软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(6):1207−1221 [doi: 10.3724/SP.J.1001.2013.04259]

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

基于路径平衡的工作流费用优化方法*

刘灿灿[1], 张卫民[2], 骆志刚[2]

[1](空军军训器材研究所,北京 100195)

[2](国防科学技术大学 计算机学院 软件所,湖南 长沙 410073)

通讯作者: 刘灿灿, E-mail: cancanliu@nudt.edu.cn

摘 要: 针对效用网格下截止期约束的工作流费用优化问题,提出了路径平衡(path balance,简称 PB)算法,对工作流中各路径长度进行调整,并提出基于路径平衡的费用优化(path balance based cost optimization,简称 PBCO)算法. PBCO 基于 PB 的计算结果设置初始约束时间,充分利用了工作流的费用优化空间.同时,采用逆向分层策略对任务进行分层,并根据各层任务数按比例分配冗余时间,有效地增大了多数任务的费用优化空间,进一步改善了工作流的费用优化效果.实验结果表明,PBCO 比另外几种著名算法(如 DET,DBL 等)改进了约 35%.

关键词: 工作流调度;效用网格;路径平衡;截止期约束;费用优化;逆向分层

中图法分类号: TP301 文献标识码: A

中文引用格式: 刘灿灿,张卫民,骆志刚.基于路径平衡的工作流费用优化方法.软件学报,2013,24(6):1207−1221. http://www.jos.org.cn/1000-9825/4259.htm

英文引用格式: Liu CC, Zhang WM, Luo ZG. Path balance based heuristics for cost optimization in workflow scheduling. Ruan Jian Xue Bao/Journal of Software, 2013,24(6):1207−1221 (in Chinese). http://www.jos.org.cn/1000-9825/4259.htm

Path Balance Based Heuristics for Cost Optimization in Workflow Scheduling

LIU Can-Can[1], ZHANG Wei-Min[2], LUO Zhi-Gang[2]

[1](Air Force Training Equipment Institute, Beijing 100195, China)

[2](Institute of Software, School of Computer Science, National University of Defense Technology, Changsha 410073, China)

Corresponding author: LIU Can-Can, E-mail: cancanliu@nudt.edu.cn

Abstract: In order to address the trade-off problems between time and cost in the grid workflow scheduling with deadline constraints, this study proposes a new algorithm named path balance (PB) to adjust the length for each path in workflow and develop a heuristic referred as path balance based cost optimization (PBCO). PBCO makes full use of the workflow cost optimization space by setting a deadline for each task, based on the results of PB, and enlarges the optimization space for multitude tasks by distributing the redundancy time, based on the quantities of tasks at each level, this is divided by using the bottom level strategy. According to the experimental results, the average execution cost of PBCO decreases by about 35% relative to other famous heuristics, such as DET and DBL.

Key words: workflow scheduling; utility grids; path balance; deadline constraint; cost optimization; bottom level

随着计算网格、效用网格和网格经济模型的研究与发展[1],在网格环境中对服务进行付费使用已经成为一种趋势.服务提供者在网格资源上部署了多种服务并提供服务的统一访问接口,同时根据服务消费的资源数(如 CPU 数量、内存大小、存储空间、网络带宽等)与服务的 QoS 性能(如执行速度、可靠性、响应时间等)制定收费标准,在为用户提供服务的同时也向用户收取一定费用.在这类效用网格中对工作流表示的复杂计算流程进行调度时,通常需要考虑工作流的执行时间、执行费用、可靠性及数据质量等多个目标[2−4],这些目标间相互联系且相互制约,如何在多个目标间进行权衡并达到多目标综合性能的最优值,是一个非常重要且具有挑战性的

* 基金项目: 国家高技术研究发展计划(863)(2006AA01A123); 国家自然科学基金(60903042)
收稿时间: 2010-12-21; 修改时间: 2011-10-17; 定稿时间: 2012-04-16

课题[5].在这类多目标优化问题中,工作流的执行时间与执行费用是两个备受关注的调度目标,而截止期约束的工作流费用优化问题更是当前的研究热点之一.

针对工作流的截止期约束-费用优化问题,有多个学者进行了大量研究,最具代表性的有东南大学苑迎春等人提出的几种启发式算法:截止期约束的逆向分层算法 DBL(deadline bottom level)[6]、逆向/正向串归约算法 BSRD/FSRD(backward/forward serial reduction with deadline)[7]、截止期约束的最早树算法 DET (deadline early tree)[8]以及基于优先级规则 BFTCS(best fit with time-depending coupling strength)的迭代算法[9]等.其中,DBL 基于工作流的逆向深度对任务进行分层,并在各层间分配截止时间,有效利用了各任务的同步完成特征,增大了各任务的费用优化空间,达到了工作流的费用优化效果;BSRD/FSRD 在 DBL 的基础上进行了改进,在 DBL 为任务分配截止时间的基础上,合并工作流中可归约串的费用优化空间,采用动态规划方法在可归约串组内进行费用优化.这种串归约方法充分利用了各归约串内产生的时间碎片,进一步改善了工作流的费用优化效果;DET 采用串归约方法为关键路径上的任务选择服务,并根据关键任务的完成时间对非关键任务的约束时间进行设置,也达到了较好的费用优化效果;BFTCS 则是一种基于优先级规则的迭代算法,该算法基于工作流的时间耦合强度建立优先级规则并基于该规则选择任务,在满足时间约束的条件下,将所选任务的当前服务替换成执行时间稍长但执行费用较低的服务,从而优化工作流的执行费用.Yu 等人也进行了大量研究,提出了基于时间分配的 TD(time distribution)算法[10].该算法根据各任务权重在工作流路径中所占的比例来分配截止时间,在分配时间内进行费用优化.此外,基于遗传算法的费用优化算法[11]、基于条件检查的约束违背-逆向跟踪算法 BT(back track)[12]、基于粒子群、蚁群算法等的启发式搜索算法[13−15]也是研究较多的几类方法.以上各类算法在解决工作流的截止期约束-费用优化问题时均取得了较好的效果,但也都存在一些缺陷,如:DBL 及在 DBL 基础上改进的 BSRD/FSRD 算法均存在适用性受限的问题,当约束时间小于最小分层完成时间时算法不适用,此时采用最小关键路径法(minimum critical path,简称 MCP)[6]进行调度,MCP 为关键路径上的任务选择最快服务执行,仅对非关键路径上的任务进行部分优化,因此降低了算法性能;而 BSRD/FSRD 的费用优化效果也依赖于工作流中可归约串所占的比例;DET 虽然最大限度地优化了工作流中关键路径的执行费用,但由于 DET 仅考虑了关键路径上任务的资源特征,忽略了非关键任务的资源特征及工作流的结构特征,因而使得非关键任务的费用优化效果比较差,算法整体性能不理想.此外,基于遗传算法、粒子群等的启发式搜索算法在全局解空间中进行搜索,当任务规模与可选服务的规模较大时,算法通常具有较大的时间开销.

在深入分析并研究工作流截止时间约束-费用优化问题的基础上,认为这类问题的关键在于总的费用优化空间(即给定约束时间)在各任务间的合理分配,由于工作流中各条路径长度各不相同,不同路径上任务的费用优化空间各异,因此工作流的结构特征是影响费用优化空间合理分配的重要因素之一.同时,在工作流调度过程中,某一拥有多个前驱的任务需等所有前驱执行完成后才能开始执行,如果所有前驱均能同时完成,则能够减少由于等待某一前驱完成而造成的其他前驱的费用优化空间的浪费.为达到这一目标,提出路径平衡算法 PB (path balance)对工作流结构进行调整,在保证工作流最小长度不变的条件下,逐步为短路径中的任务分配时间增量,最终达到工作流中所有路径长度相等并基于路径平衡结果进行初始约束时间分配;在初始时间分配后,采用逆向分层策略对工作流进行分层,基于各层任务数按比例分配冗余时间;最终,提出基于路径平衡的费用优化算法 PBCO(path balance based cost optimization).

