多云环境下带截止日期约束的科学工作流调度策略

林兵[1,2,3]，郭文忠[2,3,4]，陈国龙[2,3]

（1. 福建师范大学物理与能源学院，福建 福州 350117；
2. 福建省网络计算与智能信息处理重点实验室（福州大学），福建 福州 350116；
3. 空间数据挖掘与信息共享教育部重点实验室，福建 福州 350003；
4. 福州大学数学与计算机科学学院，福建 福州 350116）

摘 要：针对多云环境下带截止日期约束的科学工作流调度问题，提出一种基于遗传算法操作的自适应离散粒子群优化算法（ADPSOGA），目的是在尽可能满足工作流截止日期前提下，减少其执行代价。该方法考虑多云之间的通信代价、虚拟机的启动和关闭时间以及多云之间不同的带宽通信波动；为了避免传统粒子群优化算法（PSO, particle swarm optimization）存在的过早收敛问题，引入遗传算法的随机两点交叉操作和随机单点变异操作，有效提高种群进化过程中的多样性；在充分考虑数据通信代价和任务计算代价的情况下，设计一种基于工作流截止日期约束的代价驱动调度策略。实验结果表明，ADPSOGA 在波动因素存在情况下，对工作流截止日期满足和执行代价控制方面具有良好的性能表现。

关键词：云计算；截止日期约束；工作流调度；波动性
中图分类号：TP338
文献标识码：A
doi: 10.11959/j.issn.1000-436x.2018006

Scheduling strategy for science workflow with deadline constraint on multi-cloud

LIN Bing[1,2,3], GUO Wenzhong[2,3,4], CHEN Guolong[2,3]

1. College of Physics and Energy, Fujian Normal University, Fuzhou 350117, China
2. Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing (Fuzhou University), Fuzhou 350116, China
3. Key Laboratory of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou 350003, China
4. College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China

Abstract: In view of the deadline-constrained scientific workflow scheduling on multi-cloud, an adaptive discrete particle swarm optimization with genetic algorithm (ADPSOGA) was proposed, which aimed to minimize the execution cost of workflow while meeting its deadline constrains. Firstly, the data transfer cost, the shutdown and boot time of virtual machines, and the bandwidth fluctuations among different cloud providers were considered by this method. Secondly, in order to avoid the premature convergence of traditional particle swarm optimization (PSO), the randomly two-point crossover operator and randomly one-point mutation operator of the genetic algorithm (GA) was introduced. It could effectively improve the diversity of the population in the process of evolution. Finally, a cost-driven strategy for the deadline-constrained workflow was designed. It both considered the data transfer cost and the computing cost. Experimental results show that the ADPSOGA has better performance in terms of deadline and cost reducing in the fluctuant environment.

Key words: cloud computing, deadline constraint, workflow scheduling, fluctuation

收稿日期：2017-05-24；**修回日期**：2018-01-01
通信作者：郭文忠，fzugwz@163.com
基金项目：国家重点研发计划基金资助项目（No.2017YFB1002000）；国家自然科学基金资助项目（No.61672159）；福建省科技创新平台计划基金资助项目（No.2009J1007, No.2014H2005）；海峡政务大数据应用省级协同创新中心基金资助项目
Foundation Items: The National Key R&D Program of China (No.2017YFB1002000), The National Natural Science Foundation of China (No.61672159), The Technology Innovation Platform Project of Fujian Province (No.2009J1007, No.2014H2005), The Foundation of The Fujian Collaborative Innovation Center for Big Data Application in Governments

1　引言

随着云计算技术[1]的不断发展，当前云市场上出现多个云服务提供商共存的“多云”局面[2]。云计算弹性供应虚拟资源并按需付费的性质，为处理大规模科学工作流（以下简称工作流）提供便利[3]。然而，云异构环境下的任务调度是一个 NP-hard 问题[4]，工作流自身的子任务之间存在复杂的时间依赖和数据依赖关系，且多个云服务提供商之间存在诸多差异（如要价机制、实例类型、通信带宽等），因此，需要一种合适的调度策略在尽可能满足工作流服务质量（QoS, quality of service）前提下，减少其执行代价。当前云环境下的工作流调度策略大多是在传统分布式计算环境（如网格）的工作流调度算法基础上做一些改进，未考虑云环境自身特性[3,5,6]，或一些调度策略仅考虑在静态单一云环境中，单纯追求执行时间最小化目标，未对带约束的工作流（如截止日期）的代价优化调度问题展开研究[7,8]。

关于多云环境存在波动因素的带截止日期约束工作流的代价驱动调度问题，目前国内外还未有相关研究工作。而其中最具相关性的研究工作是关于单云环境存在波动因素的基于截止日期约束的工作流调度研究[1]，该工作利用传统粒子群优化（PSO, particle swarm optimization）调度策略来处理全局的工作流子任务调度方案。受到文献[1]的启发，提出一种基于遗传算法操作的自适应离散粒子群优化算法（ADPSOGA, adaptive discrete particle swarm optimization with genetic algorithm），主要研究的问题及思路包括 3 个部分：首先，该方法考虑多云环境下更多的基本要素，如多云之间的数据通信代价、虚拟机的启停时间以及多云之间的差异化带宽通信速率，同时还考虑虚拟机的性能波动和带宽通信波动；其次，为了避免传统 PSO 存在的过早收敛问题，引入遗传算法的随机两点交叉操作和随机单点变异操作，有效提高种群进化过程中的多样性；最后，在充分考虑数据通信代价和工作流子任务计算代价的前提下，设计一种基于工作流截止日期约束的代价驱动调度策略。

2　相关工作

近年来，传统分布式环境下的工作流调度已得到广泛研究。网格环境下调度工作流的目标通常是最小化工作流执行时间。Chen 等[9]设计一种网格环境下基于蚁群优化（ACO, ant colony optimization）的工作流调度方法，其目的是优化工作流执行时间，同时满足其 QoS 需求。Cao 等[10]提出基于链表调度和组调度的工作流调度算法，该方法可以有效降低工作流执行时间，并提高网格环境下的资源利用率。Higashino 等[11]提出基于 PSO 和遗传算法（GA, genetic algorithm）的混合工作流调度策略，实验表明该方法在执行时间优化方面效果明显。另外，部分研究工作在调度策略中考虑工作流执行代价因素。苑迎春等[12]基于逆向分层思想，将截止日期约束转化为层截止约束，在满足约束前提下大幅度降低工作流执行代价。Abrishami 等[13]提出网格环境下基于局部关键路径（PCP, partial critical path）的代价驱动工作流调度策略，实验表明该方法能够在满足截止日期约束前提下降低工作流执行代价。Khajemohammadi 等[14]设计一种基于遗传算法的多目标工作流调度策略，通过分层工作流模型达到同时优化执行时间和执行代价的目标。

上述方法为网格环境下工作流调度的机遇和挑战提供宝贵经验。然而，云计算环境和网格环境在资源供应和资源要价机制上存在巨大差异[1]。沈虹等[15]提出云环境下带准备时间和截止日期约束的代价驱动工作流调度算法，实验表明该算法可以在合理的 CPU 时间内减少工作流执行代价。Malawski 等[3]提出调度工作流组的动态和静态算法，其目的是在满足预算和截止日期双约束的前提下提高工作流组的完成率。该工作提到虚拟机需要一定时间来启动工作，然而其仅考虑一种虚拟机类型，不符合现实云环境。肖鹏等[16]在经典的 CPOP 算法和 HEFT 算法基础上，引入启发式策略并设计 2 种工作流能耗感知调度算法 CPOP-MECP 和 HEFT-MECP，目的是降低工作流子任务之间的数据访问能耗开销。Mao 等[17]提出一种动态的工作流组调度方案，基于按需付费模式优化工作流组的总执行代价，该工作考虑多种虚拟机类型，然而该方案对于代价优化的单一工作流调度并不适用。部分研究工作引入 PSO 指导云环境下的工作流调度问题，并取得良好效果。Wu 等[5]提出基于 PSO 的单一工作流调度方案，他们分别针对截止日期约束的代价优化问题和预算约束的执行时间优化问题展开研究，该工作的虚拟机类型和数量是固定的，不符合云环境弹性供应性质。Pandey 等[18]设计一种基于离散 PSO 的优化

求解单一工作流调度执行代价问题，该工作同样忽略云环境资源弹性提供的性质。以上云环境下的工作流调度工作均考虑单个云服务提供商，未对多云环境展开研究。

多云协同计算的定义由 Keahey 等[19]提出。而 Fard 等[2]提出一种同时优化执行时间和执行代价的多云环境下工作流调度策略，他们虽然考虑虚拟资源的弹性供应，但忽略虚拟资源的异质性，仅考虑一种虚拟机类型。之前研究工作中[20]提出多云环境下带截止日期约束的代价驱动工作流调度策略，该策略引入局部关键路径理论，对工作流的子任务进行局部整体分配，达到压缩数据通信、减少执行代价的目的，然而该策略并未考虑波动因素对调度结果带来的干扰，同时该工作忽略云间的数据通信代价和实例启停时间等因素。

