云中截止时间动态分配的工作流调度成本优化算法*

潘纪奎[1,2]，董心仪[1,2]，王子健[1]，卢政昊[1,2]，孙福权[1†]
(1. 东北大学秦皇岛分校 数学与统计学院，河北 秦皇岛 066000；2. 东北大学 信息科学与工程学院，沈阳 110000)

摘 要：现如今，如何在满足截止时间约束的前提下降低工作流的执行成本，是云中工作流调度的主要问题之一。三步列表调度算法可以有效解决这一问题。但该算法在截止时间分配阶段只能形成静态的子截止时间。为方便用户部署工作流任务，云服务商为用户提供了的三种实例类型，其中竞价实例具有非常大的价格优势。为解决上述问题，提出了截止时间动态分配的工作流调度成本优化算法(S-DTDA)。该算法利用粒子群算法对截止时间进行动态分配，弥补了三步列表调度算法的缺陷。在虚拟机选择阶段，该算法在候选资源中增加了竞价实例，大大降低了执行成本。实验结果表明，相较于其他经典算法，该算法在实验成功率和执行成本上具有明显优势。综上所述，S-DTDA 算法可以有效解决工作流调度中截止时间约束的成本优化问题。

关键词：云计算；工作流调度；截止期限；竞价实例；成本优化
中图分类号：TP391 **文献标志码**：A **文章编号**：1001-3695(2023)01-028-0172-06
doi：10.19734/j.issn.1001-3695.2022.06.0292

Cost optimization algorithm based on dynamic allocation of deadline for workflow scheduling in clouds

Pan Jikui[1,2], Dong Xinyi[1,2], Wang Zijian[1], Lu Zhenghao[1,2], Sun Fuquan[1†]
(1. *School of Mathematics & Statistics, Northeastern University at Qinhuangdao, Qinhuangdao Hebei* 066000, *China*; 2. *College of Information Science & Engineering, Northeastern University, Shenyang* 110000, *China*)

Abstract: Nowadays, how to reduce the execution cost of workflow under the premise of meeting the deadline constraint is one of the main problems of workflow scheduling in cloud. A three-step list scheduling algorithm can solve this problem effectively. However, the algorithm can only form static sub-deadline in deadline allocation stage. To provide convenience for users to deploy workflow tasks, cloud service providers provide users with three types of instances. Among them, spot instance has great price advantage and can greatly reduce the execution cost of workflow. To solve the above problems, this paper proposed a cost optimization algorithm based on dynamic allocation of deadline for workflow scheduling(S-DTDA). This algorithm used particle swarm optimization to dynamically allocate the deadline, which made up for the defect of the three-step list scheduling algorithm. In the virtual machine selection stage, the algorithm added the spot instance to the candidate resources, which greatly reduced the execution cost. Experimental results show that compared with other classical algorithms, this algorithm has obvious advantages in experimental success rate and execution cost. In conclusion, S-DTDA algorithm can effectively solve the cost optimization problem of deadline constraint in workflow scheduling.

Key words: cloud computing; workflow scheduling; deadline; spot instance; cost optimization

0 引言

工作流在生物信息学、天文学和物理学等大规模建模科学问题中应用广泛[1]。一个工作流由成百上千个任务和任务之间的依赖关系组成。通常使用有向无环图(DAG)来描述工作流。工作流调度的目的是在满足服务质量约束(QoS)的同时为工作流中的任务找到合适的执行资源[2]。云计算的出现和发展使得将大规模的工作流应用程序部署到云环境中执行成为一种更流行的方法[3]。云环境包含各种各样无限的云资源。用户采用按次付费的模式租用云资源来执行工作流[4]。更高性能的云资源拥有更快的执行速度，同时也需要更高的租用费用。因此同时考虑执行成本和完工时间是工作流调度的主要问题之一。在满足截止时间约束的前提下最小化执行成本是一个已知的 NP 难问题[5]。在分布式社区中，这一问题一直是多年来的热点[6]。

列表调度算法是用来解决工作流调度问题的常用算法。该算法一般包含两个步骤：a)利用预先定义的启发式信息设置各个任务的优先级并利用优先级构建一个任务列表；b)按照任务列表中顺序为每个任务配置合适的云资源。有些研究[7]在两步列表调度之前增加了截止时间环节，形成了三步列表调度算法，并且通过实验验证了三步列表调度算法确实优于两步列表调度算法。然而现有截止时间分配策略智能形成静态的子截止时间，所以还有进一步优化的空间。

在现实中，云服务商往往会提供过度的云资源以满足用户的峰值需求，从而导致了云数据中心的计算资源未得到充分利用。云服务商为了减少计算资源的浪费，为用户提供了更加经济的计算资源。Amazon 于 2009 年提出了一种竞价实例(spot

收稿日期：2022-06-01；**修回日期**：2022-08-05 **基金项目**：国家重点研发计划资助项目(2018YFB1402800)
作者简介：潘纪奎，男，山东临沂人，硕士研究生，主要研究方向为工作流调度；董心仪，女，山东烟台人，硕士研究生，主要研究方向为工作流调度；王子健，男，河北秦皇岛人，讲师，硕士，主要研究方向为工作流调度；卢政昊，男，辽宁本溪人，硕士研究生，主要研究方向为工作流调度；孙福权，男(通信作者)，辽宁大连人，教授，硕导，博士，主要研究方向为云资源调度分配、大数据分析(panjikuidxy@163.com).

