基于逆向分层的工作流时间 - 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. 国防科技大学 计算机学院，湖南 长沙 410073）

摘　要： 针对效用网格下基于优先级因子的工作流时间 - 费用优化问题，基于工作流的同步完成特征对任务进行分层并提出三种实时调度算法：基于逆向分层的 sufferage（BLSuff）、基于逆向分层的 min-min（BLMin）及基于逆向分层的 min-max（BLMax）。算法设计基于优先级因子的衡量标准对时间与费用同时进行优化，并为任务设置期望完成时间以达到充分利用费用优化空间进行费用优化的目标。实验结果表明这三种算法在各种优先级因子下都能对工作流的执行时间与执行费用进行较好的优化。

关键词： 工作流调度；时间 - 费用优化；优先级因子；逆向分层
中图分类号： TP 393　　**文献标志码：** A　　**文章编号：** 1001 - 2486(2013)03 - 0061 - 06

Time and cost trade-off heuristics for workflow scheduling based on bottom level

LIU Cancan[1], *ZHANG Weimin*[2], *LUO Zhigang*[2]
(1. Air Force Training Equipment Institute, Beijing 100195, China;
2. College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: In order to manage the trade-off between the workflow execution time and the workflow execution cost on utility grids, the task was divided into several groups by using the workflow synchronization properties, and three real-time heuristics were proposed: the bottom level based sufferage (BLSuff), the bottom level based min-min (BLMin) and the bottom level based min-max (BLMax). A metric was designed in these heuristics to optimize the execution time and the execution cost simultaneously based on the trade-off factor, and the expected finish time was set for the task trying to make most of the cost optimization space to save the workflow execution cost. The experimental results demonstrate that these heuristics can optimize the execution time and execution cost simultaneously with various trade-off factors.

Key words: workflow scheduling; time cost trade-off; trade-off factor; bottom level

工作流作为一种组织和管理大规模科学和工程计算的有效工具在生产自动化和科研领域中发挥着越来越重要的推动作用，在效用网格中进行工作流调度时通常需要考虑其执行时间、执行费用及可靠性等多个目标，不同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已被公认为是一个 NP 难问题[1-2]。针对工作流的时间 - 费用优化问题，大多研究均忽略了任务间的资源竞争，基于候选服务在任意时刻可用这一假设将该问题抽象成项目管理中的离散时间 - 费用平衡问题（DTCTP，Discrete Time/Cost Trade-off Problem，）[3]，同时对其中一方进行约束而对另一方进行优化[4-6]。然而，这类方法无法对执行时间与执行费用同时进行优化；同时，算法在调度之前需对一方进行预测，在资源状态动态变化且各资源收费标准可能不明确的网格环境中，对执行时间或执行费用进行预测通常比较困难。与以上方法不同的是，本文在存在资源竞争的动态网格环境下对工作流进行实时调度，基于优先级因子对工作流的执行时间与执行费用同时进行优化（其中优先级因子为不同目标间的权重）。文献[7-9]对这类问题进行了研究，在深入研究已有工作的基础上，本文分析工作流的同步完成特征，基于逆向分层策略对任务进行分层调度，并在 sufferage、min-min 及 min-max 策略[10]的基础上提出三种实时优化算法：基于逆向分层的 sufferage（bottom level based sufferage，BLSuff）、基于逆向分层的 min-min（bottom level based min-min，BLMin）以及基于逆向分层的 min-max（bottom level based min-max，BLMax）。实验结果表明了三种分层优化算法的优越性。

* 收稿日期：2012 - 09 - 15
基金项目：国家 863 计划资助项目（2006AA01A123）；国家自然科学基金资助项目（60903042）
作者简介：刘灿灿（1980—），女，湖南涟源人，工程师，博士，E-mail：cancanliu@nudt.edu.cn