云环境下关于科学工作流调度的相关工作分析如表 1 所示。其主要分成 6 类：设施环境、应用类型、环境波动性、目标、限制条件和方法。其中，设施环境包括单云和多云环境，应用类型包括作业、单工作流和工作流组，环境波动性包括静态和动态，目标分为代价和时间要求，限制条件包括截止日期和预算约束，方法主要包括数学规划、启发式和元启发式。表 1 中的“+”代表存在该性质，“−”代表不存在该性质。

3 问题定义

工作流调度模型框架如图 1 所示，它主要包括工作流、多云环境以及代价驱动调度器。

工作流 w 是用有向无环图 $G(V,E)$来表示，其中，V 表示包含 n 个任务的顶点集合$\{t_1,t_2,\cdots,t_n\}$，而 E

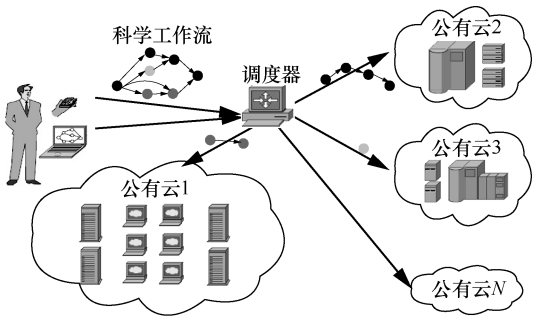


图 1 工作流调度模型

则表示任务之间数据依赖关系$\{e_{12},e_{13},\cdots,e_{ij}\}$。每条数据依赖边 $e_{ij}=(t_i,t_j)$代表任务 t_i 和任务 t_j 之间存在数据依赖关系，其中，任务 t_i 是任务 t_j 的直接先驱（父）节点，而任务 t_j 则是任务 t_i 的直接后继（子）节点。在工作流调度过程中，一个任务必须在其所有先驱节点都已被执行完毕后，才能开始执行。在某个给定的代表工作流的有向无环图中，把没有先驱节点的任务称为“入任务”，同理，把没有后继节点的任务称为“出任务”。每个工作流 w 都有一个对应的截止日期 $D(w)$，当某个调度策略能够在对应截止日期前执行完成该工作流，则称其为一种可行解。

多云环境中存在多个云服务提供商 $P=\{p,q,\cdots,r\}$，服务提供商 p 提供多种虚拟机实例类型 $S_p=\{s_{p_1},s_{p_2},\cdots,s_{p_k}\}$。每种类型的虚拟机实例都有其特定的计算能力和存储能力，假设虚拟机在执行子任务过程中有足够的存储空间来存储传输数据[17]，因此，本文主要关注虚拟机计算能力（即 CPU 数量）。子任务 t_i 在虚拟机 $vm_{p_{ij}}$ 上的估计执行时间为 $Exe\_T(t_i,vm_{p_{ij}})$，给定任务在不同类型虚拟机上的执行性价比不同。

虚拟机在初次启动时，需要一定的初始化启动

表 1 **云环境下关于科学工作流调度的相关工作分析**

相关文献	设施环境		应用类型			环境波动性		目标		限制条件		方法		
	单云	多云	作业	单工作流	工作流组	静态	动态	代价	时间	截止日期	预算	数据规划	启发式	元启发式
文献[1]	+	−	−	+	−	−	+	+	−	+	−	−	−	+
文献[2]	−	+	−	+	−	−	+	+	+	−	−	−	+	−
文献[3]	+	−	−	−	+	+	−	−	−	+	+	+	+	−
文献[5]	+	−	−	+	−	+	−	+	+	+	+	−	−	+
文献[15]	+	−	−	+	−	+	−	+	−	+	−	−	+	−
文献[17]	+	−	−	−	+	−	+	+	−	+	−	−	+	−
文献[18]	+	−	−	+	−	+	−	+	+	−	−	−	−	+
文献[20]	−	+	−	+	−	+	−	+	−	+	−	−	+	−

时间 $T_{\text{boot}}(vm_{p_{ij}})$来进行初始化配置。在工作流调度过程中，这种虚拟机初始化时间应得到重视，因为它对工作流调度方案的形成会产生重大影响[21]。同样地，当虚拟机上的所有子任务被执行完成后，对应的虚拟机并不是立刻关闭，而是要等到虚拟机上所有子任务把自身的输出数据完整通信到其后辈任务对应的虚拟机上为止[1]。多云环境下，云服务提供商 p 对于其提供的服务设定特定的要价单元时间 λ_p，每种虚拟机类型 s_{p_i} 都有对应的单位时间价格 c_{p_i}。

同一个云服务提供商的基础设施通常都集中在较小区域内，而不同云服务提供商的基础设施则相距较远[2]，因此，假设单云的云内带宽要快于不同云之间的云间带宽。云服务提供商 p 内，数据从子任务 t_i 传输到子任务 t_j 的云内通信时间为 $T_{\text{intra}}(e_{ij},p)$，而数据在云服务提供商 p 和 q 之间的通信时间为 $T_{\text{inter}}(e_{ij},p,q)$，如式(1)和式(2)所示。

$$T_{\text{intra}}(e_{ij},p)=\frac{Data(e_{ij})}{B_{\text{intra}}(p)} \tag{1}$$

$$T_{\text{inter}}(e_{ij},p,q)=\frac{Data(e_{ij})}{B_{\text{inter}}(p,q)} \tag{2}$$

其中，$Data(e_{ij})$是子任务 t_i 传输到子任务 t_j 的数据量大小，$B_{\text{intra}}(p)$是云 p 的云内带宽，而 $B_{\text{inter}}(p,q)$则是云 p 和云 q 之间的云间带宽。假设单个虚拟机上的带宽无穷大[8]，因此，当 2 个子任务被分配到同一台虚拟机上执行时，$T_{\text{intra}}(e_{ij},p)$的值为 0。

不同云之间的数据通信代价将会影响到最终的调度决策[16]，$c_{p,q}$ 表示从云 p 通信 1 GB 的数据量到云 q 所需的单价。不考虑资源监控、数据存储和负载均衡等业务产生的代价，因为这些代价与计算代价或数据通信代价相比可以忽略不计[3]。

代价调度器的目的是在满足截止日期约束的前提下，最小化工作流执行代价。调度器的输入包括工作流信息（结构、数据、依赖等）和多云环境信息（虚拟机、带宽、要价等），其输出是对应的调度方案。整个调度方案的定义为 $S=(Re,Map,T_{\text{total}},C_{\text{total}})$，其中，$Re$ 表示一组需要启用的虚拟机资源，$Re=\{vm_1,vm_2,\cdots,vm_r\}$，$Map=\{(t_i,vm_j)|t_i\in V,vm_j\in Re\}$表示工作流中子任务对应虚拟机资源 Re 的映射关系，T_{total} 表示工作流的执行完成时间，而 C_{total} 则表示工作流总执行代价。每台虚拟机都有对应的虚拟机类型 s_{p_i} 以及对应的开启时刻 $Tls(vm_i)$和关闭时刻 $Tle(vm_i)$。当子任务被调度完成后，都有一组对应的实际执行开始时间 $AST(t_i)$和实际执行完成时间 $AET(t_i)$，且“出任务”不会再产生并通信数据。因此，工作流执行完成时间 T_{total} 和对应的总执行代价 C_{total} 分别如式(3)和式(4)所示。

$$T_{\text{total}}=\max_{t_i\in W}\{AET(t_i)\} \tag{3}$$

$$C_{\text{total}}=\sum_{i=1}^{|Re|}c_{p_j}\left\lceil\frac{Tle(vm_i)-Tls(vm_i)}{\lambda_p}\right\rceil+\sum_{j=1}^{n}\sum_{k=j+1}^{n}c_{p(t_j),q(t_k)}Data(e_{jk})s_{jk} \tag{4}$$

其中，式(4)的前半部分表示虚拟机的执行代价，后半部分表示数据通信代价。子任务 t_k 是子任务 t_j 的后继节点，$p(t_j)$和 $p(t_k)$分别表示执行 t_j 和 t_k 的服务提供商。当 t_j 和 t_k 由同一个云服务提供商执行时，s_{jk} 为 0（即不产生云间数据通信），否则，s_{jk} 为 1。

基于以上相关定义，多云环境下带截止日期约束的工作流调度问题，可形式化表示为式(5)，其核心思想是在追求执行代价 C_{total} 最低的同时，使执行时间 T_{total} 小于或等于工作流截止日期 $D(w)$。

$$\begin{aligned}&\min C_{\text{total}}\\&\text{s.t.}\,T_{\text{total}}\leqslant D(w)\end{aligned} \tag{5}$$