instance)[8],这是一种基于出价的实例类型。当用户的出价不低于市场价格时,用户可以访问竞价实例;当用户出价低于市场价格时,将撤销该实例。虽然竞价实例具有较低的可靠性,但这类资源具有极大的价格优势。与配置相同的按需实例相比,竞价实例可以享受 50% ~90% 的折扣。这不仅为云服务商降低了资源闲置所带来的操作和维护成本,也为用户节省了运行大规模工作所需要的成本。虽然如此,但竞价实例存在可能被随时中断的问题,这会严重影响所提交应用程序的性能。所以在使用竞价实例时需要特别注意。

本文提出了截止时间动态分配的工作流调度成本优化算法(S-DTDA)。该算法利用粒子群算法对截止时间进行动态分配,弥补了三步列表调度算法的缺陷。在配置虚拟机时,该算法将竞价实例添加到候选资源池中,大大降低了执行成本。

1　相关研究

工作流调度最开始研究的是多处理器系统[9]和阵列系统[10]。这些系统都是免费使用的,所以最开始的研究都是以最小化完工时间为目的。随着云计算的出现和兴起,云工作流调度成为了新的研究课题。用户以租用的形式使用云资源,使减小执行成本成为云中工作流调度必须要考虑的问题。为解决该问题,相继出现了很多研究。文献[11]使用了粒子群算法。文献[12]改进了传统的粒子群算法,提出了免疫粒子群算法,改善了粒子群算法的收敛速度,提高了算法质量。本文提出的 S-DTDA 算法也使用了粒子群算法,用粒子群中每一个粒子代表一个解决方案,则最优的粒子代表最佳的解决方案。有些文献利用了分治的思想进行工作流调度。例如文献[13]利用了关键路径的概念,提出了 PCP 算法。该算法首先找到工作流图中的关键路径,并为关键路径上的任务分配截止时间,然后以关键路径上的任务为终点继续反向查找关键路径并分配最后期限。文献[14]在 PCP 算法的基础上提出了 IC-PCP 算法。该算法与 PCP 具有类似的关键路径查找方案,然后直接进行服务资源的配置。列表算法经常被用于解决工作流调度问题。有些研究在传统列表调度算法的基础上增加了截止时间分配的步骤。文献[7]提出的 ProLis 算法、文献[15]提出的 DCO/DUCO 算法和文献[16]提出的 HAS 算法都是按照自定义的启发式信息进行最后期限的分配。文献[17]提出了 HCDCW 算法,该算法将优先在私有云中调度执行,当任务执行时间超出任务截止期约束时,使用公有云调度部分工作流。文献[18]提出了 HAPSO 算法,该算法在工作流截止期约束下,有效权衡了完成时间与执行成本。虽然这些方法的调度结果都很优异,但其大多针对每个任务生成静态子截止时间,还可以进一步优化。上述优化算法都是将调度任务部署在按需实例上进行。虽然按需实例工作可靠,但是相对于竞价实例,其价格太过昂贵,需要更高的执行成本。

当竞价实例作为一种新的资源类型出现后,相继涌现了大量关于竞价实例科研成果。由于竞价实例的未来市场价格的不确定性,许多研究对竞价实例的价格进行预测并对其容错性进行了研究。文献[19]研究了 AR、ETS、ARIMA 和 GARCH 模型。GARCH 能够更好地动态预测竞价实例的短期和中期实例价格,AR 对 GPU 实例中长期价格预测具有较好的预测效果,ETS 为基于平滑度的长期竞价实例价格预测提供了较好的结果,ARIMA 给出了基于长期竞价实例价格波动的最佳预测结果。文献[20]提出一种随机森林回归算法,能够对次日及一周内的竞价实例的价格进行准确预测。文献[21]利用了竞价实例的价格历史,建立了 K-nearest neighbors(KNN)回归模型,该模型可以准确预测竞价实例未来的价格。文献[22]提出了一个动态调度程序(burst-HADS),有效降低了资金成本,甚至在竞价实例休眠场景下也满足了应用程序的截止日期。文献[23]引入了一个称为 MASA-CUDAlign 的新框架,即使多个实例被撤销,也能保证满足截止时间。文献[24]提出了 chaos-ANFIS 模型,能够对竞价实例的未来价格进行精准预测。竞价实例虽然可靠性较低,但其容错机制和价格预测研究已经很成熟。利用竞价实例进行工作流调度任务能更加有效地降低执行成本。