本文的主要贡献有:

- 第一,设计并实现 PB 算法并基于该算法的计算结果进行初始时间分配,该过程为所有具有相同后继的任务分配相同的截止时间,达到了合理分配约束时间并充分利用费用优化空间的目标;
- 第二,基于各任务的最小执行时间进行路径平衡和初始时间分配,在此过程中工作流的最小长度不发生变化,当约束时间不小于工作流最小长度时,各任务在初始约束时间内一定能找到至少一个满足时间约束的服务,解决了 DBL 等算法的适用性受限问题;
- 第三,PBCO 算法基于逆向分层策略对工作流进行分层,根据各层任务数按比例分配冗余时间,由于为任务数较多的层分配了较大的费用优化空间,有效增大了多数任务的费用优化空间,因此整体上提

高了工作流的费用优化效果.

1 截止期约束的费用优化问题描述

工作流(W)通常通过有向无环图(directed acrylic graph,简称 DAG)进行描述:$DAG=\{V,E\}$,其中,$V=\{1,2,\ldots,n\}$表示 W 中的任务集,而 $E=\{(i,j)|i,j\in V\}$表示任务间的依赖关系.$(i,j)\in E$ 表示任务 i 与 j 的依赖关系,同时称 i 为 j 的前驱,j 为 i 的后继;i 的多个前驱为前驱列表,记为 $pre(i)$;多个后继为后继列表,记为 $succ(i)$;当 $pre(i)$全部执行完成时,i 便达到就绪条件.设定 W 中只有一个开始任务(记为 1)和一个结束任务(记为 n).对于 V 中的任意任务 i,在调度空间中均存在一个对应的候选服务列表,记为 $S(i)$,服务列表的长度(即服务个数)记为 $l(i)$.同一列表中,不同候选服务的执行时间和执行费用各不相同,$s_{ik}=(\tau_{ik},c_{ik})$表示 $S(i)$中第 $k(0\le k<l(i))$个服务,其中,τ_{ik} 与 c_{ik} 分别表示 s_{ik} 的执行时间与执行费用,设定执行时间较长的服务执行费用较低.

工作流的调度过程即为各任务在对应的候选服务列表中选择一个服务,假定所有服务都能提供与承诺一致的服务质量.工作流中所有任务的服务均选定之后,工作流的执行时间与执行费用便确定了.工作流的执行时间为该调度下的工作流长度,即为结束任务 n 的完成时间,执行费用为所有任务的执行费用之和.本文的优化目标即为 V 中的任意任务 i 在对应的服务列表 $S(i)$中选择一个服务 s_{ik},使得工作流的执行时间在约束时间(记为δ)之内,执行费用达到最小化,该问题可形式化描述为

$$\min\sum_{i=1}^{n}\sum_{k=0}^{l(i)-1}(c_{ik}\times x_{ik}) \tag{1}$$

$$\text{S.T.}\quad \beta_n\le\delta \tag{2}$$

$$\beta_j-\beta_i-\sum_{k=0}^{l(j)-1}\tau_{jk}\times x_{jk}\ge 0,\forall(i,j)\in E \tag{3}$$

$$\sum_{k=0}^{l(i)-1}x_{ik}=1,\forall i\in V \tag{4}$$

$$x_{ik}\in\{0,1\},\forall i\in V,0\le k<l(i) \tag{5}$$

其中,x_{ik} 为布尔函数.当 i 选择 s_{ik} 时,$x_{ik}=1$;否则,$x_{ik}=0$;β_i 为任务 i 的完成时间.公式(1)描述了工作流的费用优化目标;公式(2)描述了工作流的时间约束目标,其中,β_n 为结束任务 n 的完成时间;公式(3)描述了工作流的偏序关系;公式(4)和公式(5)表示每个任务只能选择且必须选择一个服务执行.

2 基于路径平衡的初始时间分配

在工作流的截止期约束-费用优化问题中,为减少由于等待某一前驱任务完成而引起的其他前驱费用优化空间的浪费,应充分利用工作流的同步完成特征,即尽量使得工作流中具有相同后继的任务均能同步完成.为达到这一目标,设计路径平衡算法 PB.设定工作流中任务的权值为该任务的执行时间,PB 算法的目标描述为:从某一任务(含开始任务)到工作流中其他任务(两任务间可达,含结束任务)间的多条路径长度相等.PB 算法对工作流中较短路径上任务的权值进行调整,逐步为这些任务分配时间增量,尽量达到各条路径的长度相等,并在此基础上进行初始约束时间分配.此外,PB 算法基于所有任务均选择最快服务进行计算的条件下,对短路径上任务的权值进行调整,该过程中始终保持工作流长度不发生变化,因此,当给定的约束时间不小于工作流最小长度时,各任务在分配的费用优化空间内一定能找到至少一个满足时间约束的解,从而解决 DBL 等算法的适用性受限问题.在对 PB 算法进行描述之前,先给出以下相关定义,以下定义均基于工作流中所有任务,均选择最快服务执行.

2.1 相关定义

定义 1(路径长度). 某一路径上所有任务的执行时间之和定义为该路径的路径长度.

定义 2(关键路径,critical path,简称 CP). 工作流中从开始任务到结束任务的最长路径定义为工作流的关键路径,记为 CP.

定义 3(工作流长度,*makespon*,简称 *mp*). 工作流中关键路径的长度定义为工作流长度,记为 *mp*.

定义 4(时间灵活度,temporal mobility,简称 TM). 经过任务 i 的最长路径长度与工作流长度之差定义为 i

的时间灵活度,记为 $TM(i)$.

定义 5(关键任务,critical task,简称 CT). 工作流中所有时间灵活度为 0 的任务定义为关键任务,记为 CT.

显然,关键路径上的任务均为关键任务.其他任务的时间灵活度可通过该调度下的最早与最迟开始时间进行求解.最早开始时间为所有前驱执行完成时的任务就绪时间;而最迟开始时间为保证所有后继均能在限定时间内执行完成的任务最迟开始时间.首先,通过前向宽度优先方法对各任务的最早开始时间(earlest start time,简称 EST)和最早完成时间(earlest finish time,简称 EFT)进行求解:

$$\begin{cases} EST(i)=0, & i=1 \\ EST(i)=\max_{j\in pre(i)} EFT(j), & \text{else} \\ EFT(i)=EST(i)+\tau(i) \end{cases} \tag{6}$$

其中,$\tau(i)$为 i 的最小执行时间:$\tau(i)=\min_{0\le k<l(i)}\tau_{ik}$;$EST(i)$为 i 的最早开始时间,即为该调度下从开始任务 1 到 i 的最长路径长度(不含 i);$EFT(i)$则为 i 的最早完成时间.易知工作流最小长度为结束任务 n 的最早完成时间:$mp=EFT(n)$.在保证结束任务能在 mp 时刻执行完成的条件下,各任务的最迟开始和最迟完成时间(latest start/finish time,简称 LST/LFT)通过逆向宽度优先方法进行求解:

$$\begin{cases} LFT(i)=mp, & i=n \\ LFT(i)=\min_{j\in succ(i)} LST(j), & \text{else} \\ LST(i)=LFT(i)-\tau(i) \end{cases} \tag{7}$$

其中:$LFT(i)$为 i 的最迟完成时间,当 i 在该时刻之前或该时刻执行完成时,工作流长度不超过 mp;$LST(i)$则为 i 的最迟开始时间,易知该调度下从 i 到结束任务 n 的路径长度为 $mp-LST(i)$(含 i).因此,i 的时间灵活度为

$$TM(i)=mp-(EST(i)+mp-LST(i))=LST(i)-EST(i) \tag{8}$$

图 1 为一个简单的工作流实例,表 1 为各任务对应的候选服务列表.各任务的最小执行时间 $\tau(i)$(右侧值)与时间灵活度 $TM(i)$(括号内的值)如图 1 所示,而关键路径 CP={1,3,7,9,10}.

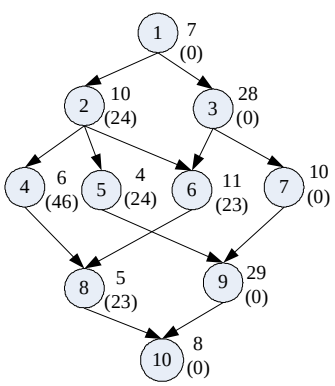

Fig.1 A simple workflow instance

图 1 一个简单的工作流实例

定义 6(正向临界任务/正向临界列表). 如果 $TM(i)>0$,且工作流中存在从关键任务到 i 的边,则称 i 为正向临界任务;所有正向临界任务组成的列表定义为正向临界列表,记为 φ_F.

