基于启发式分段的网格工作流费用优化方法

龙 浩[1,2]，邸瑞华[1]，梁 毅[1]
(1. 北京工业大学 计算机学院，北京 100124; 2. 江西师范大学 软件学院，南昌 330022)

摘 要：针对有向无环图(directed acrylic graph，DAG)表示的截止期约束下的网格工作流费用优化问题，提出启发式分段(segment level，SL)费用优化算法. 通过分析DAG图中活动的并行和同步特征，算法对活动进行分段，时间浮差按比例分配到各段，段内的费用优化采用动态规划的求解策略实现. 通过将工作流截止期转换为段截止时间，扩大了活动的费用优化区间，通过大量模拟实验将SL算法和MCP(minimum critical path)、DTL(deadline top level)、DBL(deadline bottom level)算法比较，证明了SL算法的有效性.

关键词：网格工作流；有向无环图；启发式算法；分段
中图分类号：TP 393 **文献标志码**：A **文章编号**：0254－0037(2011)04－0583－08

网格[1-2]把分布、异构、自治的资源汇集成一个支持资源共享和任务协作的动态计算环境，是近年来在商务和科学研究中兴起的一种Internet计算模式. 网格工作流把一些关联的网格服务组合成能自动调度与执行的集成应用，有效地增强了网格系统的可扩展性和适应性. 由于网格应用在资源种类、管理策略、用户需求和应用场景的巨大差异，人们使用不同的QoS(quality of service)参数(如执行时间、可靠性、可访问性等)来描述网格服务质量，基于不同QoS需求的网格工作流调度成为一个复杂的问题，它们通常都是NP-hard[3-4]. 时间－成本均衡问题是用户最关心的，它考虑在截止期(deadline)约束下如何最小化总费用.

有向无环图DAG(directed acyclic graph)[5-6]是网格工作流的常用描述方法，线性规划[7]、遗传算法[8-9]、模拟退火算法[10]、混合粒子群算法[11]等都被用来解决DAG描述的网格工作流调度问题，它们通常能获得较好的性能，但当问题规模较大时，算法的时间开销很大. 构造式启发算法能保证以合理的时间开销获得问题的可接受解，被广泛用于求解工作流的时间费用优化问题. Jia等[12]和苑迎春等[13]提出的正向截止期优化方法DTL(deadline top level)和反向截止期优化方法DBL(deadline bottom level)按模型结构对工作流进行分层，将截止期平均分配到各层，通过各层活动的局部费用优化最终得到全局近似最优解. 由于分层的宽度很小，费用的优化区间很小而且没有考虑到任务的可选服务之间的联系，算法性能还有很大的改进空间. 文献[11]提出的TD(greedy cost-time distribution)方法按照DAG图中活动的并行和同步特征对工作流进行分层和时间划分，但没有给出具体的算法，而且该方法只适用于简单结构的工作流调度.

针对截止期优化算法和TD算法的不足，本文提出截止期约束的启发式分段(segment level，SL)费用优化算法，通过分析DAG图中活动的并行和同步特征，对活动进行分段，将工作流截止期转换为段截止时间. 将时间浮差按比例分配到各段，以尽量扩大活动的费用优化区间，段内费用的最优化采用动态规划的求解策略实现. 算法同时考虑了工作流模型结构和任务的可选服务之间的关联，能更合理地优化费用. 模拟结果验证了算法的性能和效率较高.

1 问题描述

描述网格工作流的DAG可用三元组$G=\langle V,L,E\rangle$定义. 其中，$V=\{v_i \mid 1\leqslant i\leqslant n\}$表示图中任务(或称

收稿日期：2009-06-17.
基金项目：北京市教育委员会基金资助项目(JJ007011200901).
作者简介：龙 浩(1970—)，男，江西泰和人，博士研究生.