2　问题描述

2.1　工作流模型

一个工作流由大量的任务与任务之间的依赖关系组成。工作流模型通常用有向无环图 $DAG=(V, E)$ 来表示。其中,$V=\{t_1,t_2,\cdots,t_n\}$ 是有向无环图中节点的集合,工作流中的每一个任务都用一个节点表示,w_i 表示任务 t_i 的工作量。E 是有向无环图中边的集合,E 中的每一个元素 $e_{ij}=\{t_i,t_j\}$ 表示任务 t_i 和 t_j 的依赖关系,表示必须在任务 t_i 执行结束之后才能执行任务 t_j,这时把任务 t_i 和 t_j 分别叫做父任务和子任务。如果任务 t_i 向任务 t_j 传输数据,传输的数据量用 d_{ij} 表示。本文把一个没有父任务的任务叫做起始任务,把一个没有子任务的任务叫做终止任务。由于大部分的调度算法不能适用于一个工作流存在多个起始任务或者多个终止任务的情况,所以需要对工作流模型进行一般化处理。本文分别在工作流的起始处和终止处添加一个负载为零的任务来代表整个工作流的起始任务和终止任务,分别用 t_{entry} 和 t_{exit} 表示。图 1 展示了一个工作流模型。

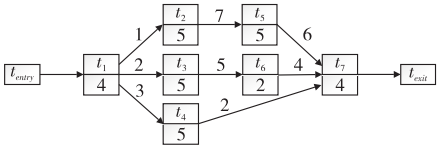


图 1　工作流示例

Fig. 1　Example of workflow

2.2　云资源模型

为了对云资源进行统一管理和为用户提供服务,云服务商将云资源虚拟化成资源池的形式。云服务商为用户提供保留实例、按需实例和竞价实例三种拥有不同计费模式的实例。每一种资源实例都拥有多种类型,不同类型的资源拥有不同的处理速度和价格。越高性能的资源代表越快的处理速度和高昂的价格。本文基于 Amazon EC2 的实际产品建立了按需实例和竞价实例两种资源模型。除竞价实例的可靠性低外,配置相同的按需实例和竞价实例只有计费模式不同,在其他方面没有区别。集合 $R=\{r_1,r_2,\cdots,r_n\}$ 代表用户可以租用的资源类型的集合。一个虚拟机就是一个具体的资源实例,用 vm_k^{type} 表示。其中 $type\in\{d,s\}$ 被用来区分按需实例和竞价实例,k 类型的按需实例和竞价实例分别用 vm_k^d 和 vm_k^s 表示;$k\in R$ 表示虚拟机的资源类型。工作流中的任务是在虚拟机上执行的,每一个虚拟机 vm_k^{type} 都是集合 $VMS=\{vm_1^d,vm_2^d,\cdots,vm_n^d,vm_1^s,vm_2^s,\cdots,vm_n^s\}$ 中的一个元素。一个任务只能部署到一个虚拟机上执行,而一个虚拟机可以按照既定的顺序执行多个任务。虚拟机采用即付即用的收费模式,以用户租用的单元时间数量为标准进行收费。如果用户没有使用完一个完整的单元时间时,用户也需要支付一个完整的单元时间的费用。如图 2 所示,T 是虚拟机的单元时间长度。在 0 时刻虚拟机开始被租用,3.2 T 时刻虚拟机结束租用,此时用户需要支付 4 个单元时间的费用。

竞价实例的租用受市场价格和用户出价的影响。当市场

价格低于用户出价时,用户才可以使用竞价实例。如图 3 所示,虚线表示用户的出价,实线表示竞价实例的市场价格。当虚线高于实线时,用户才可以使用竞价实例。当时间在 T_b 和 T_d 之间时,竞价实例的市场价格低于用户的出价 B。用户在 T_b 时刻可以访问竞价实例,在 T_d 时刻终止访问。这段时间内,用户按照竞价实例的市场价格支付租用费用,而不是按照用户的出价。

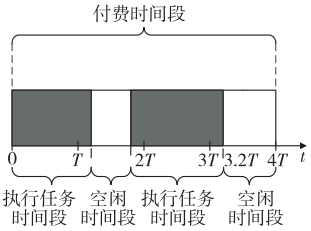


图 2 付费时间段示例

Fig. 2 Example of a paid time period

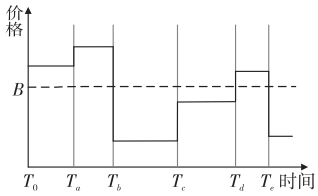


图 3 竞价实例的价格波动示例

Fig. 3 Examples of the fluctuations about spot insance price

当任务 t_i 部署到虚拟机 vm_k^{type} 上执行时,假设虚拟机 vm_k^{type} 的处理能力为 $P(vm_k^{type})$,那么任务 t_i 的执行时长为

$$ET_{i,k}^{type}=\frac{w_i}{P(vm_k^{type})} \tag{1}$$

假设所有虚拟机都在同一物理区域内,而且虚拟机的平均带宽基本相同。$\mathrm{vm}(t_i)$ 代表执行任务 t_i 的虚拟机,bw 代表虚拟机的带宽,那么任务 t_i 与 t_j 之间的数据传输时长为

$$TT_{i,j}=\begin{cases}\dfrac{d_{i,j}}{bw} & \text{if} \quad \mathrm{vm}(t_i)\neq\mathrm{vm}(t_j)\\ 0 & \text{otherwise}\end{cases} \tag{2}$$