1 工作流时间 – 费用优化问题描述

工作流(W) 通常通过有向无环图(Directed Acrylic Graph, DAG) 进行描述: DAG = { V, E} , 其中 $V=\{1,2,\cdots,n\}$ 表示 W 中的任务集,而 $E=\{(i,j)\mid i,j\in V\}$ 表示任务间的依赖关系。W 中的各任务均配置了一定的处理器数,且执行过程中不支持资源抢占。调度空间中存在多个支持预留的计算资源,执行过程中资源代理能实时预测各资源的空闲处理器数及相应的时间槽,用户通过资源代理对空闲资源进行查询并付费使用。任务 i 在匹配资源 r 上的实际执行时间 e_{ir} 与执行费用 c_{ir} 为

$$e_{ir}=\beta_{ir}-\alpha_{ir} \tag{1}$$

$$c_{ir}=p_{ir}\times\varepsilon_i\times c_r \tag{2}$$

其中 β_{ir} 为 i 在 r 上的完成时间; α_{ir} 为 i 的任务提交时间(不考虑任务间的数据传输时间和算法运行开销,因此 α_{ir} 即为 i 的就绪时间) ; ε_i 为任务 i 配置的处理器数; c_r 为资源 r 上每处理器单位时间收取的费用。在某一调度解 S: $V\to R$ 下,工作流的执行时间 $T(S)$ 与执行费用 $C(S)$ 为

$$T(S)=\beta_n \tag{3}$$

$$C(S)=\sum_{i=1}^{n}\sum_{r=1}^{m}c_{ir}\times x_{ir} \tag{4}$$

其中 β_n 为结束任务 n 的完成时间; m 为调度空间中的资源个数; $x_{ir}=\{0,1\}$,当 i 选择 r 时, $x_{ir}=0$,否则 $x_{ir}=1$。假定 $\mathrm{S}=\{S_0,S_1,\cdots\}$ 为所有可能的调度解, S 中的最优解定义如下:

定义 1(最优解) 当优先级因子为 $\sigma(\sigma\in[0,1])$ 时,将满足以下条件的调度解定义为最优解,记为 $\tilde{S}$:

$$\frac{T(S)}{T(\tilde{S})}\times\sigma+\frac{C(S)}{C(\tilde{S})}\times(1-\sigma)\geqslant 1,\forall S\in \mathrm{S} \tag{5}$$

其中 $T(\tilde{S})$ 与 $C(\tilde{S})$ 分别为 $\tilde{S}$ 下的工作流执行时间与执行费用。S 的调度性能通过 $\varphi(S,\tilde{S},\sigma)$ 进行衡量:

$$\varphi(S,\tilde{S},\sigma)=\frac{T(S)}{T(\tilde{S})}\times\sigma+\frac{C(S)}{C(\tilde{S})}\times(1-\sigma) \tag{6}$$

$\varphi(S,\tilde{S},\sigma)$ 越小, S 越接近最优解 $\tilde{S}$,其性能越高。因此基于优先级因子的工作流时间 – 费用优化目标可描述为

$$\text{minimize } \varphi(S,\tilde{S},\sigma) \tag{7}$$

$$\text{s.t. } \beta_j-\beta_i-\sum_{r=1}^{m}e_{jr}\times x_{jr}\geqslant 0,\forall(i,j)\in E \tag{8}$$

$$x_{ir}=\{0,1\},\forall i\in V,\forall r\in R \tag{9}$$

$$\sum_{r=1}^{m}x_{ir}=1,\forall i\in V \tag{10}$$

式(7) 描述了调度目标,式(8) 则表示调度过程中需满足工作流的偏序关系,式(9) 与(10) 表示 W 中的所有任务只能选择且必须选择一个资源进行调度。

2 时间 – 费用优化算法设计与描述

2.1 逆向分层策略

分层策略根据工作流的拓扑结构对任务进行分层,将存在偏序关系的工作流分解成多个独立并行任务集进行调度。本文分析工作流的同步完成特征,基于任务的逆向深度(距离结束任务的最大边数,记为 BD) 对工作流进行分层,其中第 k 层任务 $BL(k)$ 中包含所有逆向深度为 k 的任务:

$$BL(k)=\{i\mid BD(i)=k\},0\leqslant k\leqslant BD(1) \tag{11}$$

对于 W 中的各层任务,当该层任务均能在同一时刻执行完成时,便能充分利用工作流的同步完成特征,达到较好的费用优化效果。以下为各层定义层完成时间:

定义 2(层完成时间) 定义某层任务的层完成时间为该层任务的最大完成时间:

$$\beta_{BL}(k)=\max_{\forall i\in BL(k)}\beta_i,0\leqslant k\leqslant BD(1) \tag{12}$$