活动、节点) 集合,n 为节点数; $L=\{l_{ij}\mid 1\leqslant i,j\leqslant n,\ i\neq j\}$ 是图中的有向边集合,l_{ij} 表示存在有向边 $v_i\rightarrow v_j$,代表节点 v_i 与 v_j 执行上的时序关系,即节点 v_j 在 v_i 执行完成之后才能开始,称 v_i 为 v_j 的前驱节点,v_j 为 v_i 的后继节点; 每个节点有不同的可选服务集,$E=\{E_i\mid 1\leqslant i\leqslant n\}=\{e_{is}\mid 1\leqslant i\leqslant n,1\leqslant s\leqslant m_i\}$ 表示任务 i 的可选服务的 QoS 属性集合,m_i 是任务 i 的可选服务数. 本文主要关注可选服务的执行时间与费用,$e_{is}=\langle c_{is},t_{is}\rangle$,其中 c_{is}、t_{is} 分别是任务 i 的可选服务 s 的执行时间和费用. 为方便算法实现,对 DAG 描述的工作流应用作以下假设:

1) 对 DAG 图进行变形,使入度为 0 的节点(即起点) 和出度为 0 的节点(即终点) 唯一. 若图有 2 个或 2 个以上的起点,则加入 1 个虚拟节点,使之成为唯一的入口节点; 若有 2 个或 2 个以上的终点,则加入 1 个虚拟节点,使之成为唯一的出口节点. 入口和出口节点的执行时间和费用都为 0. 因此,结点 1 和 n 分别是图中的入口节点和出口节点.

2) 工作流内部处理时间忽略不计,所有可选服务都能提供所承诺的服务质量,则工作流运行时间和成本取决于各个节点选择的服务.

3) 每个节点的各可选服务的执行时间和费用均不相同且费用是时间的离散递减函数. 对于任务 i,可选服务 s 编号越大,执行成本越低,执行时间越长.

任何工作流调度都有一条关键路径(运行时间最长的路径) 与之对应. 工作流完工时间就是关键路径的运行时间,工作流总费用是所有活动花费之和. 图 1 是一个简单的工作流实例,表 1 是各节点的服务池实例.

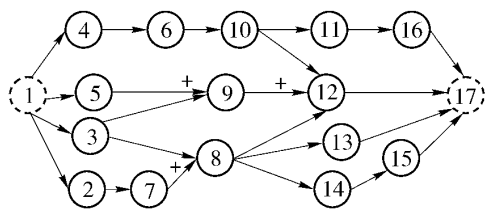


图 1 简单的工作流实例

Fig. 1 A workflow example represented with DAG

表 1 图 1 中各活动的服务池实例

Table 1 Service instances for different nodes of the workflow example in Fig. 1

活动编号	E_i	m_i	活动编号	E_i	m_i
1	{(0,0)}	1	10	{(200,15),(150,18),(120,25),(100,30)}	4
2	{(11,6),(8,8),(6,10)}	3	11	{(80,6),(50,10)}	2
3	{(12,4),(10,5)}	2	12	{(18,9)}	1
4	{(5,6)}	1	13	{(50,20),(40,25)}	2
5	{(15,2),(10,4)}	2	14	{(150,15),(120,20),(80,30)}	3
6	{(20,1),(10,2),(5,3)}	3	15	{(60,10),(50,13)}	2
7	{(30,10),(25,15)}	2	16	{(30,15),(25,20),(20,30)}	3
8	{(30,3),(20,6),(10,9),(5,10)}	4	17	{(0,0)}	1
9	{(18,5),(14,8),(10,12),(8,16),(6,18)}	5			

时间费用优化问题的形式化描述如下

$$\min \sum_{i}\sum_{j} x_{is}c_{is},\ 1\leqslant i\leqslant n,1\leqslant s\leqslant m_i \tag{1}$$

$$\text{s.t.}\sum_{i=1}^{m_j} x_{is}=1,\ 1\leqslant i\leqslant n \tag{2}$$

$$f_u+\sum_{s=1}^{m_i} x_{is}t_{is}\leqslant f_v,\ 1\leqslant u\leqslant i\leqslant v\leqslant n \tag{3}$$

$$f_1=0 \tag{4}$$

$$f_n\leqslant T_0 \tag{5}$$

$$x_{is}=\{0,1\},\ 1\leqslant i\leqslant n,1\leqslant s\leqslant m_i \tag{6}$$

其中，f_i 表示节点 i 的完工时间；f_n 是工作流完工时间；T_0 是截止期，表示完成工作流所有任务所需的最晚完成时间，一般情况下它是用户指定的一个常数；x_{is} 是布尔变量，它的取值表示节点 i 的可选服务 s 是否被调度方案选中. 式(1)是最小化工作流总费用；式(2)表示各节点只能选择 1 个服务执行；式(3)表示任务满足偏序约束；式(4)、(5)是截止期约束；式(6)说明 x_{is} 是布尔变量.