本文用 $RT_{i,k}^{type}$ 代表 vm_k^{type} 可执行 t_i 的最早时刻,即 vm_k^{type} 上排列在任务 t_i 之前任务的完成时刻,当任务 t_i 是 vm_k^{type} 上的第一个任务时,$RT_{i,k}^{type}$ 为 vm_k^{type} 的启动完成时刻,则任务 t_i 在 vm_k^{type} 上的执行开始时刻 $ST_{i,k}^{type}$ 用式(3)计算,执行完成时刻 $FT_{i,k}^{type}$ 用式(4)计算。

$$ST_{i,k}^{type}=\max\{RT_{i,k}^{type},\max_{t_j\in t_i's\ parents}\{FT_{j,k}^{type}+TT_{j,i}\}\} \tag{3}$$

$$FT_{i,k}^{type}=ST_{i,k}^{type}+ET_{i,k}^{type} \tag{4}$$

用 LST_k^{type} 代表虚拟机 vm_k^{type} 的租用开始时刻,即虚拟机 vm_k^{type} 上第一个任务开始接收数据的时刻。LST_k^{type} 代表虚拟机 vm_k^{type} 的租用结束时刻,即 vm_k^{type} 上最后一个任务完成数据传输的时刻。LST_k^{type} 和 LFT_k^{type} 分别为

$$LST_k^{type}=ST_{i,k}^{type}-\max_{t_j\in t_i's\ parents}\{TT_{j,i}\} \tag{5}$$

$$LFT_k^{type}=FT_{i,k}^{type}+\max_{t_j\in t_i's\ children}\{TT_{i,j}\} \tag{6}$$

云工作流调度成本通常包含执行成本和传输成本。本文的数据传输时间包含在虚拟机的租用时间中,使用虚拟机的租用时间计算执行成本,因此传输成本包含在执行成本中。假设租用 vm_k^{type} 一个单位时长 T 所需要承担的租用费用为 UC_k^{type},则虚拟机 vm_k^{type} 的租用费用 EC_k^{type} 为

$$EC_k^{type}=\left\lceil\frac{LFT_k^{type}-LST_k^{type}}{TI}\right\rceil\times UC_k^{type} \tag{7}$$

2.3 调度模型

调度的目的是将工作流中的任务部署到合适的虚拟机上,并使每一个任务都合理有序地进行。将已经被租用的虚拟机用集合 $I=\{vm_1^d,vm_2^d,\cdots,vm_I^d,vm_1^s,vm_2^s,\cdots,vm_J^s\}$ 表示。一个映射 $m=\langle t_i,vm_k^{type},ST_{i,k}^{type},FT_{i,k}^{type}\rangle$ 表示将一个任务 t_i 部署在 vm_k^{type} 上执行。其中,任务 t_i 在 vm_k^{type} 上执行开始时刻用 $ST_{i,k}^{type}$ 表示,任务 t_i 在 vm_k^{type} 上执行结束时刻用 $FT_{i,k}^{type}$ 表示。用集合 M 表示所有任务到虚拟机的映射集合。如上所述,一次调度结果可以描述为$\langle I,M\rangle$。整个虚拟机 vm_k^{type} 的租用结束时刻用 FT_k^{type} 表示。分别用 $\mathrm{makespan}(I,M)$ 和 $\mathrm{cost}(I,M)$ 代表一次调度结果的完工时间的执行成本,并由式(8)和(9)计算。

$$\mathrm{makespan}(I,M)=\max_{k\in I}\{FT_k^{type}\} \tag{8}$$

$$\mathrm{cost}(I,M)=\sum_{i=1}^{|I|}EC_i^d+\sum_{j=1}^{|J|}EC_j^s \tag{9}$$

其中:$\sum_{i=1}^{|I|}EC_i^d$ 表示租用的全部按需实例的执行成本的和;$\sum_{j=1}^{|J|}EC_j^s$ 表示租用的全部竞价实例的执行成本的和,两部份相加得到执行成本的总和。一个工作流的截止时间用 D 表示,那么工作流调度问题可以表示为

$$\begin{aligned}&\min \quad \mathrm{cost}(I,M)\\ &\text{s. t.} \quad \mathrm{makespan}(I,M)\leqslant D\end{aligned} \tag{10}$$

3 算法设计

如何确定每一个任务期限是本文所研究调度问题的关键。粒子群算法作为一种随机优化技术,非常适用于解决本文所面对的问题。文献[25]提出了采用三步列表调度算法进行云工作流调度,并提出一种基于粒子群的动态最后期限分配方法(DY-DD)。本文改进了 DY-DD 算法,在之前算法的基础上增加了竞价实例,提出了一种表现更加优异的算法(S-DTDA)。粒子群算法中的每一个粒子的本质是一个多维向量,通过渐进式的计算求解后,得到最优粒子。这一部分应用列表调度算法计算了 *t-level*、*b-level* 与 *alap* 的值[9],提出了一种基于粒子群的截止时间分配方法对截止时间进行分配,并利用竞价实例的价格优势提出了一种截止时间动态分配的工作流调度成本优化算法。