定义 7(逆向临界任务/逆向临界列表). 如果 $TM(i)>0$,且工作流中存在从 i 到关键任务的边,则称 i 为逆向临界任务;所有逆向临界任务组成的列表定义为逆向临界列表,记为 φ_B.

Table 1 Services of tasks in Fig.1

表 1 图 1 中各任务的服务列表

i	$S(i)$
1	{(7,25),(25,7)}
2	{(10,55),(24,50.6),(41,33.6)(55,10)}
3	{(28,51),(51,28)}
4	{(6,26),(8,25.8),(18,18.8),(26,6)}
5	{(4,23),(6,22.8),(12,19.6),(20,9.5),(23,4)}
6	{(11,40),(25,33.2),(29,28.8),(40,11)}
7	{(10,45),(16,44),(45,10)}
8	{(5,22),(16,14.9),(19,10.5),(22,5)}
9	{(29,45),(45,29)}
10	{(8,30),(11,29.6),(15,27.8),(27,13.6),(30,8)}

定义 8(最短可达路径). 对于具有可达路径的两任务集φ与φ',从φ中任务开始到φ'中任务结束的最短路径定义为φ与φ'间的最短可达路径(此时,路径长度为该路径的任务个数),记为 $p(\varphi,\varphi')$.

由于工作流中只存在一个开始任务 1 和一个结束任务 n,且 1 与 n 均为关键任务,由定义 6 与定义 7 可知,从φ_F中的某一任务开始,一定存在至少一条路径到达φ_B中的某个任务.通过反证法可以证明:假设工作流中存在一条从开始任务 1 到结束任务 n 的路径 p,p 通过φ_F中的某一任务 i,但不经过φ_B中的任意任务,由定义 7 可知,路径 p 无法到达结束任务 n,问题得证;该结论也说明了φ_F与φ_B间一定存在可达路径.此外,φ_F与φ_B间的最短可达路径 $p(\varphi_F,\varphi_B)$中所有任务的时间灵活度一定大于 0.

同样,通过反证法可以证明:假设 $p(\varphi_F,\varphi_B)=\{i,\ldots,k-1,k,k+1,\ldots,j\}$,$i\in\varphi_F$,$j\in\varphi_B$,其中,$TM(k)=0$,则 $k-1\in\varphi_B$,因此,从φ_F到φ_B的最短路径应为 $p(\varphi_F,\varphi_B)=\{i,\ldots,k-1\}$,与假设 $p(\varphi_F,\varphi_B)=\{i,\ldots,k-1,k,k+1,\ldots,j\}$矛盾,问题得证.

图 1 中,$\varphi_F=\{2,6\}$,$\varphi_B=\{5,8\}$,而φ_F与φ_B间的最短可达路径 $p(\varphi_F,\varphi_B)$为{2,5}和{6,8}.

2.2 路径平衡算法描述

TM 反映了任务可用的最大松弛时间,当任务 i 的执行时间增长不超过其最大松弛时间时,工作流长度不发生改变.然而,松弛时间由同一路径上的多个任务共享,当某一任务的执行时间增大时,与该任务具有依赖关系(包括直接依赖和间接依赖)的所有任务的松弛时间便减少.如果将松弛时间在各任务间进行合理分配并最大限度地达到各条路径的长度平衡时,具有相同后继的多个任务便能够同步完成,从而减少了后继任务由于等待某一前驱执行完成而造成的其他前驱任务的费用优化空间的浪费,使得工作流的费用优化空间得到充分利用.以下分两种情况为工作流中的任务分配时间增量,即对任务的执行时间进行调整:

(a) 如果$|p(\varphi_F,\varphi_B)|=1$($|p(\varphi_F,\varphi_B)|$为路径 $p(\varphi_F,\varphi_B)$的任务个数),即工作流中存在一个或多个任务 i 满足条件 $i\in\varphi_F$ && $i\in\varphi_B$ 时,此时,i 的松弛时间由该任务所有,将 i 的执行时间 $\tau(i)$进行如下调整:

$$\tau(i)\leftarrow TM(i)+\tau(i) \tag{9}$$

(b) 如果$|p(\varphi_F,\varphi_B)|>1$,则工作流中任意任务的松弛时间均由多个具有依赖关系的任务共享,这种情况下的时间分配比较复杂,在时间调整过程中遵从以下两条原则:第一,优先调整最短可达路径中的任务;第二,优先调整长路径中的任务.其原因为:最短可达路径中的任务数较少,该路径的松弛时间由少数任务共享,受其他任务的影响相对较少;同时,长路径中任务的松弛时间较小,优先对其进行调整能够较快使得该路径的路径长度与工作流长度相等.基于以上原则,在对任务进行时间调整时,选择φ_F与φ_B之间的最短可达路径优先调整,当工作流中存在多条最短可达路径时,选择路径长度较长(即 TM 值较少)的路径优先调整.以下分几个步骤对$|p(\varphi_F,\varphi_B)|>1$时的任务执行时间进行调整:

步骤 1. 在φ_F中选择 TM 最小的一个或多个任务,记为 φ_F^*;

步骤 2. 通过宽度优先搜索方法查找 φ_F^* 与φ_B间的最短可达路径,当最短可达路径不止 1 条时,选择结束任务的 TM 值最小的一条路径,记为 $p(\varphi_F^*,\varphi_B^*)$.详细步骤见最短可达路径算法 SRP(shortest reachable path,简称 SRP);

步骤 3. 在 $p(\varphi_F^*,\varphi_B^*)$ 中查找时间灵活度最大的一条子路径,记为 p^*;当 $p(\varphi_F^*,\varphi_B^*)$ 中所有任务的时间灵活度均相等时, $p^*=p(\varphi_F^*,\varphi_B^*)$.

假定 $p(\varphi_F^*,\varphi_B^*)$ 中任务的最大时间灵活度为 TM^*,为 p^*中任务的执行时间进行如下调整:

$$\tau(i)\leftarrow\tau(i)+\frac{TM^*-\max\{TM(pre(p^*)),TM(succ(p^*))\}}{|p^*|} \tag{10}$$

其中,$pre(p^*)$为 p^*在路径 $p(\varphi_F^*,\varphi_B^*)$ 中的前一任务,如果 $pre(p^*)=null$,则 $TM(pre(p^*))=0$.同理,$succ(p^*)$为 p^*在路径 $p(\varphi_F^*,\varphi_B^*)$ 中的后一任务,如果 $succ(p^*)=null$,$TM(succ(p^*))=0$.

算法 1. SRP.

1) 初始化:min*Size*=∞;

2) FOR $\forall i\in\varphi_F^*$.

2.1) 初始化:$L\leftarrow i$, $p(i)=i$, *tmpSize*=∞;

2.2) WHILE (true)

2.2.1) 从 L 中选择第 1 个未被遍历的任务 j;

2.2.2) IF $p(j)>tmpSize$, break;

2.2.3) FOR $k\in succ(j)$ && $k\notin L$

2.2.3.1) $p(k)\leftarrow p(j)+k$, $L\leftarrow L+k$;

2.2.3.2) IF $k\in\varphi_B$ && $\{(|p(k)|<tmpSize)\|(|p(k)|=tmpSize$ && $TM(k)<TM(p(i,\varphi_B^*).last))\}$

$tmpSize=|p(k)|$;

$p(i,\varphi_B^*)\leftarrow p(k)$;

2.3) IF $\{|p(i,\varphi_B^*)|<\min Size\}\|\{|p(i,\varphi_B^*)|=\min Size$ && $TM(p(i,\varphi_B^*).last)<TM(p(\varphi_F^*,\varphi_B^*).last)\}$

$\min Size=|p(i,\varphi_B^*)|$;

$p(\varphi_F^*,\varphi_B^*)\leftarrow p(i,\varphi_B^*)$;

3) 返回路径 $p(\varphi_F^*,\varphi_B^*)$.