2　分段时间成本优化算法

SL 算法的基本思想是把一个全局优化问题转化为多个局部优化的子问题. 首先挑选出 DAG 图中的段节点，将段节点的并行节点集、相邻段节点内部的可串归约化节点都构成不同的段，将时间浮差按比例分配到各段，这样就将工作流截止期转换为段截止时间，可以尽量扩大活动的费用优化区间. 段内费用的最优化采用动态规划的求解策略实现.

2.1　相关定义与性质

首先给出最大完工时间、最小完工时间、时间浮差、段节点、直接前驱节点集的定义.

定义 1　给定图 $G=\langle V,L,E\rangle$，如果每个节点都选择运行最快的服务，则该调度方案的关键路径称为最小关键路径 MCP(minimum critical path)，其路径长度称为最小完工时间，记为 M，$M=\sum_{i\in P}\min\{c_{is}\}$，$1\leqslant s\leqslant m_i$，$P$ 是关键活动集. 如果每个节点都选择最慢的服务，则该调度方案的关键路径称为最大关键路径，其路径长度称为最大完工时间，记为 N，$N=\sum_{i\in P}\max\{c_{is}\}$，$1\leqslant s\leqslant m_i$，$P$ 是关键活动集.

M 是完成工作流所有任务所需的最小时间，N 是完成工作流所有任务所需的最大时间. 用户给定的截止期 T_0 必须满足 $M\leqslant T_0\leqslant N$，$T_0$ 距 M 越近，所花费的成本越高；T_0 距 N 越近，所花费成本越低.

定义 2　给定图 $G=\langle V,L,E\rangle$，对节点集 $\{v_i,\cdots,v_j\}\in V$，若同时满足：

1) 除节点集的起点 v_i 和终点 v_j 外，每个节点都有唯一的入度和出度；

2) 每个节点的前驱节点和后续节点都属于该节点集；则称节点集 $v_i,\cdots,v_j$ 为可串归约，$\{v_{i0},\cdots,v_j\}$ 称为一条可串归约路径.

定义 3　$W=\{v_0,v_n\}\cup\{v_i\mid\exists l_{ij},l_{ik}\in L,i\neq j,i\neq k\}\cup\{v_j\mid\exists l_{ij},l_{kj}\in L,i\neq j,i\neq k\}$ 称为 DAG 图的段节点集，W 包括开始节点、结束节点、所有的“AND-JOIN”节点和“AND-SPLIT”节点. 如果 2 个段节点之间(不含段节点本身)至少有 1 条可串归约路径，则这 2 个段节点称为相邻段节点.

定义 4(直接前驱节点集)　任务 v_j 的直接前驱节点集 P^j 是 V 的子集，它可以递归定义如下

$$\begin{cases}Q_1^j=\{v_j\}\\ P_1^j=\{v_i\mid\exists l_{ik}\in L\wedge\neg\ \exists(l_{ij}\in L\wedge l_{jk}\in L),v_i,v_j\in V,v_k\in Q_1^j\}\\ Q_n^j=Q_{n-1}^j\cup\{v_i\mid\exists l_{ki}\in L\wedge\neg\ \exists(l_{ij}\in L\wedge l_{kj}\in L),v_k\in P_{n-1}^j,v_i,v_j\in V\}\\ P_n^j=P_{n-1}^j\cup\{v_i\mid\exists l_{ik}\in L\wedge\neg\ \exists(l_{ij}\in L\wedge l_{jk}\in L),v_i,v_j\in V,v_k\in Q_n^j\}\end{cases}\tag{7}$$

相应地，P^j 称为 Q^j 的直接前驱节点集，Q^j 称为 P^j 的直接后继节点集. 显然有 $v_j\in Q^j$，如果 v_j 是段节点，则称 $Q^j-\{v_i\mid v_i\in V,v_i$ 不是段节点$\}$ 为 v_j 的并行节点集.

DAG 图的分段规则是：首先将图中包括开始节点、结束节点、所有的“AND-JOIN”节点和“AND-SPLIT”节点作为段节点，按照定义 5 获得段节点的并行节点集，将段节点的并行节点集、相邻段节点内部的节点集分别划分为不同的段. 划分得到的段用 S_x 表示，其中 x 是段编号.