3.1 粒子群算法

粒子群算法在随机优化的基础上,通过迭代的方法来获得粒子的最优位置。一个拥有 *size* 个粒子的粒子群用 $X=\{x^1,x^2,\cdots,x^{size}\}$ 表示,粒子群中的第 k 个粒子 $\boldsymbol{x}^k=\{x_1^k,x_2^k,\cdots,x_d^k\}$ 是一个 d 维向量,其物理含义是 d 维空间的一个点,同时也表示一个独立的解决方案。粒子群的计算按照迭代的方式进行,在每一轮迭代中需要移动所有的粒子,对粒子的位置进行调整。当满足终止条件时,最终的解决方案就是当前最优的粒子。第 k 个粒子的移动速度 $v^k=\{v_1^k,v_2^k,\cdots,v_d^k\}$,记忆中粒子 $\boldsymbol{x}^k$ 的最佳位置就是该粒子的局部最优解,用 $lb^k=\{lb_1^k,lb_2^k,\cdots,lb_d^k\}$ 表示。记忆中粒子群的最佳位置就是该粒子群中所有粒子的全局最优解,用 $gb=\{gb_1,gb_2,\cdots,gb_d\}$ 表示。粒子每个分量的最大值用 $x^{max}=\{x_1^{max},x_2^{max},\cdots,x_d^{max}\}$ 表示,粒子每个分量的最小值用 $x^{min}=\{x_1^{min},x_2^{min},\cdots,x_d^{min}\}$ 表示,粒子速度每个分量的最大值用 $v^{max}=\{v_1^{max},v_2^{max},\cdots,v_d^{max}\}$ 表示,粒子速度每个分量的最小值用 $v^{min}=\{v_1^{min},v_2^{min},\cdots,v_d^{min}\}$ 表示。假设 $x_i^k(t)$ 为粒子 $\boldsymbol{x}^k$ 在第 t 轮迭代时第 i 个分量的值,那么可以用式(11)和(12)计算得到该粒子在第 $t+1$ 轮迭代的更新方式。

$$x_i^k(t+1)=x_i^k(t)+v_i^k(t) \tag{11}$$

$$x_i^k(t+1)=\max(x_i^{min},\min(x_i^k(t+1),x_i^{max})) \tag{12}$$

其中:$v_i^k(t)$ 的值通过式(13)和(14)计算得出。

$$\begin{aligned}v_i^k(t+1)=&wv_i^k(t)+c_1r_1(lb_i^k(t)-x_i^k(t))+\\&c_2r_2(gb_i(t)-x_i^k(t))\end{aligned} \tag{13}$$

$$v_i^k(t+1)=\max(v_i^{min},\min(v_i^k(t+1),v_i^{max})) \tag{14}$$

其中:w 代表惯性参数,其值取随机数;c_1 和 c_2 代表加速系数;r_1 和 r_2 取随机数。

3.2 截止时间动态分配的 S-DTDA 算法

三步列表调度算法通常先对截止时间进行分配,将整个工作流的解释时间 D 分配给各个子任务,使得每个子任务 t_i 获得一个子截止时间 sd_i;然后对任务进行排序,得到一个任务队

列 *list*;最后按照队列顺序为每一个任务配置合适的虚拟机。本文提出了截止时间动态分配的工作流调度成本优化算法(S-DTDA),对三度列表调度算法的执行步骤进行了调整。首先对任务进行拓扑排序生成队列 *list*;然后利用粒子群算法找到近似最优的截止时间分配结果;最后在虚拟机配置阶段,利用竞价实例的价格优势,在候选资源池中增加了竞价实例,为每个任务分配合适的虚拟机。

在列表调度算法的应用中,经常需要计算并使用一个任务的 *t-level*、*b-level*、*alap* 值。从开始节点 t_{entry} 到任务 t_i 最长路径长度为该任务的 *t-level* 值,*t-level* 值的大小关系到该任务可以开始执行的最早时刻。从任务 t_i 到终止节点 t_{exit} 的最长路径长度为该任务的 *b-level* 值。在整个工作流的关键路径长度不受影响的前提下,任务 t_i 开始执行时刻可拖延的最大时间为该任务的 *alap* 值。已知每个任务的执行时间是计算这些值对的前提。然而任务精确的执行时间只有在调度结束后才能进行计算。所以本文采取一种近似的方法来对任务 t_i 的执行时间 ET_i 进行计算,如式(15)所示。

$$ET_i = \frac{w_i}{P(r^*)} \tag{15}$$

其中:r^* 表示云平台能够提供的处理速度最快的虚拟机类型。任务 t_i 的 *t-level*、*b-level*、*alap* 值分别用 tl_i、bl_i 和 $alap_i$ 表示,通过式(16)~(18)计算。

$$\begin{cases} tl_i = \max\limits_{t_j \in t_i's\ parent} \{tl_j + ET_j + d_{j,i}/bw\} \\ tl_{entry} = 0 \end{cases} \tag{16}$$

$$\begin{cases} bl_i = \max\limits_{t_j \in t_i's\ children} \{bl_j + d_{i,j}/bw\} + ET_i \\ bl_{exit} = 0 \end{cases} \tag{17}$$

$$\begin{cases} alap_i = \min\limits_{t_j \in t_i's\ children} \{alap_j - d_{i,j}/bw\} - ET_i \\ alap_{exit} = bl_{entry} \end{cases} \tag{18}$$