算法 1 中, $p(i,\varphi_B^*).last$ 表示路径 $p(i,\varphi_B^*)$ 的最后一个作业;同理, $p(\varphi_F^*,\varphi_B^*).last$ 表示路径 $p(\varphi_F^*,\varphi_B^*)$ 中的最后一个作业.SRP 需对 φ_F^* 中的任务进行遍历,为 φ_F^* 中的所有任务查找到达φ_B 的最短路径(步骤 2)的 FOR 结构);在为 φ_F^* 中的某个任务 i 查找最短路径时,需对该任务的后继(包括部分间接后继)进行遍历(步骤 2.2)的 WHILE 结构),时间复杂度为 $O(n)$;因此,SRP 总的时间复杂度为 $O(n^2)$.

以上分两种情况为工作流中的某些任务分配了时间增量,当这些任务的执行时间调整后,部分与其具有依赖关系的任务时间灵活度也将随之发生改变.因此需对工作流重新进行遍历,并更新各任务的时间灵活度及工作流中的φ_F及φ_B.路径平衡算法 PB 对以上过程进行循环,直到工作流中所有任务的时间灵活度均为 0,即所有路径的长度均相等为止.PB 的具体描述如下:

算法 2. PB.

1) 初始化任务执行时间:$\tau(i)=\min_{0\le k<l(i)}\tau_{ik},\forall i\in V$;

2) 循环条件检查:

2.1) 根据公式(6)~公式(8)计算 $TM(i),\forall i\in V$;

2.2) $U=\{i|TM(i)=0,\forall i\in V\}$;

2.3) $\bar{U}=V\setminus U$;

2.4) IF $\bar{U}=\varnothing$,算法结束;

3) 遍历工作流,得到φ_F和φ_B:

3.1) $\varphi_F=\{j|(i,j)\in E,i\in U,j\in\bar{U}\}$;

3.2) $\varphi_B=\{j\mid(j,i)\in E,i\in U,j\in\bar{U}\}$;

4) 对φ_F与φ_B中的任务进行检查:

4.1) IF $\exists i(i\in\varphi_F)$ && $(i\in\varphi_B)$

FOR $\forall i(i\in\varphi_F)$ && $(i\in\varphi_B)$: $\tau(i)\leftarrow TM(i)+\tau(i)$;

4.2) ELSE

4.2.1) 在φ_F中查找 *TM* 最小的任务列表 φ_F^*;

4.2.2) 调用 SRP 得到 φ_F^* 与φ_B间的最短可达路径 $p(\varphi_F^*,\varphi_B^*)$;

4.2.3) 在 $p(\varphi_F^*,\varphi_B^*)$ 中查找时间灵活度最大的路径 p^*,并按公式(9)为 p^*中的任务分配时间增量;

5) goto step 2);

PB 算法在保证工作流长度不发生变化的情况下,每次增大一个或多个任务的执行时间,最终使得通过工作流中任意任务的至少一条路径长度与工作流长度相等.其中,步骤 1)的时间复杂度为 $O(n)$;步骤 2)和步骤 3)的时间复杂度均为 $O(n^2)$;步骤 4)的时间复杂度同步骤 4.2.2)中 SRP 的时间复杂度为 $O(n^2)$;因此,每次循环(步骤 2)~步骤 4))的时间复杂度为 $O(n^2)$.循环次数与工作流中的非关键任务数相关,与任务数同数量级,因此最坏情况下,PB 算法的时间复杂度为 $O(n^3)$.

2.3 路径平衡算法的实例说明

在图 1 所示的工作流中,采用 PB 算法为各任务分配时间增量的计算结果如图 2 所示,其详细步骤如下:

1) $U=\{1,3,7,9,10\}$, $\bar{U}=\{2,4,5,6,8\}$, $\varphi_F=\{2,6\}$, $\varphi_B=\{5,8\}$, $\varphi_F^*=\{6\}$, $p(\varphi_F^*,\varphi_B^*)=\{6,8\}=p^*$, $TM^*=23$, $\tau(6)=11+23/2=22.5$, $\tau(8)=5+23/2=16.5$;

2) $U=\{1,3,7,9,10,6,8\}$, $\bar{U}=\{2,4,5\}$, $\varphi_F=\{2\}$, $\varphi_B=\{2,4,5\}$, $TM(2)=18$, $\tau(2)=10+18=28$;

3) $U=\{1,3,7,9,10,6,8,2\}$, $\bar{U}=\{4,5\}$, $\varphi_F=\varphi_B=\{4,5\}$, $TM(4)=16.5$, $TM(5)=6$, $\tau(4)=6+16.5=22.5$, $\tau(5)=4+6=10$;

4) $U=\{1,3,7,9,10,6,8,2,4,5\}$, $\bar{U}=\varnothing$, 算法结束.

由以上结果可知,PB 算法最大限度地使工作流中各条路径的长度相等.假定工作流中所有任务的执行时间均为调整后的执行时间,则具有同步完成特征的任务均能同步完成,工作流的费用优化空间也能得到充分利用.

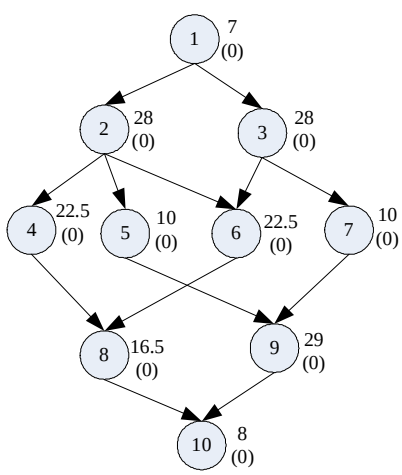

Fig.2 Path balance results of workflow in Fig.1

图 2 图 1 中工作流的路径平衡结果

2.4 基于路径平衡的初始时间分配

本节基于 PB 的计算结果为工作流中的任务设置初始约束时间,将任务 *i* 的最早完成时间(同最迟完成时间)

设为该任务的初始约束时间,记为$\delta_t(i)$.

$$\delta_t(i)=EFT(i) \tag{11}$$

$EFT(i)$由公式(6)可得,其中,$\tau(i)$为 PB 得到的计算结果.由于 PB 基于各任务的最小执行时间进行调整,因此工作流中各路径的长度不超过工作流的最小长度 mp.当约束时间$\delta \geq mp$ 时,各任务在初始约束时间内一定能找到至少一个满足条件的服务,解决了 DBL 等算法的适用性受限问题.

3 基于路径平衡的费用优化算法

基于PB算法为各任务设置了初始约束时间,当工作流总的约束时间δ等于工作流的最小长度mp时,工作流中各任务的约束时间即为初始约束时间,此时,关键路径上的任务均只能选择最快服务执行,无法进行费用优化,而基于PB的初始时间分配策略为非关键路径上的任务分配了时间增量,因此在一定程度上增大了非关键路径上任务的费用优化空间,非关键任务能在满足时间约束的条件下选择执行时间稍长但执行费用较低的服务执行;当$\delta<mp$ 时,关键路径上的任务全选择最快服务执行也无法满足工作流的时间约束目标,因此不存在任何有效的调度解;当$\delta>mp$ 时,工作流调度过程中存在冗余时间($\delta-mp$),将这些冗余时间分配到工作流各任务上会增大这些任务的费用优化空间,然而在进行冗余时间分配时,某一任务的费用优化空间的增加,将导致其他与其具有依赖关系(包括直接依赖与间接依赖)的任务的费用优化空间的减少,因此,冗余时间在工作流中各任务上的分配策略也将影响到工作流的费用优化效果,本节主要针对这种情况进行详细讨论.

3.1 基于逆向分层的冗余时间分配

DBL 算法中的逆向分层策略基于任务的逆向深度对任务进行分层,并将冗余时间在各层内进行平均分配.由于DBL利用了工作流的同步完成特征,达到了较好的费用优化效果.本文借鉴DBL中的分层策略对工作流进行分层,但与 DBL 不同的是,本文根据各层任务数按比例分配冗余时间,由于为任务数较多的层分配了较大的费用优化空间,因此能从整体上改善工作流的费用优化效果.基于逆向分层的冗余时间分配过程如下:

步骤 1. 通过逆向宽度优先方法计算任务 i 的逆向深度,记为 $BD(i)$.

$$BD(i)=\begin{cases}0, & i=n\\ \max_{j\in succ(i)} BD(j)+1, & \text{else}\end{cases} \tag{12}$$

易知:开始任务的逆向深度最大,记为 $BD(1)$;结束任务的逆向深度最小,为 0.