定义 5　$\forall S_x,S_y\subset V,x\neq y$，如果 S_x 的任务完成后 S_y 中的任务才能开始，则称 S_x 是 S_y 的前段，相应地称 S_y 为 S_x 的后段. 如果 S_x、S_y 的活动可以并行执行，则称 S_x 和 S_y 是并行段.

定理 1　给定图 $G=\langle V,L,E\rangle$，$V=\bigcup_x S_x$，则：

1) $\forall v_i,v_j\in S_x,i\neq j$，如果 S_x 内包含段节点，则 v_i 和 v_j 之间无偏序约束，可并行执行.

2) 可串归约路径上的活动集划分在同一个段.

3) $\forall S_x, S_y \subset V, x \neq y$,有 $S_x \cap S_y = \varnothing$.

证明: 定理 1 中的 1)、2) 直接根据定义 5 和分段规则得证,3) 用反证法证明. 假设 $\exists S_x, S_y \subset V, x \neq y$,有 $S_x \cap S_y \neq \varnothing$,则至少存在节点 v_k,满足 $v_k \in S_x \wedge v_k \in S_y$.

a) 如果 v_k 是段节点,则 S_x、S_y 都是 v_k 的并行节点集,S_x、S_y 是同一个段,与假设矛盾.

b) 如果 v_k 是段节点 $v_{k'}$ 的并行节点,则 S_x、S_y 都是 $v_{k'}$ 的并行节点集,S_x、S_y 是同一个段,与假设矛盾.

c) 如果 v_k 是入度和出度均为 1 的节点,必然有 $v_i \in S_x, l_{ik} \in L$, $v_j \in S_y, l_{kj} \in L$. 显然 v_i、v_j 都不是段节点(如果 v_i 是段节点,则 $v_k \notin S_x$; 如果 v_j 是段节点,则 $v_k \notin S_x$,与假设矛盾),所以$\{v_i, v_j, v_k\}$ 是可串行归约的,根据 2),它们必然被分割到同一个段,与假设矛盾.

综合 a)、b)、c) 得证.

定理 1 说明,同属于一个段的任务或者具有并行执行特点,或者属于 1 个或几个可串行归约路径. 对同段任务设置相同的开始时间和截止时间,可保持同步完成特征. 包含较多任务的段具有较大的时间区间,能尽量对成本进行优化. 当每个任务选择最快服务执行时,各段的截止时间设置为

$$\begin{cases} \delta_y = 0, & S_y \text{ 包含 } v_0 \\ \delta_y = \max\{\delta_x\} + \max \sum_{k \in y_l} \min_{1 \leqslant j \leqslant m_k} (t_{kj}), & S_x \text{ 是 } S_y \text{ 的前段,} y_l \text{ 是 } S_y \text{ 中的一条路径} \\ \delta_y = \max\{\delta_x\}, & S_x \text{ 是 } S_y \text{ 的前段,} S_y \text{ 包含 } v_n \end{cases} \tag{8}$$

由式(8) 确定的工作流完工时间称为分段最小完工时间,记作 $\delta_{\min}$.

定理 2 给定图 $G = \langle V, L, E \rangle$,都有 $\delta_{\min} \geqslant M$.

证明: 由定义 1 可知 $M = \sum_{i \in P} \min\{c_{is}\}, 1 \leqslant s \leqslant m_i$,假设 $P = \{v_{i_0}, v_{i_1}, \cdots, v_{i_k}\}$. 根据分段原则,$P$ 中的每个段节点都划分为 1 个段,P 上每 2 个相邻段节点内部的节点也相应地划分为 1 个段. 假设划分得到的段是$\{S_{x_0}, S_{x_1}, \cdots, S_{x_k}\}$.

1) 如果 S_{x_i} 包含段节点,则段中所有节点可以并行执行,有

$$\delta_{x_i} = \max\{\delta_x\} + \max_{k \in S_{x_i}} \min_{1 \leqslant j \leqslant m_k} (t_{kj}) \geqslant \max\{\delta_x\} + \max \min_{1 \leqslant j \leqslant m_k} (t_{k_0, j})$$

S_x 是 S_{x_i} 的前段,k_0 是 S_{x_i} 中的关键节点.