如果 $BL(k)$ 中所有任务的完成时间均为 $\beta_{BL}(k)$,该层中具有相同后继的多个任务便能同步完成。

2.2 任务资源对的衡量标准

在存在资源竞争的网格环境中,任务的资源选择及任务的调度顺序是影响工作流优化效果的关键,为了比较各任务资源对的优劣,本节基于优先级因子设计衡量标准。首先在定义 2 的基础上为任务定义期望完成时间,记为 $\rho(i)$ 。

定义 3(期望完成时间) 定义任务 $i(i\in BL(k))$ 的期望完成时间为该层的层完成时间 $\beta_{BL}(k)$ 。

由以上分析可知,当任一任务 i 均在 $\rho(i)$ 时刻执行完成时,同层任务便能在该层的层完成时间同步完成,从而达到充分利用费用优化空间进行费用优化的目标。然而,在工作流的实时调度过

程中，某层任务的层完成时间无法在该层任务全部完成之前提前预知，因此将任务的期望完成时间设为该层所有已被调度任务的最大完成时间。当 $BL(k)$ 中没有任务被调度时，将 $\rho(i)$ 初始化为上层任务的完成时间：$\rho(i) = \beta_{BL}(k+1)$。在对某层任务的调度过程中，由于存在多个可选任务，同一任务也可以在多个匹配资源上进行选择，以下基于任务的期望完成时间设计衡量标准 θ 对这些可行任务资源对进行衡量：

$$\theta(i,r) = \frac{\max\{\beta_{ir},\rho(i)\} - \alpha_{ir}}{\overline{p_i}} \times \sigma + \frac{c_{ir}}{\overline{c_i}} \times (1-\sigma) \tag{13}$$

其中 $\max\{\beta_{ir},\rho(i)\} - \alpha_{ir}$ 为任务资源对 (i,r) 基于期望完成时间的实际执行时间，c_{ir} 为实际执行费用，$\overline{p_i}$ 与 $\overline{c_i}$ 为 i 在所有匹配资源 R_i 上的平均执行时间与平均执行费用：

$$\overline{p_i} = \frac{\sum_{r\in R_i} p_{ir}}{|R_i|} \tag{14}$$

$$\overline{c_i} = \frac{\sum_{r\in R_i} c_{ir}}{|R_i|} \tag{15}$$

$|R_i|$ 为 R_i 中的资源个数。式(13)中目标值与平均值的比值反应了任务资源对 (i,r) 的各目标值相对于平均值的改进比例，因此 $\theta(i,r)$ 比较客观地衡量了 (i,r) 在优先级因子 σ 下的综合性能，解决了不同优化目标间由于单位和度量标准不同而造成的无法比较的问题。

2.3　算法描述及复杂性分析

在以上逆向分层策略和衡量标准的基础上，本节在 sufferage、min-min 及 min-max 策略的基础上提出三种基于逆向分层的实时优化算法：BLSuff、BLMin 及 BLMax。算法均采用 2.1 节描述的分层策略对任务进行分层并逐层进行调度，BLSuff 算法描述如下：

算法 1：BLSuff

输入：W 及 R 的相关信息；

输出：T 和 C；

1) 基于逆向分层策略对 W 进行分层；

2) for $k = BD(1)$ to 0

3) foreach $BL(k)$ 中未被调度的任务 i

4) foreach $r \in R_i // R_i$ 为 i 的匹配资源列表

5) 在 r 上查找时间槽并按式(13)计算 $\theta(i,r)$；

6) 将 $(i,r,\theta(i,r))$ 加入列表 q_i 中；

7) endfch

8) 在 q_i 中选择 θ 最小的任务资源对 (i,r') 并计算其 sufferage 值，记为 $\mu(i,r')$；

9) 将 $(i,r',\mu(i,r'))$ 加入列表 q_k 中；

10) endfch

11) 在 q_k 中选择 μ 值最大的任务资源对 (i,r')，将 i 调度到 r' 上，并更新 i、r' 的状态；

12) 如果 $BL(k)$ 中仍存在未被调度的任务则转 3)；

13) endfor

14) 按式(3)、(4)计算 T、C 并返回结果；

算法通过两层 foreach 循环在某层任务中选择一个任务进行调度，而调度过程需对 W 中的所有任务进行调度，因此总的时间复杂度为 $o(n^2 \times m)$。