4　ADPSOGA 调度策略

本节首先介绍基础 PSO，然后具体介绍 ADPSOGA 调度策略。

4.1　PSO

PSO 是一种基于鸟群社会行为的动物进化计算技术，它是在 1995 年由 Kennedy 等[22]提出的。粒子在 PSO 中尤为重要，每个粒子代表优化问题的候选解，它们可以在整个问题空间范围内移动。每个粒子以某种速度移动更新自己的移动方向，该速度受粒子自身情况、粒子自身最佳历史位置以及整个种群的历史最佳位置这 3 个方面影响。为了判断每个粒子在问题空间中的不同位置所产生解的优劣性，引入适应度函数来评估每个粒子的解质量。每个粒子是由它自身的位置和速度决定，它们根据周围粒子和自身的经验在问题搜索空间中不断迭代更新调整自己的位置和速度。其中，速度根据式(6)进行更新，位置根据式(7)进行更新。

$$V_i^{t+1}=w\cdot V_i^t+c_1r_1(pBest_i^t-X_i^t)+c_2r_2(gBest^t-X_i^t) \tag{6}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{7}$$

其中，t 表示当前的迭代次数，V_i^t 和 X_i^t 分别表示第 i 个粒子在第 t 次迭代时的速度和位置，通常需要定义一个最大速度 $V_{\max}$ 来限制粒子速度，使搜索结果在问题解空间内。$pBest_i^t$ 和 $gBest^t$ 分别是经过 t 次迭代后粒子 i 的自身历史最优位置和整个种群的历史最佳位置。w 是惯性权重，它决定前一次迭代速度对当前速度的影响大小，对算法的收敛性至关重要。c_1 和 c_2 是认知因子，它们体现当前粒子对自身历史最优值和种群全局历史最优值的认知学习能力。r_1 和 r_2 是范围为 0~1 的 2 个随机变量，用于加强迭代搜索过程中的随机性。

4.2 ADPSOGA

从以下 7 个部分具体阐述 ADPSOGA。

4.2.1 问题编码

提高算法搜索效率和性能，需要一种好的编码方式。编码策略的评价标准主要考虑其健全性、完备性和非冗余性 3 个基本原则[23]。

定义 1 健全性。编码空间中的某个编码粒子，必须对应问题空间中的某个潜在问题优化解。

定义 2 完备性。问题空间中的全部可行解，都能在编码空间中被相应的粒子表现出来。

定义 3 非冗余性。问题空间中的潜在解，只能和编码空间中的相关粒子一一对应。

实现同时满足以上 3 个原则的编码策略十分困难。受到文献[1]的启发，采用云提供商—实例类型—具体实例的嵌套方式编码工作流调度问题。一个粒子代表多云环境下工作流的一个调度方案，粒子 i 在 t 时刻的位置 X_i^t 如式(8)所示。

$$X_i^t = (x_{i1}^t, x_{i2}^t, \cdots, x_{in}^t) \tag{8}$$

其中，x_{ik}^t (k=1,2,⋯,n)表示第 k 个子任务在 t 时刻的分配位置，如式(9)所示。

$$x_{ik}^t = (p, s_{p_j}, vm_{p_{jr}})_{ik}^t \tag{9}$$

其中，$(p,s_{p_j},vm_{p_{jr}})$表示该子任务被分配到云 p 中实例类型是 s_{p_j} 的第 r 个具体实例上。粒子上的每一个节点位被嵌套划分成 3 个小分位，分别表示云服务提供商、实例类型和具体实例，因此，编码空间大小是子任务数量的 3 倍。初始化种群时，粒子的节点小分位各自随机初始化为 0 到其对应最大值之间的整数值。图 2 展示调度包含 8 个子任务工作流的粒子编码策略，其中假设多云环境包含 3 个云服务提供商，且每个云服务提供商均提供 8 种实例类型。因此，p 坐标值是从 0~2，s_{p_j} 坐标值是从 0~7。由图 2 可知，子任务 t_1 被分配给云第 0 个云中类型是 s_{00} 的虚拟机 vm_{000}。

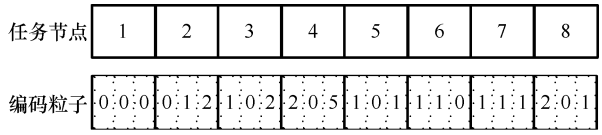


图 2 粒子编码

性质 1 云提供商—实例类型—具体实例的嵌套离散编码方式满足编码完备性和非冗余性基本原则，但不满足编码的健全性基本原则。

粒子的每个编码分位代表对应子任务的分配位置，满足完备性原则。不同编码粒子分别代表不同调度方案，问题空间的某个可行解只与编码空间的一个编码粒子对应，因此，满足非冗余性原则。部分粒子存在对应问题空间中不可行解情况，例如，调度方案超过工作流截止日期，因此，不满足健全性原则。

4.2.2 初始化资源池

多云环境资源的弹性供应模式，导致算法无法得到初始资源集合。对于 PSO 而言，初始化资源池的大小将决定搜索空间的范围，对算法复杂度以及工作流执行性能起着关键作用。当初始化资源池太小时，会出现本可以在截止日期前完成的工作流，由于资源的缺乏而无法及时完成。当初始化资源池太大时，PSO 编码的潜在解过于庞大，使算法无法及时收敛。一种简单可行的初始化资源分配方案，是为每个子任务分配多云环境中所有类别的虚拟机各一台，这样可以保证搜索空间的多样性和完整性。然而，这种方案的初始化资源池 R_{intial} 的大小为 $n{\cdot}Num_{type(vm)}$，搜索空间比较大，增加算法复杂度。其中，n 为工作流 w 中的子任务数量，$Num_{type(vm)}$为所有云服务提供商的实例类型数量总和，其定义为

$$Num_{type(vm)} = \sum_{p}^{P} Num_{vm(p)} \tag{10}$$

其中，$Num_{vm(p)}$为云服务提供商 p 所提供的实例类型数量。

为进一步压缩搜索空间，同时保持原有潜在解的多样性，设计初始化资源池 R_{intial} 的大小是 $|S_{par(w)}|Num_{type(vm)}$，其中，$S_{par(w)}$是工作流 w 中的最大可并行子任务集合。由于除了 $S_{par(w)}$中的子任务外，

其他子任务都会和 $S_{par(w)}$中的子任务存在直接和间接的依赖关系，所以该初始化资源策略可以保证每个子任务都有选择一种类型实例的机会，从而保证潜在解的多样性，同时降低搜索空间。

4.2.3 适应度函数

粒子的适应度函数是用来评价 2 个相比较粒子的优劣性，通常较小的适应度函数值对应的粒子较优。由于前期的粒子编码策略不满足健全性原则，即会出现工作流的执行时间超过对应截止日期，所以需要对可行解和超过截止日期的不可行解的适应度函数区分定义。判断 2 个粒子优劣的适应度函数分 3 种不同情况定义。

情况 1 一个粒子是可行解，另一个粒子是不可行解。毫无争议地选择可行解，其适应度值定义为

$$fitness=\begin{cases}0, T_{\text{total}(X_i)} \leqslant D(w)\\ 1, \text{其他}\end{cases} \tag{11}$$

情况 2 2 个粒子都是可行解。选择执行代价较低的粒子，其适应度值定义为

$$fitness = C_{\text{total}(X_i)} \tag{12}$$

情况 3 2 个粒子都是不可行解。选择执行时间较小的粒子，因为该粒子进化后更有可能变为可行解。其适应度值定义为

$$fitness = T_{\text{total}(X_i)} \tag{13}$$

4.2.4 粒子的更新策略

如式(6)所示，PSO 包括 3 个核心部分：惯性部分、个体认知部分和社会认知部分。为克服传统 PSO 存在的过早收敛缺陷，ADPSOGA 引入遗传算法的变异和交叉操作，对式(6)中相应部分进行更新操作。粒子 i 在 t 时刻的更新方式为

$$X_i^t = c_2 \oplus C_g(c_1 \oplus C_p(w \oplus M_u(X_i^{t-1}), pBest_i^{t-1}), gBest^{t-1}) \tag{14}$$

其中，$M_u()$表示变异操作，$C_g()$和 $C_p()$表示交叉操作。

式(6)中的惯性部分结合遗传算法中变异操作思想，惯性部分的更新方式为

$$A_i^t = w \oplus M_u(X_i^{t-1}) = \begin{cases} M_u(X_i^{t-1}), & r_1 < w \\ X_i^{t-1}, & \text{其他} \end{cases} \tag{15}$$