2) 如果 S_{x_i} 不包含段节点,则段中所有节点是 1 条或多条可串归约化路径,有

$$\delta_{x_i} = \max\{\delta_x\} + \max_{y_l \in S_{x_i}} \sum_{k \in y_l} \min_{1 \leqslant j \leqslant m_k} (t_{kj}) = \max\{\delta_x\} + \sum_{k \in y_{l_0}} \min_{1 \leqslant j \leqslant m_k} (t_{kj})$$

S_x 是 S_{x_i} 的前段,y_l 是 S_{x_i} 中的可串归约化路径,y_{l_0} 是 S_{x_i} 中属于关键路径一部分的可串归约化路径.

综合 1)、2),显然有 $\delta_{\min} \geqslant M$.

定义 6 给定 $G = \langle V, L, E \rangle$,定义时间浮差 $T_f = T_0 - \delta_{\min}$. 对于 DAG 的最小关键路径 P,找出覆盖这条路径的各个分段,各段按比例分配到的时间浮差为

$$T_{S_x} = \frac{T_f}{\delta_{\min}} \times \max \sum_{k \in x_l} \min_{1 \leqslant j \leqslant m_k} (t_{kj}) \tag{9}$$

定义 7 给定 $G = \langle V, L, E \rangle$,忽略最小关键路径 P 所包含的节点和对应的并行节点集,剩余节点中具有最大时间长度的路径称为二次最小关键路径,依此类推,以后得到的最大时间长度的路径称为 n 次最小关键路径.

2.2 SL 算法描述

当截止期 T_0 在 $[M, \delta_{\min}]$ 时,采用 MCP 算法对工作流进行调度; 当截止期大于 $\delta_{\min}$ 时,将时间浮差按比例分配到各段. 从定理 2 可以看到,对于给定的 T_0($T_0 \geqslant \delta_{\min}$),DAG 图越简单,$M$ 越接近 $\delta_{\min}$,时间浮差越大,分段算法适用的范围越大. 各段时间浮差按照下列步骤进行分配: 首先找出 DAG 的最小关键路径,找出覆盖这条路径的各个分段,将时间浮差按式(9) 分配到各段; 如果存在未被最小关键路径上的分段覆盖子图 $G' = \langle V', L', E' \rangle$,它们必然是以最小关键路径上的某 2 个节点为开始和结束,据此计算子图的截止期 T_0'、最小关

键路径 P'、最小分段完工时间 $\delta'_{\min}$、时间浮差 T'_{f}，参照式(9)可以计算各段分配到的时间浮差. 如此不断迭代，可得到各分段的截止时间

$$\begin{cases} \delta_y = 0, & S_y \text{ 包含 } v_0 \\ \delta_y = \max\{\delta_x\} + \max \sum_{k \in y_l} \min_{1 \leqslant j \leqslant m_k} (t_{kj}) + T_{S_y}, & S_x \text{ 是 } S_y \text{ 的前段，} y_l \text{ 是 } S_y \text{ 中的一条路径} \\ \delta_y = \max\{\delta_x\}, & S_x \text{ 是 } S_y \text{ 的前段，} S_y \text{ 包含 } v_n \end{cases} \tag{10}$$

算法　SL 算法

1) 输入工作流 DAG 图 $G = \langle V, L, E \rangle$;

2) 计算工作流最小完工时间 M、工作流最大完工时间 N;

3) 输入截止期 T_0($M \leqslant T_0 \leqslant N$);

4) 将 DAG 中的开始节点、结束节点和所有的“AND-JOIN”节点和“AND-SPLIT”节点，加入节点集 SynTaskList;

5) 按照式(6)计算 SynTaskList 中各个节点的并行节点集，每个并行节点集都是并行节点集队列 ParrallelTaskLists 的一个成员;

6) 按照分段规则对 DAG 分段;

7) 计算 P-DAG 的 n 次最小关键路径(n 初值为 1);

8) 计算 P 包含的节点及对应的并行节点集;

9) 按照式(8)计算 $\delta_{\min}$;

10) 如果 SynTaskList = ParrallelTaskLists and $T_0 < \delta_{\min}$，采用 MCP 进行调度，调转 16);

11) 按照式(9)计算各段的时间浮差;

12) 按照式(10)计算 P 上各段的开始和截止时间;

13) 各段按照动态规划方法选择各节点的可选服务;

14) 将 P 上各段包含的节点从 V、ParrallelTaskLists 中分别删除;

15) 如果 $V \langle\rangle$null，$n = n + 1$，跳转 7);

16) 计算调度结果的总成本 C、完工时间 T.