步骤 2. 基于逆向深度对工作流进行分层,其中,第 k 层任务 $BL(k)$包括所有逆向深度为 k 的任务.

$$BL(k)=\{i|BD(i)=k,\forall i\in V\} \tag{13}$$

步骤 3. 根据各层任务数分配冗余时间,并为第 k 层设置冗余约束时间,记为$\delta_\Delta(i)$.

$$\delta_\Delta(k)=\frac{\sum_{i=k}^{BD(1)}|BL(i)|}{\sum_{i-0}^{BD(1)}|BL(i)|}\times(\delta-mp) \tag{14}$$

其中,$|BL(i)|$为 $BL(i)$的任务个数,$\sum_{i=k}^{BD(1)}|BL(i)|$为从工作流开始任务到第 k 层任务的任务数之和,$\sum_{i=0}^{BD(1)}|BL(i)|$为工作流的任务总数.

步骤 4. 为各任务 i 设置冗余约束时间$\delta_\Delta(i)$.

$$\delta_\Delta(i)=\delta_\Delta(k),i\in BL(k) \tag{15}$$

3.2 基于路径平衡的费用优化算法描述

以上对各任务的初始约束时间和冗余约束时间进行了分配,当所有任务均在分配的子约束时间之前执行完成时,工作流便能在约束时间内执行完成.为了充分利用费用优化空间,将任务的开始时间设为任务就绪时间,即所有前驱执行完成后该任务便开始执行(本文忽略了算法运行时间和任务间的数据传输开销).本节设计基于路径平衡的费用优化(path balance based cost optimization,简称 PBCO)算法对工作流执行费用进行优化,PBCO 算法描述如下.

算法 3. PBCO.

输入:工作流及各任务的候选服务列表信息;

约束时间δ;

输出:工作流的执行费用 C.

1) 调用 PB 算法对工作流中各任务的执行时间进行调整;
2) 根据公式(11)为任务设置初始约束时间$\delta_t(i)$;
3) 根据第 3.1 节的方法为各任务设置冗余约束时间$\delta_\Delta(i)$;
4) 为工作流中的各任务 i 设置约束时间:$\delta(i)=\delta_t(i)+\delta_\Delta(i)$;
5) 初始化就绪任务列表 RL 与结束任务列表 FL:$RL=\{1\}$,$FL=\varnothing$,$C=0$;
6) WHILE $FL\neq V$
6.1) 在 RL 中选取一个任务 i;
6.2) 为 i 设置开始时间α_i:$\alpha_i=\max_{j\in pre(i)}\beta_j$;
6.3) 在 $S(i)$中选择满足条件$\tau_{ik}\leq\delta(i)-\alpha_i$且费用最小的服务 s_{ik};
6.4) 调度 s_{ik} 执行并更新 RL; //从 RL 中移除 i,并将 $succ(i)$中的就绪任务加入到 RL 中
6.5) $C=C+c_{ik}$;
6.6) $FL\leftarrow FL+i$;
7) 返回 C;

PBCO 中:步骤 1)的时间复杂度为 $O(n^3)$;而步骤 2)和步骤 3)的时间复杂度为 $O(n^2)$;步骤 4)为 $O(n)$;步骤 5)为常数时间;步骤 6)为 WHILE 循环,对工作流中所有任务进行调度,在对某一任务 i 进行调度的过程中,步骤 6.2)与 6.4)对 i 的前驱或后继列表进行遍历,时间复杂度为 $O(n)$;步骤 6.3)对服务列表进行遍历,时间复杂度为 $O(m)$,其中,m 为服务规模;而其他步骤的复杂度为常数时间;由于 $n>>m$,因此步骤 6)的时间复杂度为 $O(n^2)$.综上所述,PBCO 的时间复杂度同 PB 算法的时间复杂度,为 $O(n^3)$.

通过以上分析可知,截止期约束的工作流费用优化效果依赖于费用优化空间在各任务间的合理分配.由于某一任务费用优化空间的增加将减少其他与其具有依赖关系(包括间接依赖和直接依赖)的任务的费用优化空间,因此在分配工作流的费用优化空间时不仅需要考虑各任务的资源特征,工作流的结构特征也是必须考虑的重要因素之一.PB 算法考虑了工作流中各条路径的长度差异,为短路径上的任务逐步分配时间增量,尽量使得各条路径的长度相等;PBCO 基于 PB 的计算结果进行初始时间分配,为某一任务的所有前驱分配了相同的初始约束时间,利用了工作流的同步完成特征,在工作流的执行过程中减少了因等待某一前驱任务执行完成而造成的其他前驱任务费用优化空间的浪费;同时,由于逆向深度相同的任务通常具有同步完成特征,PBCO 在进行冗余时间分配时也利用了这一特征,根据任务的逆向深度进行分层并按层分配冗余时间,为同一层的任务分配相同的冗余时间,使得前驱列表中的任务均能同步完成.此外,基于任务数的冗余时间分配策略将冗余时间在各层间按任务数比例进行分配,为任务数较多的层分配了较多的费用优化空间,一定程度上增大了多数任务的费用优化空间,从整体上优化了工作流的费用优化效果.

3.3 基于路径平衡的费用优化算法实例说明

为进一步说明 PBCO 的优越性,本节以图 1 中的工作流实例为例,分别采用 PBCO,DET 及 DBL 算法对其进行调度.由图 1 可知,$mp=BL_{\min}=82$,其中,$BL_{\min}$ 为工作流最小分层完成时间[6].当$\delta=93$ 时,DBL 的适用条件满足,各算法为任务设置的约束时间和具体调度结果见表 2.由表可知,PBCO 对各任务的约束时间进行了较为合理的设置.而 DET 算法则存在明显的缺陷:DET 采用串归约方法对关键路径{1,3,7,9,10}进行费用优化,而任务{2,4,5,6,8}的约束时间设置则依赖于关键任务的完成时间.由于串归约方法只考虑了关键任务的资源特征,为结束任务 10 分配了较大的费用优化空间,从而使得{1,3,7,9}的费用优化空间相对减少,进而使得非关键任务{2,4,5,6,8}的费用优化空间也比较小.如 DET 为任务 8 设置的约束时间为 74,远小于 PBCO(为 83.9)与 DBL(为 83)中设置的约束时间.因此,DET 虽然对关键路径上的任务进行了最大限度的优化,但由于其他任务的费用优

化效果比较差,因此工作流的费用优化效果比较差.此外,DBL 为所有逆向深度相同的任务分配了相同的约束时间,如为任务{4,5,6,7}设置的约束时间均为 52.事实上,{4,6}和{5,7}并不具有同步完成特征,因此也造成时间分配的不合理.由实际调度结果(C_{PBCO}=300.4,C_{DET}=324.9,C_{DBL}=318.5)可知,PBCO 的算法性能优于 DET 和 DBL.

Table 2 Comparison of scheduling results in PBCO, DET and DBL

表 2 PBCO,DET 及 DBL 对工作流的调度结果比较

i	$\delta(i)$			s_{ik}		
	PBCO	DET	DBL	PBCO	DET	DBL
1	8.1	7	9	(7,25)	(7,25)	(7,25)
2	38.3	35	39	(24,50.6)	(24,50.6)	(24,50.6)
3	38.3	35	39	(28,51)	(28,51)	(28,51)
4	65.2	57.5	52	(26,6)	(26,6)	(18,18.8)
5	52.7	45	52	(22,9.5)	(12,19.6)	(22,9.5)
6	65.2	57.5	52	(29,28.8)	(11,40)	(11,40)
7	52.7	45	52	(16,44)	(10,45)	(16,44)
8	83.9	74	83	(19,10.5)	(16,14.9)	(22,5)
9	83.9	74	83	(29,45)	(29,45)	(29,45)
10	93	93	93	(8,30)	(15,27.8)	(11,29.6)

4 实验结果与性能分析

为了对 PBCO 的算法性能进行全面评估,本文在模拟实验环境中实现了另外 3 种具有代表性且较先进的启发式算法:DET,DBL 及 ATD(advanced time distribution),并在大量实例中对这 4 种算法进行测试和比较.分析可知:TD 算法基于任务权值在路径长度中所占的比例分配时间,但将任务的开始时间设为前驱任务的约束时间;ATD 在 TD 算法的基础上进行改进,将任务的开始时间设为前驱任务的完成时间而非约束时间,达到了充分利用费用优化空间提高工作流费用优化效果的目标.所有算法均采用 Java 编程,运行于内存为 2G、速度为 2.53GHz 的双核处理器上.

