method for virtualized data centers. Journal of Software, 2011,22(9):2193−2205 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4056.htm [doi: 10.3724/SP.J.1001.2011.04056]

[29] Karve A, Kimbrel T, Pacifici G, Spreitzer M, Steinder M, Sviridenko M, Tantawi A. Dynamic placement for clustered Web applications. In: Carr L, De Roure D, Iyengar A, eds. Proc. of the 15th Int'l Conf. on World Wide Web. New York: ACM Press, 2006. 593−604. [doi: 10.1145/1135777.1135865]

[30] Holland JH. Genetic algorithms. Scientific American, 1992,267(1):66−72. [doi: 10.1038/scientificamerican0792-66]

[31] Zomaya AY, The YH. Observations on using genetic algorithms for dynamic load-balancing. IEEE Trans. on Parallel and Distributed Systems, 2001,12(9):899−911. [doi: 10.1109/71.954620]

[32] Brucker P. Scheduling Algorithms. 5th ed., Heidelberg: Springer-Verlag, 2007.

附中文参考文献:

[5] 王恺,邓铁清,周堤基.协同工作流 Agent 的研究.系统工程与电子技术,1999,21(18):48−51.

[7] 王恺,邓铁清,周堤基.公文处理中的结构化工作流管理.见:第 9 届全国数字系统设计自动化暨全国协同设计会议论文集.北京:电子工业出版社,1999.81−86.

[8] 邓铁清,王恺.公文生成工作流系统的设计与实现.见:吴泉源,钱跃良,主编,第 4 届全国计算机智能接口与智能应用学术会议论文集.北京:电子工业出版社,1999.108−113.

[10] 刘英博.工作流运行时人员智能分配技术研究[博士学位论文].北京:清华大学,2008.

[11] 刘胜,范玉顺.资源约束下实例在工作流中停留时间分析方法.电子学报,2005,33(10):1867−1871.

[17] 孙瑞志,史美林.工作流活动多实例的调度控制.软件学报,2005,16(3):400−406. http://www.jos.org.cn/1000-9825/16/400.htm [doi: 10.1360/jos160400]

[25] 于炯,田国忠,曹元大,孙贤和.基于关键区间可靠度的网格工作流资源分配算法.计算机研究与发展,2009,46(11):1821−1829.

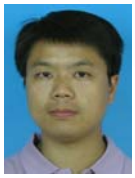
[26] 肖郑进,何钦铭,陈奇.模糊环境中工作流任务分配的多级模型.计算机研究与发展,2007,44(2):302−309. [doi: 10.1360/crad20070217]

[27] 王勇,胡春明,杜宗霞.服务质量感知的网格工作流调度.软件学报,2006,17(11):2341−2351. http://www.jos.org.cn/1000-9825/17/2341.htm [doi: 10.1360/jos172341]

[28] 米海波,王怀民,尹刚,史殿习,周扬帆,袁霖.一种面向虚拟化数字中心资源按需重配置方法.软件学报,2011,22(9):2193−2205. http://www.jos.org.cn/1000-9825/4056.htm [doi: 10.3724/SP.J.1001.2011.04056]

邓铁清(1964－),男,湖北天门人,高级工程师,CCF 高级会员,主要研究领域为信息系统,知识工程,工作流,数据中心.

刘英博(1978－),男,博士,讲师,主要研究领域为工作流,PLM.



任艮全(1978－),男,博士生,主要研究领域为工作流,PLM,数据起源.