在 SL 算法中，假定的工作流 DAG 图和各节点的服务池服务是严格按照次序输入的. 步 2)、4)、5)、6)、7)的时间复杂度为 $O(nml)$，步 7)的时间复杂度为 $O(lmn^2)$，步 8)、11)、12)的时间复杂度为 $O(mn^3)$，步 13)的时间复杂度为 $O(mn^4)$. 其中，l 为工作流 DAG 图的边数，n 为节点数，m 为节点的服务池服务数，因此整个算法的时间复杂度最大不超过 $O(nm(l + n^3))$.

2.3　实例说明

为了说明 SL 算法的求解过程，以前面的工作流应用为例介绍算法的求解过程. 工作流中的最小关键路径是{1,2,7,8,14,15,17}，最小完工时间是 44; 最大关键路径是{1,4,6,10,11,16,17}，最大完工时间是 79. 工作流中段节点集是{1,3,8,9,10,12,17}，节点 8、9、10 是节点 12 的直接前驱节点，它们是可以并行执行的. 可串归约节点分别是: 4 和 6，2 和 7，11 和 16，14 和 15，分段结果如图 2 所示.

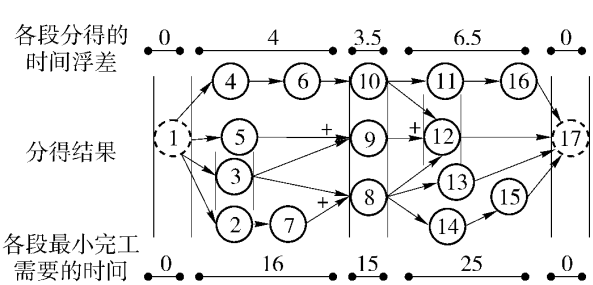


图 2　DAG 分段结果

Fig. 2　The result of SL algorithm

段最小完工时间是 56，假设用户输入的截止期是 70，将时间浮差 70 − 56 = 14 按照比例分配到各段. 按照动态规范方法对各段中的节点进行调度，得到最终方案的总成本为 550，完工时间是 68.