4.1 实验设计

实验针对大量工作流实例在不同调度空间中的调度性能进行比较.由于目前工作流调度中不存在一个标准测试集,本文设计并实现了一个工作流结构和工作流规模均可控的随机生成器,该生成器在文献[16,17]中得到了应用.在工作流的生成过程中,对工作流规模(workflow size,简称 WS)和工作流结构(即各任务的最大出入度,in out degree,简称 IOD)进行设置,本实验过程中设定 WS 以 100 为增量在[100~1000]中取值;同时设定 IOD={3,5,7,10},当 IOD=3 时,各任务的出入度取[1~3]之间的随机数.各任务的候选服务列表也随机生成,生成过程中,对服务列表的最大规模(service size,简称 SS)进行控制,设定 SS={10,20,30,40},当 SS=10 时,各任务的服务列表长度取[1~10]之间的随机数;各任务的执行时间取[5~60]之间的随机数,执行费用为执行时间的函数,实验过程中,分别对 3 类费用函数(cost function,简称 CF)进行实验,包括凸函数 concave(ccv)、凹函数 convex(cvx)及线性函数 Linearity(lin)函数,并保证执行时间较长的服务执行费用较低.此外,实验中对不同约束时间δ下的算法性能进行比较,当所有任务均选最快服务执行时,该调度下得到的工作流长度为d的下界,记为 d;而当所有任务均选最慢服务执行时,该调度下得到的工作流长度为δ的上界,记为 D.实验过程中,对δ进行如下设置:$\delta=d+\omega\times(D-d)$,$\omega$以 0.05 为增量在[0~1]间进行取值.实验总共进行的实验组数为 10×4×4×3×21=10080.

比较结果通过最优解比例(best proportion,简称 BP)、与最优解之间的平均偏差(average difference to best,简称 ADB,以下简称平均偏差)、平均执行费用(average cost,简称 AC)及 PBCO 相对于各算法的平均费用改进比(improvement ratio of average cost,简称 IRAC)进行衡量;此外,通过算法的平均运行时间(average runtime,简称 ART)比较算法效率.各性能度量指标设计为

$$BP=\frac{N_B(H)}{N} \tag{16}$$

$$ADB=\frac{\sum_{i=1}^{N}\frac{C_i(H)-C_i(B)}{C_i(B)}}{N} \tag{17}$$

$$AC=\frac{\sum_{i=1}^{N}C_i(H)}{N} \tag{18}$$

$$IRAC=\frac{AC(H)-AC(PBCO)}{AC(H)} \tag{19}$$

$$ART=\frac{\sum_{i=1}^{N}RT_i(H)}{N} \tag{20}$$

其中,H 代表不同的算法,$N_B(H)$为该组实验中 H 的目标函数值最小的实例个数,N 为总的实验组数,$C_i(H)$为第 i 个实例中 H 的目标函数值,$C_i(B)$为该组实例中目标函数的最小值,$AC(H)$为所有实例中 H 的平均执行费用;$RT(H)$为 H 的算法开销.

4.2 结果比较与分析

表 3 为所有问题参数下各种算法的比较情况.

Table 3 Comparison of all problem parameters obtained from four heuristics

表 3 所有问题参数下的算法性能比较

H	BP (%)	ADB (×100)	AC	IRAC (%)	ART (ms)
PBCO	**89.8**	**0.35**	**8 553**	**0**	**175.9**
DBL	10	48.34	12 921	34	3.3
DET	0. 2	56.59	13 104	35	70.6
ATD	0	94.29	15 981	46	5.46

从最优解比例和平均偏差来看:PBCO的计算结果明显优于其他3种算法,其最优解比例为89.8%,平均偏差则仅为 0.0035.这表明,PBCO 不仅具有绝对占优的最优解比例,而且算法稳定性非常高;DBL 与 DET 的最优解比例分别为 10%与 0.2%,而平均偏差则远大于 PBCO,分别为 0.48 与 0.57;ATD 在这 4 种算法中性能最差.从平均执行费用与PBCO相对于其他算法的平均费用改进比来看,PBCO的平均执行费用最小,相对于另外3种算法改进了 34%~46%不等.此外,从算法开销来看,PBCO 的平均开销为 176ms,虽然高于其他算法的平均开销(DET 为 70.6ms,而 DBL 与 ATD 仅为 3.3ms 和 5.46ms),但算法开销仍比较合理.

4.3 不同问题参数下的算法性能比较

表 3 表明了 PBCO 算法相对于 DET,DBL 及 ATD 的优越性,为了对算法性能进行全面比较和评估,本节在不同问题参数下(包括约束时间、工作流规模、工作流结构、服务规模和费用函数)对各算法进行比较.

4.3.1 不同约束时间下的平均执行费用比较

图 3 为各种算法在不同约束时间下得到的平均执行费用曲线图.

PBCO 算法在不同约束时间下的性能都比较理想,尤其当约束时间较小时,PBCO 的平均执行费用远低于其他 3 种算法.这是由于当约束时间较小时,关键路径上任务的费用优化空间也比较小,工作流的费用优化效果在较大程度上依赖于非关键路径上任务的费用优化情况;PBCO 采用路径平衡方法为非关键路径上的任务分配时间增量,最大限度地达到了各条路径的长度平衡,同时,基于路径平衡结果进行初始时间分配,为所有具有相同后继的任务分配相同的约束时间,充分利用了工作流中各任务的费用优化空间,因此算法性能远优于另外 3 种算法.DBL 在约束时间较小时(ω<0.25)性能比较差,这是因为当约束时间较小时,DBL 算法的适用性受限,此时采用费用优化效果较差的 MCP 算法进行调度,因此平均执行费用高于其他 3 种算法;但随着约束时间的增长,当 ω>0.25 时,DBL 的适用性逐渐得到满足,其平均执行费用迅速降低,且逐渐与 PBCO 接近;当ω接近于 1 时,DBL 的算法性能超过了 PBCO.这是由于 PBCO 基于各层任务数对冗余时间进行分配,为任务数较多的层分配较大的费用优化空间,当约束时间接近于 D 时,PBCO 可能为任务数较多的层分配了过多的费用优化空间,即任务的费用优化空间可能大于该任务的最大执行时间,从而造成了费用优化空间的部分浪费;同时,PBCO 可能为其他

任务数较少的层,尤其是逆向深度较大的层分配了较少的费用空间,造成这些层的任务无法充分地进行费用优化,因此,与将冗余时间在各层进行平均分配的 DBL 算法相比,在ω接近于 1 时,PBCO 算法性能稍差.DET 算法在各约束时间下的算法性能都比较稳定,但平均费用比 PBCO 高 35%以上.这说明 DET 在各种约束时间下算法性能都不太理想,其原因是:DET 采用串归约方法对关键路径上的任务进行费用优化,并基于关键任务的调度结果为非关键路径上的任务进行约束时间设置;虽然串归约方法使得关键路径上的任务得到了最大限度的费用优化,但是该过程中只考虑了关键任务的资源特征,忽略了工作流的结构特征和其他非关键任务的资源特征,而非关键任务的约束时间分配极大程度地依赖于关键任务的调度结果,因此可能造成非关键路径上约束时间分配的不合理,从而导致工作流的费用优化效果比较差.此外,ATD 算法基于任务长度在路径中所占的比例进行时间分配,在 4 种算法中性能最差.

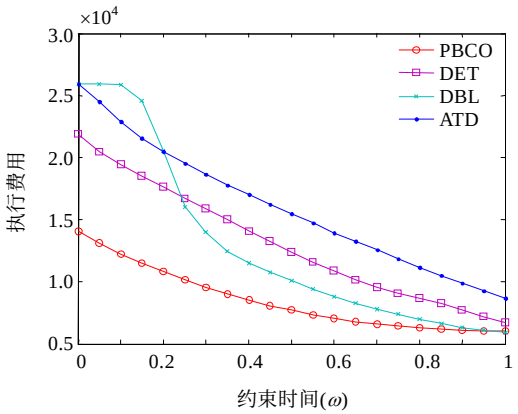

Fig.3 Comparison of average costs for each heuristic as a function of deadlines

图 3 各算法的平均费用随约束时间变化的曲线图

4.3.2 不同工作流规模下的平均执行费用比较

图 4 与图 5 分别为不同工作流规模下各算法的平均执行费用和平均算法开销曲线图.