其中，r_1 为 0~1 的随机数。$M_u()$随机选取粒子中的一个分位，不规律改变其分位值，且新值必须都在对应的阈值内。

图 3 为对图 2 编码粒子的变异操作，随机选择粒子的一个分位 mp1，mp1 位置上的值由(0,1,2)更新为(1,2,0)，该变异符合调度准则。

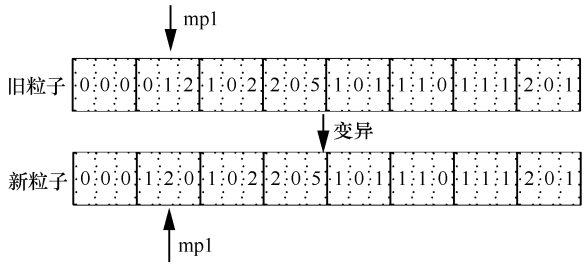


图 3　惯性部分的变异操作

式(6)中的个体认知和社会认知部分结合遗传算法交叉操作思想，其更新结果分别如式(16)和式(17)所示。

$$B_i^t = c_1 \oplus C_p(A_i^t, pBest^{t-1}) = \begin{cases} C_p(A_i^t, pBest^{t-1}), & r_2 < c_1 \\ A_i^t, & \text{其他} \end{cases} \tag{16}$$

$$C_i^t = c_2 \oplus C_g(B_i^t, gBest^{t-1}) = \begin{cases} C_g(B_i^t, gBest^{t-1}), & r_3 < c_2 \\ B_i^t, & \text{其他} \end{cases} \tag{17}$$

其中，r_2 和 r_3 为 0~1 的随机数，$C_p()$（或 $C_g()$）随机选择粒子的 2 个分位，将分位之间的数值与对应 *pBest*（或 *gBest*）分位之间的数值进行交叉。

图 4 为个人（或社会）认知部分的交叉操作，随机产生 2 个交叉位置（即 cp1 和 cp2），将粒子 cp1 和 cp2 位置之间的值替换为 *pBest*（或 *gBest*）在该区间的值。

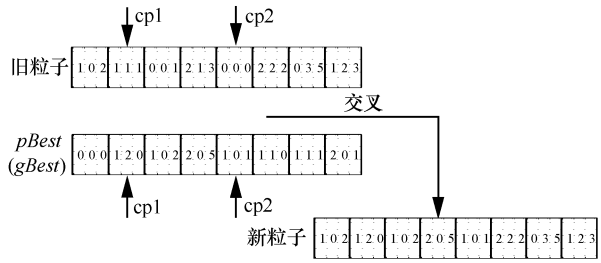


图 4　个人（或社会）认知部分的交叉操作

4.2.5 粒子到调度结果的映射

编码粒子到调度结果的映射如算法 1 所示。

算法 1　编码粒子到调度结果的映射

procedure Schedule_Generation(w, R_{intial}, X)

1) 初始化: $Re \leftarrow null$, $Map \leftarrow null$, $T_{\text{total}} \leftarrow 0$, $C_{\text{total}} \leftarrow 0$

2) 计算 $Exe\_T[|w|\times|R_{intial}|]$

3) 计算 $T_{intra}[|w|\times|w|]$, $T_{inter}[|w|\times|w|, |P|\times|P|]$

4) for $i=0$ to $i=|w|-1$

5) $t_i=w[i]$, $r_{X(i)}=R_{intial}[X(i)]$

6) if t_i 为真实入任务 then

7) if $r_{X(i)}$未启动 then

8) 启动 $r_{X(i)}$, $LETr_{X(i)}=T_{boot}(r_{X(i)})$, $Tls(r_{X(i)})=LETr_{X(i)}-T_{boot}(r_{X(i)})$

9) end if

10) $STt_i=LETr_{X(i)}$

11) else

12) 调用 max_$Parents(t_i)$过程, max$T$ = max_$Parents(t_i)$

13) if $r_{X(i)}$未启动 then

14) 启动 $r_{X(i)}$, $LETr_{X(i)}=\max(\max T, T_{boot}(r_{X(i)}))$, $Tls(r_{X(i)})=LETr_{X(i)}-T_{boot}(r_{X(i)})$

15) end if

16) $STt_i=\max(\max T, LET\ r_{X(i)})$

17) end if

18) $exe=Exe\_T[i][X(i)]$

19) for each child t_c of t_i do

20) if t_c 被调度到与 t_i 同一个云中，但不同的虚拟机上 then

21) $transfer+=T_{intra}[i][c]$

22) else if t_c 被调度到与 t_i 不同的云中 then

23) $transfer+=T_{inter}[i_p][c_q]$

24) end if

25) end for

26) $ETt_i=STt_i+exe+transfer$

27) $Map=Map\cup(t_i, r_{X(i)}, STt_i, ETt_i)$

28) if $r_{X(i)}\notin Re$ then

29) $Re=Re\cup\{r_{X(i)}\}$

30) end if

31) $LETr_{X(i)}=LETr_{X(i)}+exe+transfer$

32) end for

33)根据式(3)和式(4)分别计算 T_{total} 和 C_{total}

34)输出调度方案及其对应结果 $S=(Re, Map, T_{total}, C_{total})$

算法 1 的输入包括工作流 w、初始化资源池 R_{intial} 和编码粒子 X。首先，对调度方案 S=(Re, Map, T_{total}, C_{total})的四元素初始化（第 1)行）。初始化后，计算每个子任务对应不同类型实例的估计执行时间矩阵 $Exe\_T[|w|\times|R_{intial}|]$，矩阵中的元素 $Exe\_T[i][j]$ 表示子任务 t_i 在虚拟机 mv_j 上的估计执行时间（第 2) 行）。计算子任务之间的数据量在单云和多云之间的估计通信时间，$T_{intra}[i][j]$表示单云中子任务 t_i 产生的数据量通信到子任务 t_j 所需的估计时间，$T_{inter}[i][j][p][q]$表示子任务 t_i 产生的数据量从云 p 通信到云 q 的子任务 t_j 所需的估计时间（第 3)行）。

经过以上操作，目前已得到从编码粒子得到候选解的全部信息。逐步扫描粒子 X 的各个分位，生成对应的 Re 和 Map 集合。基于“问题编码”，粒子的编码分位对应子任务，分位的值对应实例资源，因此，确定子任务 t_i 被分配到实例 $r_{X(i)}$上（第 5) 行）。第 6)~17)行是计算子任务 t_i 的估计开始时间 STt_i，这里有 2 种情况。

1) 子任务 t_i 是真实入任务，即其没有直接先驱子任务。当虚拟机 $r_{X(i)}$可用时，子任务 t_i 立刻开始执行，其估计开始时间 STt_i 为虚拟机 $r_{X(i)}$的已租赁时间 $LETr_{X(i)}$。另外，需要判断虚拟机 $r_{X(i)}$是否已开启，如果未开启，则需要启动虚拟机，虚拟机的已租赁时间 $LETr_{X(i)}$即虚拟机的初始化时间 $T_{boot}(r_{X(i)})$。

2) 子任务 t_i 不是入任务，即其有一个或多个父任务。子任务 t_i 不但需要等待资源空闲时才可执行，还得等待其所有父任务执行完成，并将产生数据通信到虚拟机 $r_{X(i)}$上才可执行。调用 max_$Parents(t_i)$ 过程计算子任务 t_i 等待时间和数据通信代价，同时考虑虚拟机 $r_{X(i)}$是否已被启动。

计算完子任务 t_i 的估计开始时间 STt_i，需要根据其在虚拟机上的估计执行时间和数据通信时间，来计算子任务 t_i 的估计结束时间 ETt_i（第 18)~26)行）。对于数据通信时间的计算，需要根据其后辈子任务 t_c 是否与子任务 t_i 分配在同一个云中来确定，这里有 3 种情况。

1) t_c 与 t_i 在同一台虚拟机上执行，则通信时间 $transfer$ 为 0。

2) t_c 与 t_i 在同一个云中但不同虚拟机上执行，则通信时间 $transfer$ 为 $T_{intra}[i][c]$。

3) t_c 与 t_i 分别在不同云（如云 p 和云 q）上执行，则通信时间 $transfer$ 为 $T_{inter}[i_p][c_q]$。

子任务 t_i 调度到虚拟机 $r_{X(i)}$，其开始时间 STt_i 和结束时间 ETt_i 等四元素的映射关系添加到 Map 集合中（第 27)行）。随后判断虚拟机 $r_{X(i)}$是否已被添加到租赁资源 Re 中，如果未被添加，则

进行添加（第 28)~30)行）。虚拟机 $r_{X(i)}$的最新已租赁时间，等于子任务 t_i 的估计完成时间（第 31)行）。最后，根据式(3)和式(4)分别计算工作流的总执行时间和总执行代价。输出编码粒子对应的调度方案 S。子任务的等待时间和通信代价如算法 2 所示。

算法 2　子任务的等待时间和通信代价

procedure max_*Parents*(t_i)