BLMin 与 BLMax 算法流程与 BLSuff 基本相同，简单起见，本文不对 BLMin 与 BLMax 进行详细描述。

3　实验结果与分析

为了评估以上三种分层优化算法的性能，本文在模拟实验环境中实现了以上三种算法，同时将算法的调度结果与 BDLS 算法和贪婪算法 Greedy[11] 进行比较。所有算法均采用 Java 编程，运行于内存为 2G，主频为 2.53G 的 Intel 双核处理器上。

3.1　实例设计

本文实现了一个工作流随机生成器，能随机生成不同规模的工作流实例。工作流生成过程中对工作流规模(Workflow Size，WS)进行控制，本文设定 WS = {200，400，600，800，1000}。在工作流生成过程中各任务的大小及任务配置的处理器数均随机产生：任务大小通过百万指令数(MI)进行衡量，为 $(1\times10^4 \sim 2\times10^5)$ 之间的随机数；任务所配置的处理器数则取(1 ~ 16)之间的随机数。

调度空间中的计算资源也随机生成，由于网格环境下的资源规模各不相同，且多个用户同时共享计算资源，因此资源负载也可能随时发生变化。本文在资源生成过程中对资源规模(Resource Size，RS)和资源的平均负载(Resource Load，RL)进行控制：分别设定 RS = {10，20，30，40}，RL = {0.2，0.5，0.8}。同时各资源的处理器数目也随机产生，本文设定为(20 ~ 128)之间的随机数，当某一资源上的空闲处理器数满足任务所需的最小处理器要求时，该任务才能在资源上调度执行；单处理器的执行速度通过每秒执行的百万指令数(MIPS)进行衡量，为(200 ~ 600)之间

的随机数;处理器单位时间收取的费用与处理器速度相关,速度较快的处理器收取的费用也较高。此外,实验过程中设定优先级因子 $\sigma = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$,因此实验总共进行的实验组为 $5 \times 4 \times 3 \times 6 = 360$。

3.2　性能度量指标

由于多项式时间内无法得到最优解 $\tilde{S}$,因此无法直接对目标函数 $\varphi(S, \tilde{S}, \sigma)$ 进行比较。本文通过各算法调度解相对于五种算法平均值的改进比来进行比较,分别从综合性能改进比(Improvement Ratio of Time-Cost,IRTC)、执行时间改进比(Improvement Ratio of Time,IRT)与执行费用改进比 IRC(Improvement Ratio of Cost,IRC)等方面进行衡量。此外,为了比较算法效率,对算法的平均运行开销 ART(Average Runtime,ART)进行比较。各性能度量指标如下:

$$\mathrm{IRTC} = \frac{\sum_{i=1}^{N}\left\{\frac{\overline{T_i} - T_i(S_H)}{\overline{T_i}} \times \sigma_i + \frac{\overline{C_i} - C_i(S_H)}{\overline{C_i}} \times (1 - \sigma_i)\right\}}{N} \tag{16}$$

$$\mathrm{IRT} = \left(\sum_{i=1}^{N} \frac{\overline{T_i} - T_i(S_H)}{\overline{T_i}}\right) / N \tag{17}$$

$$\mathrm{IRC} = \left(\sum_{i=1}^{N} \frac{\overline{C_i} - C_i(S_H)}{\overline{C_i}}\right) / N \tag{18}$$

$$\mathrm{ART} = \left(\sum_{i=1}^{N} RT_i(H)\right) / N \tag{19}$$

其中 H 代表不同的启发式算法;S_H 为算法 H 给出的调度解;N 为实例个数;$\overline{T_i}$ 与 $\overline{C_i}$ 分别为五种算法给出调度解的平均执行时间与平均执行费用,$T_i(S_H)$ 与 $C_i(S_H)$ 则为调度解 S_H 的执行时间与执行费用,σ_i 为优先级因子的值;$RT_i(H)$ 为算法 H 的运行时间。

3.3　结果比较与分析

表1给出了五种算法在所有实例上的计算结果。

表1　五种算法的性能比较

Tab.1　Comparison of five heuristics

算法	BLSuff	BLMin	BLMax	Greedy	BDLS
IRTC(%)	6	5	4	-3	-12
IRT(%)	10	6	6	13	-35
IRC(%)	11	9	8	-27	0
ART(s)	4.9	2.1	2.3	0.74	52.1