粒子群算法的使用需要解决两个问题:a)怎样按照粒子的模型确定结题解决方案的编码方式;b)如何对两个不同粒子的优劣进行比较。本文提出的 S-DTDA 算法将粒子定义为截止时间分配的结果,不同的截止时间分配结果表示为不同的粒子。工作流中任务的个数表示为粒子的维度 d。参照队列 *list* 的顺序确定粒子的各分量值与各任务的映射关系。例如,粒子 k 的第 l 个分量 x_l^k 的值表示第 k 种分配方案的第 l 个任务的子截止时间。判断两个粒子优劣的方案如式(19)所示。

$$f_i > f_j \equiv \begin{cases} cost_i < cost_j & \text{if } makespan_i, makespan_j \leqslant D \\ makespan_i < makespan_j & \text{otherwise} \end{cases} \tag{19}$$

其中:f_i、f_j 表示对粒子 x^i、x^j 进行虚拟机配置后的最终调度结果。当两种调度方案的完工时间均满足截止时间约束时,执行费用小的方案较优,否则完工时间小的调度方案较优。

为合理控制粒子的移动范围,需要设置粒子各分量的最大值与最小值。因为任务 t_i 开始执行的时间不会早于 tl_i,最晚开始时间也不会晚于 $alap_i$。为了缩小搜索空间,粒子各分量的最大值与最小值的设置如式(20)和(21)所示。x_i^{max} 和 x_i^{min} 分别表示粒子第 i 个分量的最大值和最小值。

$$x_i^{max} = \frac{(alap_i + ET_i) \times D}{bl_{entry}} \tag{20}$$

$$x_i^{min} = \frac{(tl_i + ET_i) \times D}{bl_{entry}} \tag{21}$$

粒子移动速度的最大值与最小值如式(22)和(23)所示。

$$v_i^{max} = \frac{x_i^{max} - x_i^{min}}{\theta} \tag{22}$$

$$v_i^{min} = -v_i^{max} \tag{23}$$

其中:θ 是一个常量。

为了加快粒子群的收敛速度,按照式(24)对 $t=0$ 时刻的全局最优解进行设置。

$$gb_i(t=0) = \frac{(bl_{entry} - bl_i + ET_i) \times D}{bl_{entry}} \tag{24}$$

算法 1　动态截止时间分配方法

输入:一个工作流 $DAG=(V,E)$。
输出:调度方案 $S=\langle I,M\rangle$。
list ←按照 *b_level* 对工作流中的任务进行降序排序
设置粒子群算法参数与全局最优位置 *gb*
初始化种群,设置每个粒子的局部最优位置 *lb*
while 不满足迭代条件 do
　for 每个粒子 $\boldsymbol{x}^k, k \in \{1,2,\cdots,size\}$ do
　　计算并更新速度←通过式(13)和(14)
　　计算并更新位置←通过式(11)和(12)
　　按照 *list* 的顺序将 $\boldsymbol{x}^k$ 解码为 S^k←通过算法 2
　　更新粒子 $\boldsymbol{x}^k$ 的局部最优位置 lb^k
　end for
　更新全局最优位置 *gb*←通过式(19)
end while
按照 *list* 的顺序将 *gb* 解码为调度方案 *S*←通过算法 2
return 调度方案 *S*

算法 1 描述了动态截止时间分配方法。算法的输入是一个工作流,输出是调度方案。首先对所有的任务按照 *b_level* 值进行降序排序(第 1 行)。然后对粒子群算法的相关参数进行初始化并对全局最优位置和局部最优位置进行设置(第 2、3 行)。开始执行粒子群算法(第 4~12 行)。利用算法 2 对每个粒子进行解码,生成解决方案。利用式(19)比较生成的解决方案,更新局部最优位置和全局最优位置(第 9~11 行)。粒子群算法迭代结束后,返回整个算法的最终调度方案(第 13、14 行)。

算法 2　虚拟机选择算法

输入:任务列表 *list*;一个粒子 *x*。
输出:调度结果 $S=\langle I,M\rangle$。
$I \leftarrow \varnothing, M \leftarrow \varnothing$
for $t_i \in list$ do
　for $vm_k^{type} \in I$ do
　　vm_k^{type}←选择增加 t_i 后满足 x_i 并且租用费用增量最小的虚拟机
　end for
　if 没有找到合适的虚拟机 then
　　for $k \in R$ do
　　分别计算增加 t_i 后按需实例和竞价实例的费用增量 inC_k^d 和 inC_k^s
　　vm_k^{type}←将 inC_k^d 和 inC_k^s 与上一次迭代的费用增量进行比较,选择满足 x_i 且增量最小的虚拟机
　　end for
　end if
　if 仍旧未找到符合要求的虚拟机 then
　　vm_k^{type}←选择能够最早结束该任务的虚拟机
　　while 不能满足 x_i && vm_k^{type} 不是最快的虚拟机 do
　　vm_k^{type}←选择一个更快类型的虚拟机并更新任务结束时间
　　end while
　end if
　$vm_k^{type} = (EC_k^d > EC_k^s\ ?\ vm_k^s : vm_k^d)$
　$M = M \cup \{\langle t_i, vm_k^{type}, ST_{i,k}^{type}, FT_{i,k}^{type}\rangle\}$
　if $vm_k^{type} \notin I$ then
　　$I = I \cup \{vm_k^{type}\}$
　end if
end for
return $S=\langle I,M\rangle$