1) 初始化：$T_{\text{wait}} \leftarrow 0, C_{\text{tranfer}} \leftarrow 0$

2) for each parent t_p of t_i do

3) if t_p 被调度到与 t_i 同一个云中，但不同的虚拟机上 then

4) $T_{\text{wait}} = \max(T_{\text{wait}}, T_{\text{intra}}[p][i])$

5) else if t_p 被调度到与 t_i 不同的云中 then

6) $T_{\text{wait}} = \max(T_{\text{wait}}, T_{\text{inter}}[p_q][i_p])$

7) $C_{\text{tranfer}} \mathrel{+}= Data(e_{p_i})c_{q,p}$

8) end if

9) end for

10) 输出最长等待时间 T_{wait}，通信代价 C_{tranfer}

首先，初始化等待时间 T_{wait} 和数据通信代价 C_{tranfer}（第 1)行）。子任务 t_i 等待时间等于其所有父任务中最大的数据通信时间（第 3)~6)行）。当子任务 t_i 和其父任务 t_p 分配到不同云中时，才考虑数据通信代价（第 7)行）。

4.2.6　参数设置

式(6)的惯性权重因子 w 能够决定 PSO 的收敛性和搜索能力[4]。当 w 较小时，算法具有较强的局部搜索能力；否则，算法具有较强的全局搜索能力。在算法执行初期，更注重问题空间搜索的多样性和粒子全局搜索能力，随着搜索深入，后期则更加注重局部搜索方面的能力。因此，惯性权重因子 w 的权值应随着算法迭代次数的增多而逐渐减少。式(18)是经典的惯性权重因子调整策略。

$$w = w_{\max} - iters_{\text{cur}} \times \frac{w_{\max} - w_{\min}}{iters_{\max}} \tag{18}$$

其中，$w_{\max}$ 和 $w_{\min}$ 分别是 w 初始化时设定的最大值和最小值，$iters_{\text{cur}}$ 和 $iters_{\max}$ 分别表示当前算法迭代次数和初始化设定的最大迭代次数。

以上的经典惯性权重因子调整策略，w 的变化仅和迭代次数有关，不能很好地满足实际问题的非线性和复杂多变特性。惯性权重因子 w 的权值大小应该随着种群粒子的进化而不断演变，因此，构建一种根据当前种群粒子的优劣而自适应调整的惯性权重因子调整策略。如式(19)所示，该策略基于当前粒子与全局历史最优粒子之间的差异程度来调整惯性权重因子大小。

$$d(X^{t-1}) = \frac{div(X^{t-1}, gBest^{t-1})}{3T} \tag{19}$$

其中，$div(X^{t-1}, gBest^{t-1})$表示粒子 X^{t-1} 和全局历史最优粒子 $gBest^{t-1}$ 之间的不同分位的位数大小，T 是工作流中子任务的个数大小。当 $div(X^{t-1})$值较小时，表示粒子 X^{t-1} 和 $gBest^{t-1}$ 之间差异程度较小，所以应该减小 w 的权值，以保证粒子可在小范围内更好搜索，找到优化解；否则，应该增加 w 的权值，使粒子的搜索空间变大，以便更快找到优化解空间。因此，惯性权重因子 w 的权值计算式更新为

$$w = w_{\max} - (w_{\max} - w_{\min}) \times \exp\left(\frac{d(X_i^{t-1})}{d(X_i^{t-1}) - 1.01}\right) \tag{20}$$

另外，算法的 2 个认知因子 c_1 和 c_2 采用线性增减策略进行设置[24]。其更新方式如式(21)和式(22)所示。

$$c_1 = c_1\_\text{start} - \frac{c_1\_\text{start} - c_1\_\text{end}}{iters_{\max}} \times iters_{\text{cur}} \tag{21}$$

$$c_2 = c_2\_\text{start} - \frac{c_2\_\text{start} - c_2\_\text{end}}{iters_{\max}} \times iters_{\text{cur}} \tag{22}$$

其中，c_1_start 和 c_2_start 分别表示参数 c_1 和 c_2 迭代的初始值，c_1_end 和 c_2_end 分别表示参数 c_1 和 c_2 迭代的最终值。

4.2.7　算法流程

ADPSOGA 流程如图 5 所示，具体步骤如下。

步骤 1　初始化 ADPSOGA 中种群大小、最大迭代次数、惯性权重因子和认知因子等参数的数值，随机生成初始种群。

步骤 2　根据第 4.2.5 节的粒子映射策略以及式(11)~式(13)计算各个粒子不同情况下的适应度值，从中选择适应度值最小的粒子作为种群全局最优粒子，将第一代中每个粒子设置为其自身历史最优粒子。

步骤 3　根据粒子更新式(14)~式(17)更新粒子。

步骤 4　重新计算每个粒子的适应度值，若当前粒子的适应度值小于其自身历史最优值，则将新粒子更新为其自身历史最优粒子。

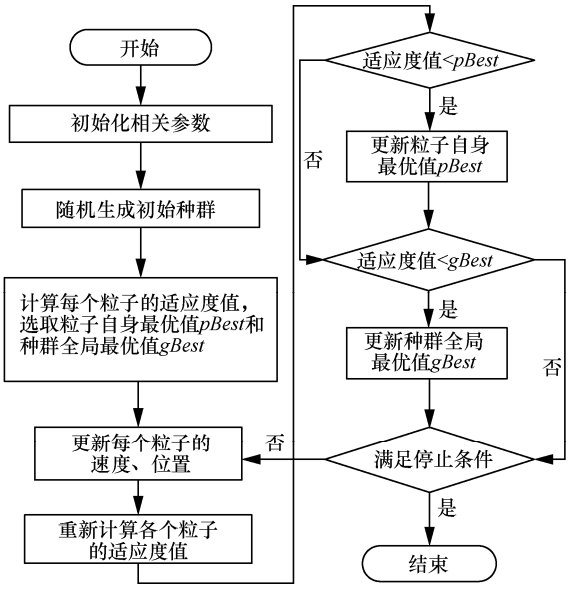


图 5　ADPSOGA 流程

步骤 5　若当前粒子的适应度值小于种群全局最优粒子的适应度值，则将该粒子更新为种群全局最优粒子。

步骤 6　检查是否满足算法终止条件，如果满足，则算法终止；反之，转到步骤 3。

5　实验仿真与结果

本文的模拟实验是在 64 位 Windows 7 系统环境下，其配置是 8 GB 内存和 2.30 GHz 的 i7 Intel 处理器。基于文献[24]，ADPSOGA 的参数设置如下。种群大小为 100，最大迭代次数为 1 000，c_1_start = 0.9，c_1_ end = 0.2，c_2_start =0.4，c_2_ end =0.9。

5.1　实验设置

工作流测试样例采用 Bharathi 等[25]研究的来自 5 个不同科学领域的工作流：天文学（Montage）、地震科学（CyberShake）、生物基因学（Epigenomics）、重力物理学（LIGO）以及生物信息学（SIPHT）。这些工作流具有不同结构，它们的信息（如子任务大小、数据通信量和依赖关系等）被存储在 xml 格式的文件中。对于每种工作流，选取其中 3 个数量级：小型（约 30 个任务），偏小型（约 50 个任务）和中型（约 100 个任务）。由于子任务在具体实例上的执行时间存在波动，因此，假设工作流子任务的大小呈正态分布，其变化率区间为(−10%,10%)[1]。

假设多云环境下存在 3 个云服务提供商：EC2 (C1)、Rackspace (C2)和 GoGrid (C3)。为了增加子任务分配的多样性，假设每个云服务提供商均有 8 个不同实例类型，每个实例类型拥有特定的任务执行速度和单位时间价格。实例的处理能力和它的单价近似成正比，因此，假设在 C1（C2 或 C3）中，处理最快的实例速度大约是最慢的 5（8 或 10）倍，单价同样大致是最慢的 5（8 或 10）倍[20]。每个云中最慢的实例类型单价相同，每个实例类型对应不同子任务的处理性价比存在差异，假设每$\frac{1}{3}$数量的子任务在其中一个云中的处理性价比高于另外 2 个云。基于实例处理性能波动[3]，假设实例处理性能的变化满足均值减少 12%、最大减少 24%、标准差为 10%的正态分布。

基于 Amazon 服务带宽[20]，假设单云的云内带宽为 20 Mbit/s。不同云之间的通信代价如表 2 所示，这些数值是通过带宽测试工具 iperf 测试得到，主要针对 Amazon EC2 的悉尼区域(C1)、Rackspace(C2)和 GoGrid(C3)。

表 2　　不同云之间的通信代价

通信代价	C1→C2	C1→C3	C2→C3
均值/(Mbit·s^{-1})	1.33	3.25	5.43
时间/(s·Gbit^{-1})	770	315	189
均方差	29.50%	18.23%	33.26%

假设带宽大小变化率满足最大减少 19%、均值减少 9.5%、标准差为 5%的正态分布[1]。另外，假设虚拟机的初始化时间是 97 s，要价区间是 1 h[21]。表 3 是不同云之间的数据通信价格，该表数值来自各自云服务提供商的官方网站。