由表可知,三种分层优化算法的效果总体上优于其他两种算法,综合性能高于五种算法平均值的5%左右,而单个目标的性能则高于平均值的6%~11%不等,同时 BLSuff 的优化效果稍优于 BLMin 与 BLMax;三种分层优化算法的单目标改进比优于综合性能改进比,这表明这些算法的优化效果受优先级因子的影响不如另外两种算法明显,在对优先级较高的目标进行优化的同时也对优先级较低的目标进行了较好的优化。Greedy 算法对执行时间进行了很好的优化,但对执行费用的优化效果则比较差,表中数据表明,Greedy 算法在不同调度实例下得到的工作流执行时间小于五种算法得到的平均执行时间,但该算法得到的工作流执行费用则远高于五种算法的平均执行费用,这说明了与 BLSuff 等算法相比,Greedy 算法以较大的费用损失(约36%)换取了较小的时间优化(约9%),因此,综合性能低于三种分层优化算法。其原因解释如下:Greedy 算法在调度过程中只考虑了单个任务的优化效果,在执行过程中可能产生较多的"时间碎片",从而浪费了费用优化空间,因此,费用优化效果比较差。BDLS 在执行时间与执行费用上优化效果都不理想,其原因为:BDLS 通过各目标上的排序值大小来对任务资源对的优劣进行衡量,没有考虑各目标值实际大小的差异,可能以某一目标的较大损失换取另一目标的较小优化,因此优化效果比较差。

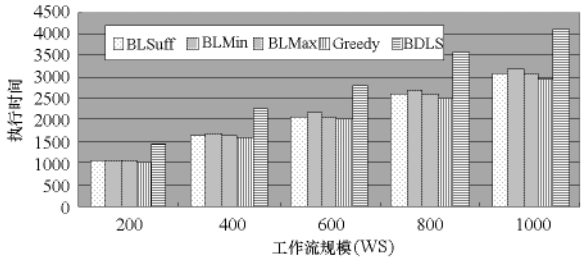
此外,从算法开销来看,三种分层优化算法的时间开销比较接近,其中 BLSuff 稍大于 BLMin 与 BLMax,为 4.9s,Greedy 的算法开销最小,为 0.74s,而 BDLS 则远大于这四种算法,平均为 52.1s。这表明 Greedy 算法效率最高,三种分层优化算法的算法开销仍比较合理,但 BDLS 的算法开销比较大,其原因为:BDLS 在每次调度时,需在各层的所有可行任务资源对中为时间维和费用维进行排序,并在此基础上进行综合排序,因此算法开销比较大。

3.4　不同问题参数下的算法比较

三种分层优化算法在所有实例上的平均性能均优于 Greedy 与 BDLS。为了对算法性能进行全面比较,以下在不同问题参数下对各算法进行比较,问题参数包括工作流规模、资源规模、资源负载及优先级因子。

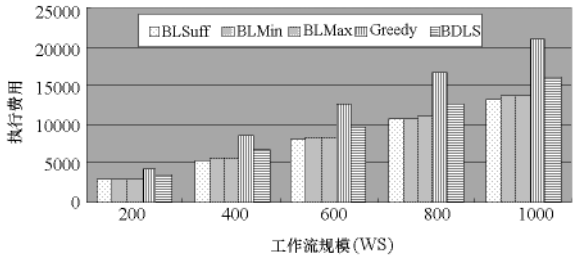
图1为工作流规模 WS = {200, 400, 600, 800, 1000} 时各算法的比较情况。如图所示,在各种工作流规模下,三种分层优化算法得到的工作流执行时间基本相等,其中 BLSuff 在时间方面的性能稍优于 BLMin 与 BLMax,Greedy 的平均执行时间最小,而 BDLS 则远大于前四种算法。从执行费用来看,三种分层优化算法的执行费用基

本相等且远小于 Greedy 与 BDLS。这表明三种分层优化算法能在不同工作流规模下对工作流的执行时间与执行费用同时进行较好的优化; Greedy 算法的时间优化效果比较好，但以较高的执行费用为代价; BDLS 的时间与费用优化效果都不太理想。同时，随着工作流规模的增长，各算法的平均执行时间与平均执行费用均有所增长。