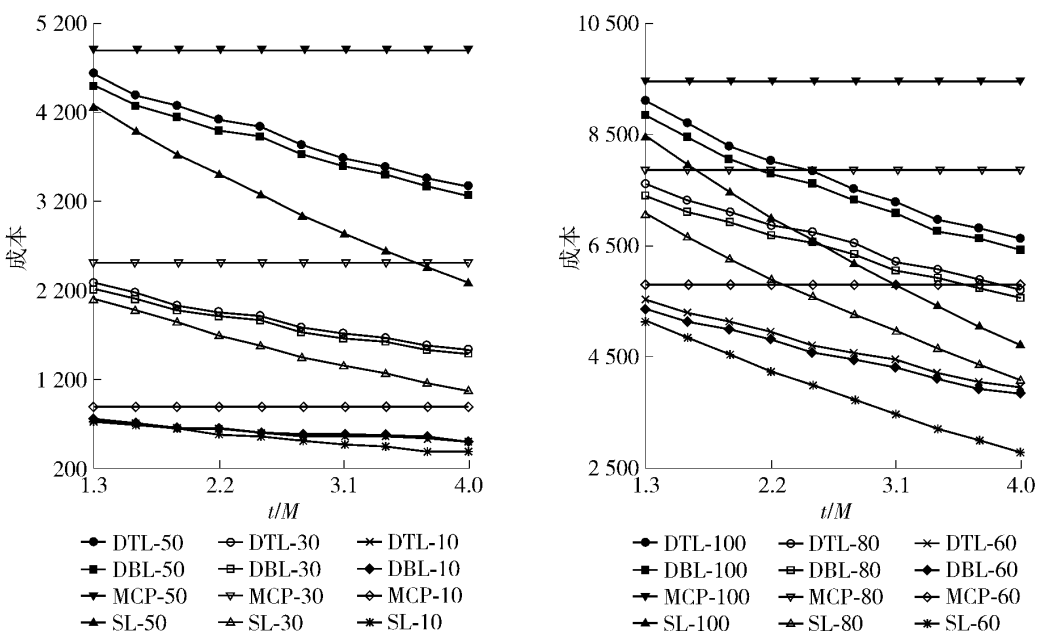
3 实验结果与分析

作者做了 2 个实验分别对算法的性能和运行效率进行模拟测试，模拟实验环境中操作系统为 Windows XP，运行在 Pentium Ⅳ、主频为 2.93 GHz，RAM 为 512 MB 的 PC 机上，算法采用 java 编程. 工作流实例的 DAG 图是自动生成的，DAG 自动生成器需要设置节点数 $|V|$，各节点服务池中服务数量是种子为 15 的随机数，服务运行时间是种子为 10 的递增随机数、成本是种子为 20 的递减随机数，费用是时间的离散严格递减函数. 节点 i 与节点 j 之间是否相连是以 $(j-i) \leqslant 3$ 并与 2 个以 i、j 为种子的随机数生成器的序列中取出的、均匀分布的 boolean 值共同判定. 所有随机数符合均匀分布. SL 算法针对其他算法的性能提高率

$$I_{\mathrm{A}}=\frac{1}{10}\times \sum_{k}\left(A_k-S_k\right)/A_k$$

式中，A_k 是节点数为 k 时算法 A 的成本；S_k 是节点数为 k 时算法 SL 的成本.

实验 1 测试算法的性能，并与 MCP、DTL、DBL 算法进行比较. 测试时节点数分别为{10，20，30，40，50，60，70，80，90，100} 的 10 个应用，每种应用设置 10 个不同截止期的实例，这些实例的截止期分别以工作流最小完工时间 M 的 30% 递增，每个应用的实验结果是所有它的实例的平均值. 图 3(a)、(b) 描述了不同节点、不同截止期下各种调度算法的成本耗费.



(a) 10/30/50任务 (b) 60/80/100任务

图 3 不同任务数、截止期下不同调度算法的成本

Fig. 3 Experimental results according to different deadlines and nodes

从图 3 可以看到，随着截止期的增加，DBL、DTL、SL 这几个算法所需费用都减少，而 MCP 算法不随截止期的变化而变化. SL 算法相对 MCP 算法的性能提高率最大，DTL 次之，相对 DBL 算法的提高率最小. 截止期距 M 越近，算法的性能提高率越不明显；截止期距 M 越远，算法的性能提高率越大. 当截止期只为 M 的 1.3 倍时，SL 算法针对 MCP、DTL、DBL 的性能提高率平均为 12.7%、6.1%、5.7%；当截止期为 M 的 4 倍时，SL 算法针对 MCP、DTL、DBL 的性能提高率分别能达到 54.2%、35.7%、28.9%.

实验 2 测试算法的运行效率，并与 TD 算法进行比较. 测试时节点数分别为{60，70，80，90，100} 的 5 个应用，截止期设定为 M 的 3 倍. 表 2 描述了不同节点、不同截止期下 2 种调度算法的成本与时间开销. 从表 2 可以看到，不同规模下 TD、SL 算法的成本没有多大变化，但运行时间平均减少 8%. 原因是 SL 算法进行了段的合并，而这 2 个算法段内的费用优化都采用动态规划实现.

表 2　不同节点、不同截止期下 2 种调度算法的成本与时间开销

Table 2　Cost and running time of TD, SL under different deadline and nodes

工作流节点	成本		I_{TD}/%	运行时间/ms		I_{TD}/%
	TD	SL		TD	SL	
60	3 619	3 593	0.72	338	312	7.69
70	4 230	4 186	1.04	605	563	6.92
80	5 164	5 077	1.68	904	826	8.32
90	5 203	5 142	1.69	1 275	1 169	8.30
100	6 012	5 967	0.75	1 553	1 416	8.80
平均值	4 846	4 793	1.18	935	857	8.00

4　结论与展望

网格环境下基于分段的工作流调度算法根据工作流的 DAG 结构对任务进行分段，不仅把可串归约活动集中到一起，也能保证并行任务(或任务组)的无冲突并发执行. 算法避免了 DTL、DBL 算法中每层宽度均为 1 的缺陷，扩大了时间费用的优化空间和算法有效使用范围；通过搜索段节点的并行节点集，把可并行执行的可串行化路径合并在同一段中，避免了 TD 算法必须在每 2 个相邻段节点间进行时间划分的缺陷. 实验结果表明，不同结构、不同截止期下算法性能均有较大提高.