表 3　不同云之间的数据通信价格（美元 · GB^{-1}）

数据量	C1→(C2,C3)	C2→(C1,C3)	C3→(C2,C3)
0~1 GB	0.00	0.12	0.00
1 GB~1 TB	0.19	0.12	0.12
1~10 TB	0.19	0.12	0.11
10~50 TB	0.15	0.10	0.10

每个工作流都有一个对应的截止日期，以此来测试调度策略性能。选择 5 个不同的截止日期 $D_i(w)$ 来测试本文涉及的调度方法。

$$D_i(w) = r_i MinE(w), i = \{1,\cdots,5\} \tag{23}$$

其中，$MinE(w)$是用 HEFT 算法[26]调度工作流 w 的执行时间，r_i 则是从集合{1.5, 2, 5, 8, 15}中依次取值。

5.2　对比算法

为了验证 ADPSOGA 调度策略的有效性，通过改写文献[20]的静态 MCPCPP 调度策略和文献[1]的 WPSO 调度策略，使它们适应多云环境，并作为本文的对比算法。

MCPCPP 调度策略是在静态多云环境下，在满足工作流截止日期前提下，追求执行代价最低。该方法通过合并包含共同“割边”的相邻子任务，降低算法复杂度并减少数据通信时间；分配子截止日期到局部关键路径上，并对局部关键路径进行整体调度，压缩数据通信时间和代价；通过调度局部关键路径到“最适合”实例上，进一步减少工作流执行代价。然而，该策略未考虑多云之间的数据通信代价，也忽视了实例启停时间对调度结果的影响。因此，在对比试验中，MCPCPP 调度策略同时考虑云间的数据通信代价和实例启停时间。

WPSO 调度策略是在单云环境下，在满足工作流截止日期前提下，追求执行代价最低。该方法利用传统 PSO 的连续编码方式，其子任务和实例的映射关系是通过选取编码粒子的实数取整部分来实现，粒子更新方式是基于式(6)的传统方式。为了和本文算法形成对比，WPSO 经过适当改写就可以适合多云环境的工作流调度。WPSO 的总体框架和文献[1]相同，但在考虑初始资源池时，需要考虑多云环境下的所有实例类型，另外，还需要考虑多云之间数据传输的通信代价，其惯性权重因子 w 和认知因子 c_1、c_2 的设置依然参照文献[1]。

5.3　结果评价

为了测试 ADPSOGA、MCPCPP 和 WPSO 策略在多云环境存在波动因素的工作流调度性能，对各种波动因素的生成函数随机取值，并对每种因素各进行 100 组实验，综合深入考察各个调度策略性能。

另外，由于测试的 5 个类型工作流的结构和子任务大小存在差异，因此，需要对工作流执行代价的结果进行归一化处理。引入工作流标准执行代价 $NWC(w)$，如式(24)所示。

$$NWC(w)=\frac{TEC(w)}{CSC(w)} \tag{24}$$

其中，$TEC(w)$表示采用本文任何一种调度策略产生的工作流执行代价，而 $CSC(w)$则表示利用不考虑截止日期约束的最廉价调度策略[6]所产生的工作流执行代价。

表 4 和表 5 分别表示在 100 组重复测试中，ADPSOGA 和 WPSO 这 2 种基于 PSO 的调度策略在 $D_1(w)$和 $D_5(w)$不同截止日期下，得到各自全局历史最优值 *gBest* 的平均迭代次数。由表 4 和表 5 可知，随着工作流子任务数量的增加，2 种算法得到各自全局历史最优值 *gBest* 的平均迭代次数均明显增多。这主要是因为子任务数量的增加，使基于编码

表 4　ADPSOGA 和 WPSO 在 $D_1(w)$达到 *gBest* 的迭代次数

算法	小型		偏小型		中型	
	ADPSOGA	WPSO	ADPSOGA	WPSO	ADPSOGA	WPSO
CyberShake	369	103	688	159	998	214
Epigenomics	392	273	934	348	997	391
LIGO	689	239	605	212	998	331
Montage	612	174	916	210	998	296
SIPHT	590	169	511	213	877	269

表 5　ADPSOGA 和 WPSO 在 $D_5(w)$达到 *gBest* 的迭代次数

算法	小型		偏小型		中型	
	ADPSOGA	WPSO	ADPSOGA	WPSO	ADPSOGA	WPSO
CyberShake	426	252	612	161	940	319
Epigenomics	457	260	585	322	861	308
LIGO	332	184	542	249	896	376
Montage	475	267	762	257	710	222
SIPHT	695	134	600	320	936	368

策略的粒子维度增加，粒子更新过程中也产生更多的新粒子和更大种群规模，从而导致出现更多的候选解粒子，并扩大问题域的搜索空间，使搜索到最佳解的迭代次数增多。随着截止日期约束的变化，2 种不同算法达到 *gBest* 的平均迭代次数并未发生明显改变，反而是不同调度策略之间在同等环境下的迭代次数存在明显区别，这说明截止日期约束对迭代次数的影响并不显著。同等条件下，ADPSOGA 调度策略的平均迭代次数明显高于 WPSO 调度策略。ADPSOGA 调度策略的自适应惯性权重因子和渐进式增减认知因子的设置，导致其编码粒子更具有多样性，在算法执行初期更加注重问题空间的广泛性搜索，而后期则倾向于对算法收敛性的关注，从而产生更佳的全局历史最优值。而 WPSO 调度策略主要采用连续的编码和更新方式来解决工作流调度这个离散问题，这种编码方式容易陷入局部最优，从而过早收敛，导致产生的优化结果并非最优结果。

图 6 是多云环境存在波动因素的 5 种中型工作流在 3 种不同调度策略（即 MCPCPP、ADPSOGA 和 WPSO）中的完成率。该完成率是指在相应的 100 组测试中，工作流完成时间达到对应截止日期约束要求的比例大小。从图 6 中的 5 个子图可以发现，随着截止日期的增大，基于 PSO 的 ADPSOGA 和 WPSO 这 2 种调度策略的完成率都呈上升趋势。特别在 $D_5(w)$阶段，除了 CyberShake 工作流，2 种调度策略的完成率都达到了 100%。MCPCPP 调度策略的完成率明显低于其他 2 种调度策略，其完成率变化规律与截止日期之间不存在明显的相关性，即其完成率不随着截止日期约束的宽松而得到明显提升。在 Montage、Epigenomics 和 LIGO 工作流的测试样例中，MCPCPP 调度策略在不同的截止日期约束下的完成率均为 0，主要是因为 MCPCPP 调度策略过于强调压缩子任务之间的数据通信代价，这种操作方式对动态因素的抗干扰能力较弱。

图 6(a)是 3 种调度策略在不同截止日期约束下的 Montage 工作流完成率。ADPSOGA 调度策略在 $D_1(w)$和 $D_2(w)$阶段的完成率明显优于 WPSO 调度策略。WPSO 调度策略在 $D_1(w)$阶段的完成率还不到 30%，但在 $D_3(w)$阶段就达到近 80%，稍逊于 ADPSOGA 调度策略。在 $D_4(w)$和 $D_5(w)$阶段，ADPSOGA 和 WPSO 调度策略均达到 100%的工作流完成率。图 6(b)则是 3 种调度策略的 CyberShake 工作流完成率。在 $D_1(w)$阶段，WPSO 调度策略的完成率超过 20%，排在 3 种策略之首，其次是 MCPCPP 调度策略，完成率最低的是 ADPSOGA 调