(a) 执行时间比较

(a) Comparison of average times



(b) 执行费用比较

(b) Comparison of average costs

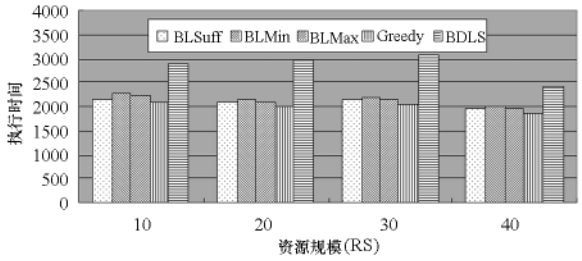
图 1　不同工作流规模下的算法比较

Fig. 1　Comparison of five heuristics at different workflow sizes

图 2 为不同资源规模下的算法比较情况。如图所示，在不同工作流规模下，BDLS 的平均执行时间远大于前四种算法，平均执行费用也明显大于 BLSuff 等算法; Greedy 算法的平均执行时间虽稍小于三种分层优化算法，但其平均执行费用远大于三种分层算法，因此 Greedy 与 BDLS 的综合性能均不太好。

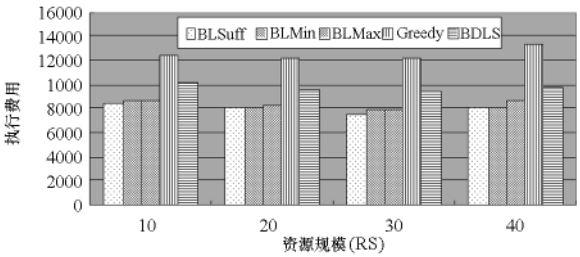
图 3 为不同资源负载下的算法比较情况。如图所示，随着资源负载的增大，BDLS 的时间优化效果有所降低，这是由于 BDLS 在优化过程中没有考虑目标值的实际大小，资源负载的增加使得各任务资源对在时间目标上的差异增大; 而各算法的费用优化效果在不同负载下基本保持不变。

图 4 为不同优先级因子下的算法比较情况。当优先级因子 $\sigma = 0$ 时，各算法只对执行费用进行优化，因此各算法均能找到费用最小的调度解，但各任务执行顺序的差异导致了执行时间稍有差异。当 $\sigma = 1$ 时，各算法只对执行时间进行优化，在此情况下，BDLS 与 Greedy 演变成工作流最小



(a) 执行时间比较

(a) Comparison of average times

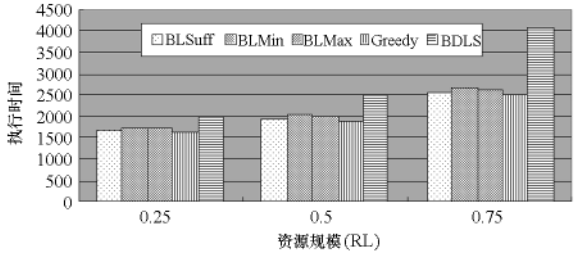


(b) 执行费用比较

(b) Comparison of average costs

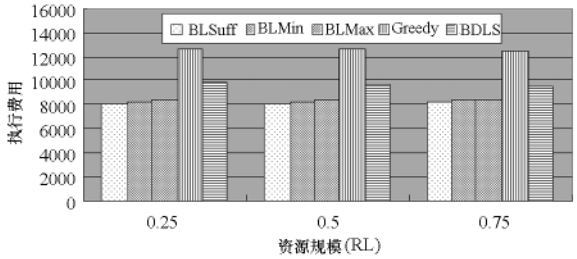
图 2　不同资源规模下的算法比较

Fig. 2　Comparison of five heuristics at different resourse sizes



(a) 执行时间比较

(a) Comparison of average times



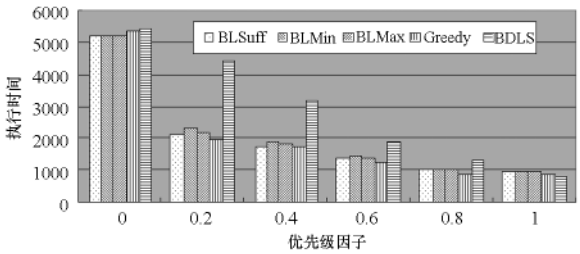
(b) 执行费用比较

(b) Comparison of average costs

图 3　不同资源负载下的算法比较

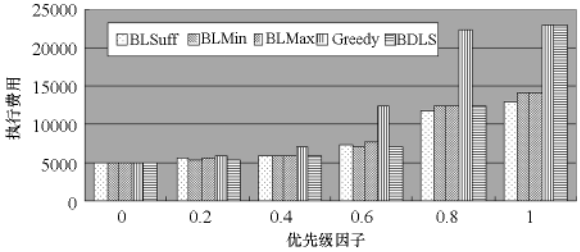
Fig. 3　Comparison of five heuristics at different resourse load