粒子群通过虚拟机选择算法进行解码,具体如算法 2 所示。算法的输入包括一个任务列表 *list* 和一个粒子 *x*,输出是由 *x* 解码获取的调度方案。算法首先对 *list* 中的所有任务进行遍历(第 2 行)。在已选择的虚拟机中选择增加 t_i 后满足 x_i 并且租用费用增量最小的虚拟机(第 3~5 行)。如果没有找到合适的虚拟机,则遍历所有的虚拟机类型,在竞价实例和按需

实例中选择满足 x_i 且费用增量最小的虚拟机(第 6 ~ 11 行)。如果仍旧未找到符合要求的虚拟机,则选择能够最早结束该任务的虚拟机。当选择的虚拟机不能满足 x_i 且不是最快的虚拟机时进行迭代,选择一个更快的虚拟机并更新任务时间,直到找到合适的虚拟机(第 12 ~ 17 行)。获得选择的虚拟机类型,计算执行开始时刻和执行结束时刻,把找到的任务到虚拟机的映射并入集合 M 中(第 18、19 行)。如果该虚拟机不在已选择的虚拟机集合 I 中,则将该虚拟机并入集合 I(第 20 ~ 22 行)。结束遍历,返回最终的调度结果(第 23、24 行)。

4 仿真实验

4.1 实验设置

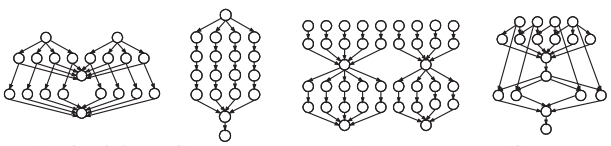
仿真实验部署在一台 Windows 10 的操作系统 PC 机上。PC 机使用 Intel ® Core™ i5 处理器,内存为 8 GB。仿真程序使用 Java 语言编写,在 Eclipse 开发环境下运行。仿真程序将文献[7]提供的开源程序进行修改,对竞价实例重新建模并合并到原有程序中。表 1 给出了算法相关参数的设置方式。

表 1 参数设置

Tab. 1 Parameter settings

参数	值	描述
t_{max}	70	粒子群迭代次数
size	25	粒子群中粒子的数量
c_1	2	粒子群算法加速系数
c_2	2	粒子群算法加速系数
θ	5	控制粒子群移动速度最大/最小值的常量

仿真实验利用 Bharathi 等人提供的工作流生成器生成了四种适用于不同领域的工作流模型,它们分别为 CyberShake、Epigenomics、LIGO 和 Montage。图 4 展示了四种工作流的结构特征。CyberShake 一般在刻画地震灾害领域中使用,是需要大量的 CPU 资源的数据密集型工作流;Epigenomics 一般在信息生物学领域中使用,是一个用来自动执行各种各样的基因组排序操作的数据处理管道;LIGO 一般在引力物理学领域中使用,也属于 CPU 密集型工作流;Montage 一般在天文学领域中使用,对 CPU 资源需求不高。四种工作流模型更加具体的描述可以参考文献[1]。



(a)CyberShake (b)Epigenomics (c)LIGO (d)Montage

图 4 四种工作流的结构特征

Fig. 4 Structural characteristics of the four workflows

本文对每个工作流模型都生成了 50,100,200,…,900 和 1 000个任务,共 11 种不同规模的工作流,并将它们用于仿真实验。

在仿真实验中,本文选用了 Amazon 公司的 r5. large、r5. xlarge、r5. 2xlarge、r5. 4xlarge 和 r5. 8xlarge 五种不同类型的虚拟机。参照 Amazon EC2 真实的收费标准并按照 r5. 8xlarge 按需实例的价格对每个虚拟机的租用费用进行了归一化处理。表 2 展示了当五种虚拟机按照按需实例类型进行收费时的处理能力和归一化成本。

表 2 五种按需实例的处理速度和价格

Tab. 2 Processing speed and cost of 5 on-demand instance

类型	处理速度	归一化成本	类型	处理速度	归一化成本
r5. large	1	0. 062 5	r5. 4xlarge	8	0. 5
r5. xlarge	2	0. 125	r5. 8xlarge	16	1
r5. 2xlarge	4	0. 25			