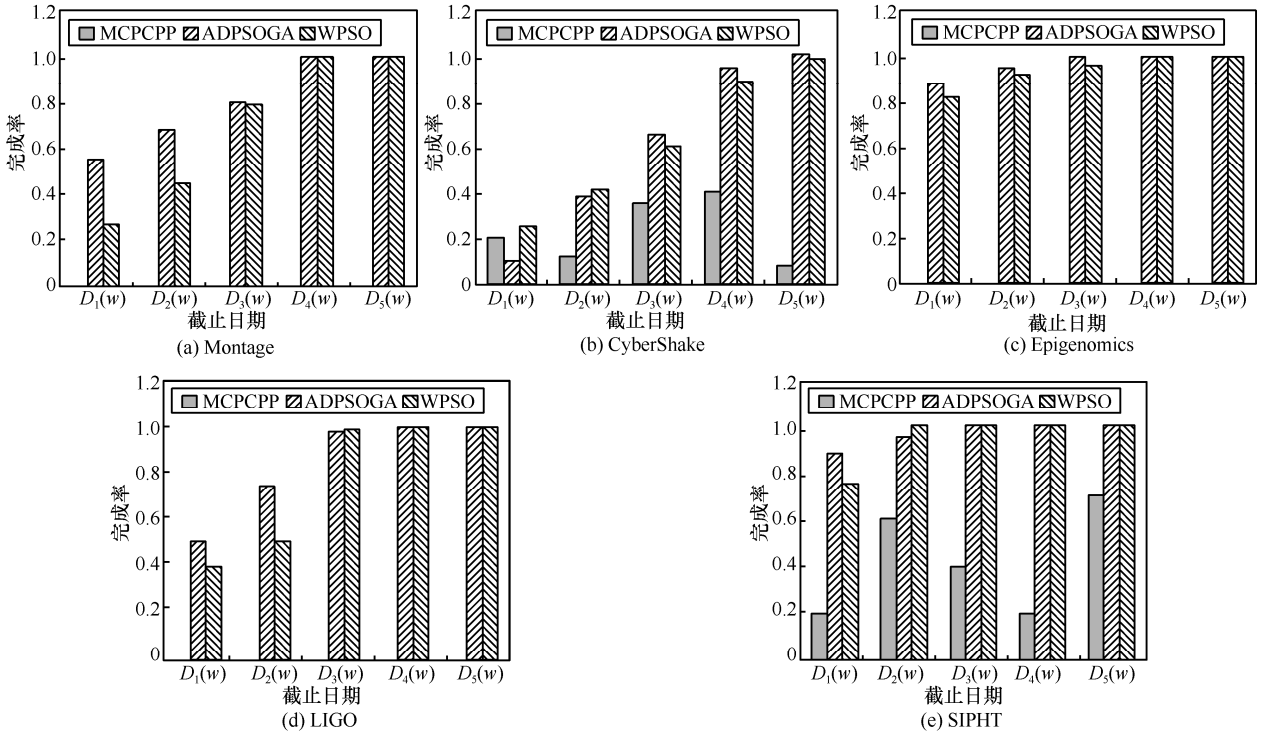


图 6 多云环境存在波动因素的 5 种中型工作流在不同调度策略的完成率

度策略。但随着截止日期约束不断宽松，如在 $D_3(w)$ 阶段之后，ADPSOGA 调度策略就处于优势地位，其完成率均超过另外 2 种调度策略。在 $D_5(w)$阶段，ADPSOGA 和 WPSO 这 2 种调度策略的工作流完成率均接近 100%，但 MCPCPP 调度策略却只有不足 10%的完成率。这主要还是归因于 MCPCPP 过分压缩子任务间的数据通信。图 6(c)是 3 种调度策略的 Epigenomics 工作流完成率。与图 6(a)和图 6(d)相似，MCPCPP 调度策略在所有截止日期约束下的完成率均为 0，除了过分压缩数据的原因外，还主要跟这 3 种工作流的结构和对应子任务大小有关。这 3 种工作流均包含较多只占实例小部分单位执行时间的子任务，通过 MCPCPP 调度策略的整合压缩，在同一个路径上的这些子任务被统一分配给特定执行实例。当环境波动时，这些子任务路径的实际执行时间往往超过估计值，所以使工作流无法在截止日期前完成。另外，各阶段的 ADPSOGA 和 WPSO 调度策略的完成率相当接近，均超过 85%。图 6(d)是 3 种调度策略的 LIGO 工作流完成率。ADPSOGA 在 $D_1(w)$和 $D_2(w)$阶段的完成率明显优于 WPSO。同时在 $D_3(w)$之后，它们二者的完成率基本相同，均接近 100%。图 6(e)是 3 种调度策略的 SIPHT 工作流完成率。在 $D_1(w)$阶段，ADPSOGA 调度策略的完成率最高，达到 85%。在 $D_2(w)$阶段，ADPSOGA 完成率稍逊于 WSPO。在后续阶段，基于 PSO 的 2 种调度策略的完成率均达到 100%。

综上，MCPCPP 调度策略在多云环境存在波动因素的适应性最差。ADPSOGA 和 WPSO 调度策略在截止日期约束宽松的情况下，均能获得良好的完成率。在截止日期约束严格的情况下，ADPSOGA 调度策略相比于其他 2 种策略，性能更加优越。这主要是因为 ADPSOGA 的编码策略和更新方式，使其获取更加多样的进化种群，适应动态变化环境。

图 7 是 3 种不同调度策略（即 MCPCPP、ADPSOGA 和 WPSO）在对应 3 种不同截止日期（即 $D_1(w)$、$D_3(w)$和 $D_5(w)$）约束下的工作流平均标准执行代价和平均执行时间（每个结果针对 100 组重复测试样例）。每个子图中的 3 条线，分别表示对应的 3 个截止日期参考线，目的是为了确定相关工作流平均执行时间是否超过对应截止日期。同一图中同时展示执行代价和执行时间的目的是为了寻找执行时间满足截止日期约束，同时其执行代价最低的有效调度策略。

图 7(a)是 Montage 工作流的平均标准执行代价和平均执行时间。MCPCPP 调度策略在 3 个截止日期约束下的平均标准执行代价 $NEC(w)$都是最低的，但其对应的平均执行时间均远超过截止日期参考线，因此，该调度策略在 3 个截止日期约束下均不是有效调度方法。在 $D_1(w)$、$D_3(w)$和 $D_5(w)$阶段，ADPSOGA 和 WPSO 调度策略均满足截止日期约束，且 ADPSOGA 调度策略的 $NEC(w)$最低。尤其在 $D_3(w)$阶段，ADPSOGA 调度策略的优势显得尤为明显。相对于其他工作流而言，3 种调度策略在执行 Montage 工作流时，产生更大的平均标准执行代价 $NEC(w)$，其大约是在同等条件下调度 SIPHT 工作流的 20 倍。造成巨大差距的原因主要是因为不同类型工作流的自身结构和子任务大小存在差异。对于 Montage 工作流而言，其第 2 层中含有许多子任务，该层子任务在最快虚拟机上的执行时间都不超过 15 s。在要价区间是 1 h 的实验环境下，这些子任务都只占用虚拟机整个要价区间的很小一部分。为了满足截止日期约束，需要启动更多的高速虚拟机来处理 Montage 工作流相关子任务，造成资源浪费、对应标准执行代价过高。MCPCPP 调度策略往往将第 2 层子任务压缩到少数几台高速虚拟机上执行。当存在波动因素时，其结果往往是执行时间超过对应截止日期约束。

对于 CyberShake 工作流，在 $D_1(w)$阶段，3 种调度策略的平均执行时间均超过对应截止日期参考线，说明它们均不满足截止日期约束。虽然 ADPSOGA 调度策略的 $NEC(w)$最低，但无法判定其就是最优调度策略，因为其不满足截止日期约束条件。在 $D_3(w)$和 $D_5(w)$阶段，其整体情况与 Montage 类似。MCPCPP 调度策略虽然在 $D_5(w)$阶段 $NEC(w)$最低，但其不满足相应截止日期约束条件。另外，2 种基于 PSO 的调度策略均满足对应截止日期约束，其中 ADPSOGA 调度策略的平均标准执行代价 $NEC(w)$最低，其是相对 MCPCPP 和 WPSO 调度策略的最优调度方法。

对于 Epigenomics 工作流，MCPCPP 调度策略虽然在每个阶段的平均标准执行代价 $NEC(w)$都是最低的，但其对应的平均执行时间均明显高于截止日期参考线，因此，MCPCPP 调度策略不是有效方法。在满足约束条件的调度策略中，ADPSOGA 调度策略在 3 个阶段的优势较为明显，其不仅在工作流平均执行时间上处于最低值，而且相应的平均标