执行时间算法，对执行时间进行了较好的优化; 而三种分层优化算法的时间优化效果较差，但费用优化效果非常明显。当 $\sigma = \{0.2, 0.4, 0.6, 0.8\}$ 时，三种分层优化算法的时间优化效果和费用优化效果均比较好，Greedy 的费用优化效果较差，BDLS 的时间优化效果较差。



(a) 执行时间比较

(a) Comparison of average times



(b) 执行费用比较

(b) Comparison of average costs

图 4 不同优先级因子下的算法比较

Fig. 4 Comparison of five heuristics at different σ

4 总 结

针对动态效用网格下的工作流时间 – 费用优化问题,分析工作流的同步完成特征,采用逆向分层策略将工作流进行分层调度,并基于 sufferage、min-min 及 min-max 策略提出三种基于优先级因子的实时优化算法: BLSuff、BLMin 与 BLMax。算法设计基于优先级因子的衡量标准对工作流的执行时间与执行费用同时进行优化,同时为任务定义期望完成时间,使得同层任务尽量同步完成,达到充分利用费用优化空间进行费用优化的目标。实验结果证明了三种分层优化算法的优越性。

参考文献(References)

[1] Wieczoreka M, Hoheisel A, Prodan R. Towards a general model of the multi-criteria workflow scheduling on the grid [J]. Future Generation Computer Systems, 2009, 25(3): 237 – 256.

[2] 张卫民,刘灿灿,骆志刚. 科学工作流技术研究综述 [J]. 国防科技大学学报, 2011, 33(3): 56 – 65.
ZHANG Weimin, LIU Cancan, LUO Zhigang. A review on scientific workflows [J]. Journal of National University of Defense Technology, 2011, 33(3): 56 – 65. (in Chinese)

[3] Hazir O, Haousri M, Erel E. Discrete time/cost trade-off problem: A decomposition-based solution algorithm for the budget version [J]. Computers & Operations Research, 2010, 37(4): 649 – 655.

[4] 苑迎春, 李小平, 王茜, 等. 基于逆向分层的网格工作流调度算法 [J]. 计算机学报, 2008, 31(2): 282 – 290.
YUAN Yingchun, LI Xiaoping, WANG Qian, et al. Time optimization heuristics for scheduling budget-constrained grid workflows [J]. Journal of Computer Research and Development, 2009, 46(2): 194 – 201. (in Chinese)

[5] Yuan Y, Li X, Wang Q, et al. Deadline division-based heuristic for cost optimization in workflow scheduling [J]. Information Sciences, 2009, 15(179): 2562 – 2575.

[6] 苑迎春, 李小平, 王茜, 等. 成本约束的网格工作流时间优化方法 [J]. 计算机研究与发展, 2009, 46(2): 194 – 201.
YUAN Yingchun, LI Xiaoping, WANG Qian, et al. Time optimization heuristics for scheduling budget-constrained grid workflows [J]. Journal of Computer Research and Development, 2009, 46(2): 194 – 201. (in Chinese)

[7] Garg S, Buyya R, Siegel H. Time and cost trade-off management for scheduling parallel applications on Utility Grids [J]. Future Generation Computer Systems, 2010, 26(18): 1344 – 1355.

[8] Dogan A, Ozguner F. Biobjective scheduling algorithm for execution time-reliability trade-off in heterogeneous computing systems [J]. The Computing Journal, 2005, 48(3): 300 – 314.

[9] Yuan Y, Li X, Wang Q. Time-cost tradeoff dynamic scheduling algorithm for workflows in grids [C] //Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design, Nanjing, 2006: 1 – 6.

[10] Deelman E, Singh G, Su M, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems [J]. Science Programming, 2005, 13(3): 219 – 237.

[11] Ruiz R, Stuzle T. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives [J]. European Journal of Operational Research, 2008, 187(3): 1143 – 1159.