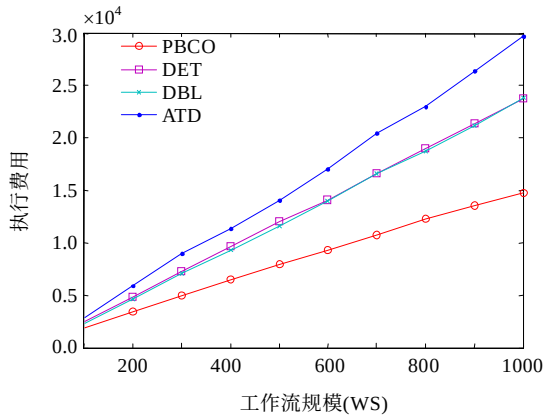

Fig.4 Comparison of average costs for each heuristic as a function of workflow sizes

图 4 各算法的平均执行费用随工作流规模变化的曲线图

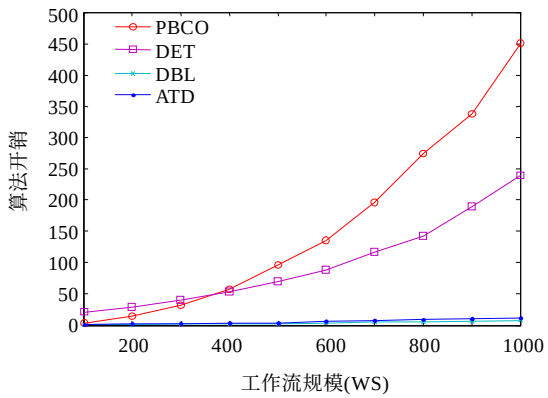

Fig.5 Comparison of average runtime for each heuristic as a function of workflow sizes

图 5 各算法的平均开销随工作流规模变化的曲线图

从各种算法的平均执行费用来看(如图 4 所示),工作流规模越大,PBCO 相对于其他算法的优越性越明显,这表明 PBCO 更加适用于规模较大的工作流调度,其原因为:工作流规模越大,结构越复杂,工作流中处于非关键路径上的任务数越多;PBCO 算法的优势在于,采用 PB 算法为短路径上的任务分配了时间增量,最大限度地达到了多条路径的长度平衡,并基于路径平衡结果进行时间分配,最终达到合理分配时间并充分利用费用优化空间的目标.因此,非关键任务数越多,PBCO 的优越性便越明显;同时,PBCO 基于各层任务数进行冗余时间分配,当工作流规模越大时,这种冗余时间分配策略的优越性也更明显.DBL 与 DET 算法在各种工作流规模下算法性能基本持平,这是由于约束时间较小时,DBL 在算法适用性上存在缺陷,但当适用条件得到满足后,其平均性能稍优于 DET,因此综合各约束时间后,DBL 和 DET 的算法性能基本持平.此外,从算法开销来看(如图 5 所示),各算法的平均开销随工作流规模的增大均有所增加,尤其 PBCO 的算法开销增加最明显,这是因为 PB 算法需对工作流进行多次遍历以达到工作流中各条路径的长度平衡,工作流规模越大,PB 算法需要遍历的次数越多,因此算法开销增长比较明显.DET 的算法开销在工作流规模较小时(<400 时)比 PBCO 大,但随着工作流规模的增大,DET 的算法开销也有所增长,但其增长幅度较 PBCO 小;而 ATD 与 DBL 的算法开销在各种工作流规模下均比较小,这表明 ATD 与 DBL 的算法效率比较高.

4.3.3 不同工作流结构下的算法性能比较

表 4 为不同工作流结构下的算法性能比较结果.表中各项数据表明,随着工作流结构参数 *IOD* 的增长,PBCO 的算法性能有所提高,最优解比例从 80.9%提高至 94.7%,而平均偏差则从 0.85 降至 0.1.这是由于 *IOD* 的增长使得工作流中非关键任务的比例有所增加,而 PBCO 对非关键任务的费用优化空间进行了合理分配,最大化利用了工作流的费用优化空间,减少了工作流执行过程中“时间碎片”的产生,因此,PBCO 的优越性随 *IOD* 的增长有所提高.同时,随着 PBCO 算法性能的提高,另外 3 种算法的性能相应有所降低,如 DBL 的最优解比例从 18.8%降至 4.9%,而平均偏差从 13.36 增加至 79.38;DET 的平均偏差则从 31.38 增长至 79.83.从平均执行费用来看,当 *IOD* 从 3 增长至 10 时,PBCO 相对于其他算法的费用改进比从 15%~26%增长至 47%~60%.由此可见,工作流结构参数 *IOD* 越大,PBCO 的算法性能相对越高.

Table 4 Comparison of four heuristics at different workflow structures

表 4 不同工作流结构下的算法性能比较

IOD	BP (%)				ADB (×100)				AC				IRAC (%)			
	3	5	7	10	3	5	7	10	3	5	7	10	3	5	7	10
PBCO	**80.9**	**89.2**	**94.5**	**94.7**	**0.85**	**0.35**	**0.11**	**0.1**	**11 520**	**9 254**	**6 860**	**6 578**	**0**	**0**	**0**	**0**
DET	0.3	0	0	0.4	31.38	41.21	73.96	79.83	14 442	13 095	12 486	12 393	20	29	45	47
DBL	18.8	10.8	5.5	4.9	13.36	28.68	71.92	79.38	13 548	12 721	12 745	12 670	15	27	46	48
ATD	0	0	0	0	40.62	77.58	120.88	138.08	15 484	16 297	15 772	16 372	26	43	57	60

4.3.4 不同服务规模下的算法性能比较

表 5 是不同服务规模下各算法的性能比较结果.从最优解比例和平均偏差来看,PBCO 的算法性能随服务规模的增长稍有降低,其最优解比例从 91.5%降至 88%,而平均偏差从 0.001 9 增加至 0.004 9;相应地,DBL 的最优解比例随服务规模的增长从 8.4%增长至 11.8%,但平均偏差的改进则不太明显.从平均执行费用与 PBCO 相对于其他算法的平均费用改进比来看,随着工作流规模的增长,各算法的平均执行费用均有不同程度的降低,这是由于服务规模的增长使得各任务的可选服务增加,各候选服务间执行时间的差距减少,在工作流执行过程中产生的“时间碎片”减小,从而进一步利用了费用空间并提高了工作流的费用优化效果.此外,从算法开销来看,PBCO, DBL 及 ATD 的算法开销随服务规模的增长变化不大,而 DET 的算法开销随服务规模的增长迅速增长,当 *SS*=10 时,DET 的平均开销为 68ms;而当 *SS*=40 时,其平均开销增长至 140ms,接近于同等条件下 PBCO 的算法开销(178ms),这是由于 DET 算法采用串归约方法对关键路径上的任务进行费用优化,该方法的时间复杂性不仅依赖于关键路径上进行串归约的任务数,同时也依赖于各任务的可选服务数,当时间窗口内满足条件的服务数增多时,完成一次迭代所需的时间开销也越大,因此当服务规模增长时,DET 的算法开销也随之增长.