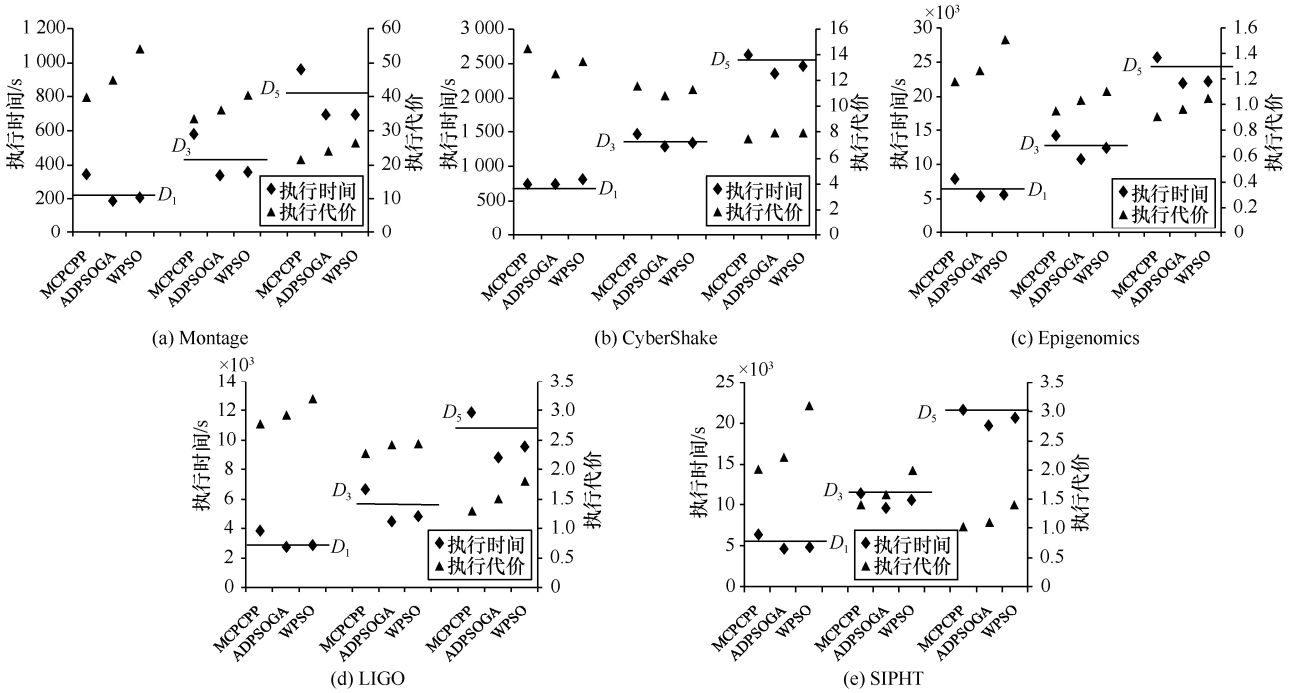


图 7 不同调度策略在 3 种截止日期下的工作流标准执行代价和执行时间

准执行代价 $NEC(w)$也是最低的。这主要是因为 ADPSOGA 调度策略的编码技巧和更新方式，使其获得更大的种群类别和问题搜索空间，能够更好地适应动态多云环境。

相对于 LIGO 和 SIPHT 工作流，3 种调度策略的表现情况与 Epigenomics 工作流类似，在此不再赘述。

综上，MCPCPP 调度策略虽然工作流平均标准执行代价 $NEC(w)$通常较低，但其往往不满足对应的截止日期约束，不能算是好的调度方法。WPSO 调度策略虽然通常满足截止日期约束，但其产生的平均标准执行代价 $NEC(w)$一般不理想。ADPSOGA 调度策略相对于所有测试样例，在满足截止日期约束下的平均标准执行代价 $NEC(w)$表现是最优的。

6 结束语

本文提出一种基于 PSO 和 GA 更新算子的多云环境下 ADPSOGA 工作流调度策略。该算法综合考虑云间数据通信代价、虚拟机启停时间和云环境下波动干扰因素等。其是以工作流截止日期为约束，优化工作流总执行代价的元启发式算法。通过对 5 个类型的中型工作流的调度测试表明，ADPSOGA 调度策略在多云环境存在波动因素干扰下，对带截止日期约束的工作流调度具有更佳的有效性和适应性。

未来工作中，将进一步考虑压缩初始资源池大小的方法，使算法的执行效率进一步提升。另外，将对动态波动的影响因子逐一分析，找出干扰最大的若干影响因子，进一步提高算法的顽健性。

参考文献：

[1] RODRIGUEZ M A, BUYYA R. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds[J]. IEEE Transactions on Cloud Computing, 2014, 2(2): 222-235.

[2] FARD H M, PRODAN R, FAHRINGER T. A truthful dynamic workflow scheduling mechanism for commercial multicloud environments[J]. IEEE Transactions on Parallel & Distributed Systems, 2013, 24(6): 1203-1212.

[3] MALAWSKI M, JUVE G, DEELMAN E, et al. Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds[C]//International Conference for High Performance Computing, Networking, Storage and Analysis. 2012: 1-11.

[4] MASDARI M, SALEHI F, JALALI M, et al. A survey of PSO-based scheduling algorithms in cloud computing[J]. Journal of Network & Systems Management, 2017, 25(1): 1-37.

[5] WU Z, NI Z, GU L, et al. A revised discrete particle swarm optimization for cloud workflow scheduling[C]//International Conference on Computational Intelligence and Security, 2010: 184-188.

[6] ABRISHAMI S, NAGHIBZADEH M, EPEMA D H J. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds[J]. Future Generation Computer Systems, 2013, 29(1): 158-169.

[7] MATTOSO M, DIAS J, OCAÑA K A C S, et al. Dynamic steering of HPC scientific workflows: a survey[J]. Future Generation Computer Systems, 2015, 46(C): 100-113.

[8] ZENG L, VEERAVALLI B, ZOMAYA A Y. An integrated task com-

putation and data management scheduling strategy for workflow applications in cloud environments[J]. Journal of Network & Computer Applications, 2015, 50(C): 39-48.

[9] CHEN W N, ZHANG J. An ant colony optimization approach to a Grid workflow scheduling problem with various QoS requirements[J]. IEEE Transactions on Systems Man & Cybernetics Part C, 2008, 39(1): 29-43.

[10] CAO H, JIN H, WU X, et al. DAGMap: efficient and dependable scheduling of DAG workflow job in grid[J]. The Journal of Supercomputing, 2010, 51(2): 201-223.

[11] HIGASHINO W A, CAPRETZ M A M, TOLEDO M B F D, et al. A hybrid particle swarm optimization-genetic algorithm applied to grid scheduling[J]. International Journal of Grid & Utility Computing, 2016, 7(2): 113-129.

[12] 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 Y C, LI X P, WANG Q, et al. Bottom level based heuristic for workflow scheduling in grids[J]. Chinese Journal of Computers, 2008, 31(2): 282-290.

[13] ABRISHAMI S, NAGHIBZADEH M, EPEMA D H J. Cost-driven scheduling of grid workflows using partial critical paths[J]. IEEE Transactions on Parallel & Distributed Systems, 2012, 23(8): 1400-1414.

[14] KHAJEMOHAMMADI H, FANIAN A, GULLIVER T A. Fast workflow scheduling for grid computing based on a multi-objective genetic algorithm[C]//Communications, Computers and Signal Processing. 2013: 96-101.

[15] 沈虹, 李小平. 带准备时间和截止期约束的云服务工作流调度算法[J]. 通信学报, 2015, 36(6): 183-192.
SHEN H, LI X P. Algorithm for the cloud service workflow scheduling with setup time and deadline constraints[J]. Journal on Communications, 2015, 36(6): 183-192.

[16] 肖鹏, 胡志刚, 屈喜龙. 面向数据密集型工作流的能耗感知调度策略[J]. 通信学报, 2015, 36(1): 149-158.
XIAO P, HU Z G, QU X L. Energy-aware scheduling policy for data-intensive workflow[J]. Journal on Communications, 2015, 36(1): 149-158.

[17] MAO M, HUMPHREY M. Auto-scaling to minimize cost and meet application deadlines in cloud workflows[C]//High Performance Computing, Networking, Storage and Analysis. 2011: 1-12.

[18] PANDEY S, WU L, GURU S M, et al. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments[C]//IEEE International Conference on Advanced Information Networking and Applications. 2010: 400-407.

[19] KEAHEY K, TSUGAWA M, MATSUNAGA A, et al. Sky computing[J]. IEEE Internet Computing, 2009, 13(5): 43-51.

[20] LIN B, GUO W, XIONG N, et al. A pretreatment workflow scheduling approach for big data applications in multicloud environments[J]. IEEE Transactions on Network & Service Management, 2016, 13(3): 581-594.

[21] WOO S S, MIRKOVIC J. Optimal application allocation on multiple public clouds[J]. Computer Networks, 2014, 68(11): 138-148.

[22] KENNEDY J, EBERHART R. Particle swarm optimization[C]//IEEE International Conference on Neural Networks. 2002: 1942-1948.

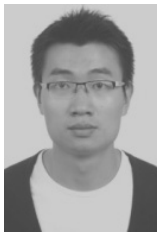
[23] SU J S, GUO W Z, YU C L, et al. Fault-tolerance clustering algorithm with load-balance aware in wireless sensor network[J]. Chinese Journal of Computers, 2014, 37(2): 445-456.

[24] SHI Y, EBERHART R. A modified particle swarm optimizer[C]// IEEE World Congress on Computational Intelligence. 1998: 69-73.

[25] BHARATHI S, CHERVENAK A, DEELMAN E, et al. Characterization of scientific workflows[C]//Workflows in Support of Large-Scale Science. 2008: 1-10.

[26] TOPCUOGLU H, HARIRI S, WU M Y. Performance-effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE Transactions on Parallel & Distributed Systems, 2002, 13(3): 260-274.

[作者简介]



林兵（1986-），男，福建福清人，博士，福建师范大学讲师，主要研究方向为云计算技术、计算智能及其应用。

郭文忠（1979-），男，福建泉港人，博士，福州大学教授、博士生导师，主要研究方向为计算智能及其应用。

陈国龙（1965-），男，福建莆田人，博士，福州大学教授、博士生导师，主要研究方向为人工智能、网络安全。