参考文献：

[1] FOSTER I, KESSELMAN C. The grid: blueprint for a new computing infrastructure [M]. 2nd ed. San Francisco: Morgan Kaufmann Publishers, 2003: 5-7.

[2] FOSTER I, KESSELMAN C. Grid service for distributed system integration [J]. IEEE Computer, 2002, 35(6): 37-46.

[3] ZENG L Z, BENATALLAH B, NGU A H H, et al. QoS-aware middleware for Web services composition [J]. IEEE Trans on Software Engineering, 2004, 30(5): 311-327.

[4] BLYTHE J, JAIN S, DEELMAN E, et al. Task scheduling strategies for workflow-based applications in grids [C] //Proceedings of the IEEE International Symposium on Cluster Computing and Grid. Cardiff, Wales, UK: IEEE Computer Society, 2005(2): 759-767.

[5] GERASOULIS A, YANG T. A comparison of clustering heuristics for scheduling directed acyclic graphs of multiprocessors [J]. Journal of Parallel and Distributed Computing, 1992, 16(4): 276-291.

[6] KWOK Y K, AMAD I. Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors [J]. IEEE Transactions on Parallel and Distributed Systems, 1996, 7(5): 506-521.

[7] DEMEULEMEESTER E, HERROELEN W, ELMAGHRABY S E. Optimal procedures for the discrete time/cost trade-off problem in project networks [J]. European Journal of Operational Research, 1996, 88(1): 50-68.

[8] 林剑柠，吴慧中. 基于遗传算法的网格资源调度算法 [J]. 计算机研究与发展，2004，41(12)：2190-2194.
LIN Jian-ning, WU Hui-zhong. Scheduling in grid computing environment based on genetic algorithm [J]. Journal of Computer Research and Development, 2004, 41(12): 2190-2194. (in Chinese)

[9] JIA Yu, BUYYA R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms [J]. Scientific Programming, 2006, 14(3/4): 217-230.

[10] 金海，陈汉华，吕志鹏，等. CGSP 作业管理其合成服务的 QoS 优化模型及求解 [J]. 计算机学报，2005，28(4)：578-588.
JIN Hai, CHEN Han-hua, LÜ Zhi-peng, et al. QoS optimizing model and solving for composite service in CGSP job manager [J]. Chinese Journal of Computers, 2005, 28(4): 578-588. (in Chinese)

[11] 于明远，朱艺华，梁荣华. 基于混合微粒群算法的网格服务工作流调度 [J]. 华中科技大学学报：自然科学版，2008，36(4)：45-47.

YU Ming-yuan, ZHU Yi-hua, LIANG Rong-hua. A grid service-workflow schedule using hybrid particle swarm [J]. Journal of Huazhong University of Science and Technology: Natural Science Edition, 2008, 36(4): 45-47. (in Chinese)

[12] JIA Yu, BUYYA R, CHEN Khong-tham. Cost-based scheduling of workflow applications on utility grids [C]// The 1st IEEE Int'l Conf on e-Science and Grid Computing. Melbourne, Australia: IEEE Press, 2005: 140-147.

[13] 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 Ying-chun, LI Xiao-ping, WANG Qian, et al. Bottom level based heuristic for workflow scheduling in grids [J]. Chinese Journal of Computers, 2008, 31(2): 282-290. (in Chinese)

Segment Level Based Heuristic for Workflow Cost-time Optimization in Grids

LONG Hao[1,2], DI Rui-hua[1], LIANG Yi[1]

(1. College of Computer Science, Beijing University of Technology, Beijing 100124, China;
2. College of Software, Jiangxi Normal University, Nanchang 330022, China)

Abstract: Workflow scheduling with the objective of time-cost optimization is a fundamental problem in grids and generally the problem is NP-hard. In this paper, a novel heuristics called SL (Segment Level) for workflows represented by DAG (Directed Acyclic Graph) is proposed. Considering the parallel and synchronization properties, the workflow application is divided into segments, and the workflow deadline is transformed into the time intervals and appointed to different segments. The floating time is prorated to each segment to enlarge cost-time duration, and a dynamic programming method is implemented to optimize cost for each segment. By comparing SL with MCP (Minimum Critical Path), DTL(Deadline Top Level), DBL(Deadline Bottom Level), the heuristics' efficiency is verified by experimental results.

Key words: grid workflow; directed acrylic graph; heuristics; segment level

(责任编辑 梁 洁)