Table 5 Comparison of four heuristics at different service sizes

表 5 不同服务规模下的算法性能比较

SS	BP (%)				ADB (×100)				AC				IRAC (%)				ART (ms)			
	10	20	30	40	10	20	30	40	10	20	30	40	10	20	30	40	10	20	30	40
PBCO	**91.5**	**90.6**	**89.2**	**88**	**0.19**	**0.32**	**0.4**	**0.49**	**10 077**	**8 395**	**7 944**	**7 796**	**0**	**0**	**0**	**0**	**178.68**	**173**	**172.67**	**178.66**
DET	0.2	0.3	0.1	0.2	40.91	58.28	63.5	63.68	14 104	13 139	12 686	12 486	29	36	37	38	68.1	98.6	117.85	139.95
DBL	8.4	09.1	10.7	11.8	43.15	51.53	53.72	44.93	14 809	12 956	12 392	11 526	32	35	36	32	3.35	3.1	3.22	3.53
ATD	0	0	0	0	68.06	95.32	113.13	100.65	16 925	15 984	16 081	14 934	40	47	51	48	5.4	5.18	5.43	5.81

4.3.5 不同费用函数下的算法性能比较

表6为不同费用函数下的算法性能比较结果.从最优解比例来看,PBCO在凹函数下的算法性能稍劣于凸函数和线性函数下的算法性能,而 DBL 在凹函数中的最优解比例则相应比较高.从平均偏差来看,各算法的平均偏差在凹函数中的值都比其他函数中的值小,同时,各算法在凹函数中的平均费用也比其他函数中的平均费用小.这是由于凹函数中各任务的平均执行费用小于凸函数和线性函数中各任务的平均执行费用,因此各算法所得到结果之间的差异减少,这表现在 IRAC 上,在凹函数中 PBCO 的 IRAC 值也小于其他函数中的 IRAC 值.

Table 6 Comparison of four heuristics using different cost functions

表 6 不同费用函数下的算法性能比较

CF	BP (%)			ADB (×100)			AC			IRAC (%)		
	ccv	cvx	lin	con	cov	lin	ccv	cvx	lin	ccv	cvx	lin
PBCO	**91.7**	**86.5**	**91.2**	**0.43**	**0.26**	**0.37**	**9 584**	**7 349**	**8 725**	**0**	**0**	**0**
DET	0.3	0.1	0.1	65.57	41.37	62.85	15 020	10 405	13 887	36	29	37
DBL	7.9	13.5	8.6	52.67	44.03	48.3	14 340	11 302	13 121	33	35	34
ATD	0	0	0	110.82	73.46	98.59	18 547	12 908	16 489	48	43	47

5 结束语

针对截止期约束的工作流费用优化问题,本文提出 PB 算法对工作流结构进行调整,在此基础上,提出基于路径平衡的费用优化算法 PBCO.PB 算法在保证工作流最小长度不发生变化的情况下,逐步为工作流短路径中的任务分配时间增量,最大限度地达到各条路径的长度平衡并基于路径平衡结果进行初始时间分配,使得各任务在初始时间内均能找到至少一个满足条件的服务,解决了 DBL 等算法的适用性受限问题.同时,基于 PB 的计算结果进行时间分配时,为所有具有相同后继的任务分配了相同的约束时间,减少了该后继在执行过程中由于等待某一前驱执行完成而造成的其他前驱的费用优化空间的浪费,达到了充分利用费用优化空间并提高费用优化效果的目标;此外,PBCO 基于任务的逆向深度对工作流进行分层,并根据各层任务数按比例分配冗余时间,由于为任务数较多的层分配了较多的费用优化空间,整体上提高了工作流的费用优化效果.大量实验结果证明了 PBCO 算法的优越性.

致谢 在此向对本文工作给予支持和建议的同行,尤其是国防科学技术大学高性能应用研究中心的老师们表示感谢;同时,对美国威斯康辛大学的刘晓阳博士表示感谢,本文的英文摘要得到了刘晓阳博士的多次修改.

References:

[1] Buyya R, Abramson D, Venugeopal S. The grid economy. Proc. of the IEEE, 2005,93(3):698−714. [doi: 10.1109/JPROC.2004.842784]

[2] Prodan R, Wieczorek M. Bi-Criteria scheduling of scientific grid workflows. IEEE Trans. on Automation Science and Engineering, 2010,7(2):364−376. [doi: 10.1109/TASE.2009.2014643]

[3] Talukder AKMKA, Kirley M, Buyya R. Multiobjective differential evolution for scheduling workflow applications on global grids. Concurrency and Cmputation: Practice and Experience, 2009,21(13):1742−1756. [doi: 10.1002/cpe.1417]

[4] Hazir Ö, Haousri M, Erel E. Discrete time/cost trade-off problem: A decomposition-based solution algorithm for the budget version. Computers & Operations Research, 2010,37(4):649−655. [doi: 10.1016/j.cor.2009.06.009]

[5] Wang Y, Hu CM, Du ZX. QoS-Awared grid workflow scheduling. Ruan Jian Xue Bao/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/jos 172341]

[6] Yuan YC, Li XP, Wang Q, Zhang Y. Bottom level based heuristic for workflow scheduling in grids. Chinese Journal of Computers, 2008,31(2):283−290 (in Chinese with English abstract).
[7] Yuan YC, Li XP, Wang Q. Cost optimization heuristics for grid workflows scheduling based on serial reduction. Journal of Computer Research and Development, 2008,45(2):246−253 (in Chinese with English abstract).
[8] Yuan YC, Li XP, Wang Q, Zhu X. Deadline division-based heuristic for cost optimization in workflow scheduling. Information Sciences, 2009,179:2562−2575. [doi: 10.1016/j.ins.2009.01.035]
[9] Yuan YC, Li XP, Wang Q, Zhang XD. Grid workflows schedule based on priority rules. Acta Electronica Sinica, 2009,37(7): 1457−1464 (in Chinese with English abstract).
[10] Yu J, Buyya R, Tham CK. Cost-Based scheduling of scientific workflow applications on utility grids. In: Proc. of the 1st Int'l Conf. on e-Science and Grid Computing (e-Science 2005). Melbourne, 2005. 140−147.
[11] Yu J, Buyya R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Science Programming, 2006,14(3-4):217−230.
[12] Menasce DA, Casalicchio E. A framework for resource allocation in grid computing. In: Proc. of the IEEE Computer Society's 12th Annual Int'l Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems. Washington, 2004. 259−267. [doi: 10.1109/MASCOT.2004.1348280]
[13] Ke H, Ma WM, Ni YD. Optimization models and a GA-based algorithm for stochastic time-cost trade-off problem. Applied Mathematics and Computation, 2009,215(1):308−313. [doi: 10.1016/j.amc.2009.05.004]
[14] Zhang XD, Li XP, Wang Q, Yuan YC. Hybrid particle swarm optimization algorithm for cost minimization in service-workflows with due dates. Journal on Communications, 2008,29(8):87−94 (in Chinese with English abstract).
[15] Chen WN, Zhang J. An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. IEEE Trans. on Systems, Man and Cybernetics-Part C: Applications and Reviews, 2009,39(1):29−43. [doi: 10.1109/TSMCC.2008.2001722]
[16] Liu CC, Zhang WM, Luo ZG, Ren KJ. Workflow cost optimization heuristics based on an advanced priority rule. Journal of Computer Research and Development, 2012,49(7):1593−1600.
[17] Liu CC, Zhang WM, Luo ZG, Ren KJ. Temporal consistency based heuristics for cost optimization in workflow scheduling. Journal of Computer Research and Development, 2012,49(6):1323−1331.

附中文参考文献:

[5] 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]
[6] 苑迎春,李小平,王茜,张毅.基于逆向分层的网格工作流调度算法.计算机学报,2008,31(2):282−290.
[7] 苑迎春,李小平,王茜.基于串归约的网格工作流费用优化方法.计算机研究与发展,2008,45(2):246−253.
[9] 苑迎春,李小平,王茜,张晓东.基于优先级规则的网格工作流调度.电子学报,2009,37(7):1457−1464.
[14] 张晓东,李小平,王茜,苑迎春.服务工作流的混合粒子群调度算法.通信学报,2008,29(8):87−94.
[16] 刘灿灿,张卫民,骆志刚,任开军.基于改进优先级规则的工作流费用优化方法.计算机研究与发展,2012,49(7):1593−1600.
[17] 刘灿灿,张卫民,骆志刚,任开军.基于时序一致的工作流费用优化方法.计算机研究与发展,2012,49(6):1323−1331.

刘灿灿(1980－),女,湖南涟源人,博士,工程师,主要研究领域为科学工作流,高性能计算,网格计算,并行与分布算法.

E-mail: cancanliu@nudt.edu.cn

骆志刚(1962－),男,博士,教授,博士生导师,主要研究领域为并行计算,生物信息学.

E-mail: zgluo@nudt.edu.cn

张卫民(1966－),男,博士,教授,博士生导师,主要研究领域为并行与分布式计算,服务计算.

E-mail: Wmzhang104@163.net