Amazon 公司在 2017 年以来调整了竞价实例的竞价策略。近年来竞价实例的价格已经相对稳定,对竞价实例未来的价格预测也变得简单。因此,本文假设竞价实例在租用期间都是可靠的。五种虚拟机按照竞价实例类型进行收费时的处理能力与它们按照按需实例类型收费时的处理能力相同。五种竞价实例的收费标准参照了 2021 年 12 月 6 日到 7 日的 48 h 历史价格进行归一化处理,详情如图 5 所示。

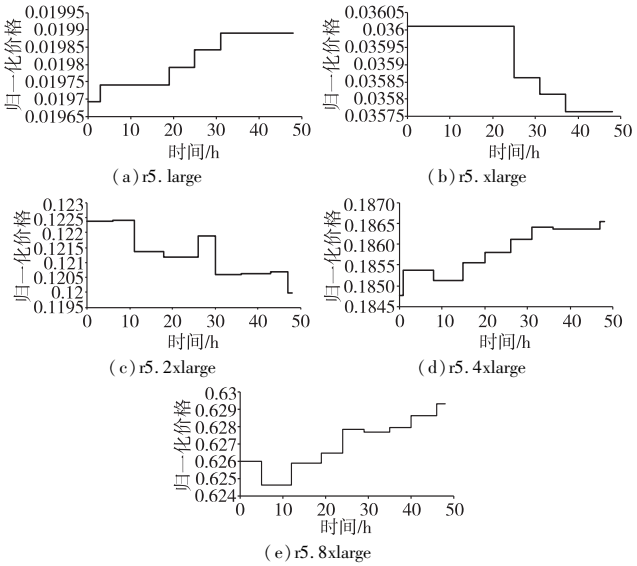


图 5 五种按需实例的价格波动图

Fig. 5 Price fluctuations of 5 on-demand instances

为评估 S-DTDA 算法的性能,仿真实验选择 ProLis[7]、PSO[11]和 IC-PCP[14]三种经典作为对比算法,采用了调度成功率和执行成本两个评估指标。性能越好的算法,成功率越高并且执行成本越低。截止时间的宽松程度对调度算法的结果有很大影响。截止时间越紧张,调度的成功率越低的同时成本越高。为了对本文算法作出更加合理的评估,引入以下两个基准对不同截止时间设置下的算法进行验证:

a)廉价调度。使用 HEFT[26]算法在一台最廉价的虚拟机上进行工作流调度。此时,完工时间用 M_c 表示,执行成本用 C_c 表示。

b)快速调度。使用 HEFT 算法在一台最昂贵的虚拟机上进行工作流调度。此时,完工时间用 M_f 表示,执行成本用 C_f 表示。

工作流截止时间的宽松程度用 $\lambda(\lambda \in [0,1])$ 表示,则工作流截止时间可表示为

$$D = M_F + (M_C - M_F) \times \lambda \tag{25}$$

4.2 实验结果

仿真实验先令 λ 从 0. 005 变化至 0. 05,步长为 0. 005 进行实验;又令 λ 从 0. 1 变化至 0. 5,步长为 0. 05 再次进行实验。分别观察在相对紧张和相对宽松的截止时间设置下,各算法在调度成功率和执行成本上的表现。

图 6 给出了 λ 从 0. 005 变化至 0. 05 时各算法的平均成功率。四种算法的平均成功率基本都随 λ 的增大而增大,直到增加到 100% 后稳定。显然,相对宽松的截止时间约束更容易找到调度方案。S-DTDA 和 ProLis 表现最优。只有在 Montage 模型下,当 λ 为 0. 005 时,S-DTDA 的成功率约为 91%,比ProLis 的表现稍差一点。IC-PCP 在 CyberShake 模型下表现最差,λ 从 0. 005 变化到 0. 05 从未达到 100%。PSO 在四种模型下的表现都较差,但随着 λ 的增大,成功率都能达到并稳定在 100%。

图 7 给出了 λ 从 0. 005 变化至 0. 05 时各算法的归一化执行成本。由实验结果可知,受实验成功率的影响,PSO 在 Epigenomics 和 Montage 下的成本先由低到低变化,等实验成功率

达到 100% 后,成本由高到低变化。另外三个算法在四个模型下的成本都表现为随 λ 的增加而降低。与 PSO 相比,其他三种算法都具有较大的优势。ProLis 和 IC-PCP 基本相差不大,但是相对于 S-DTDA,两种算法一直存在劣势。

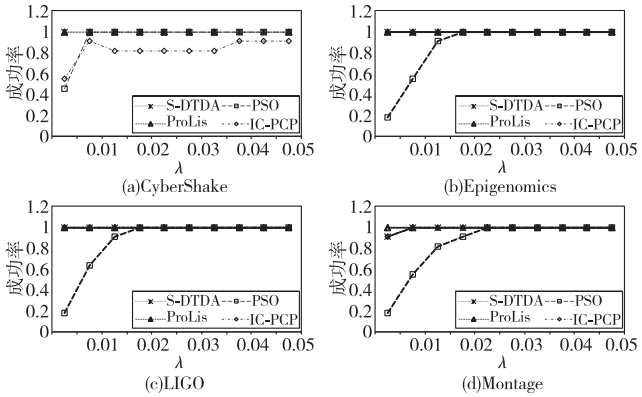


图 6　不同 λ 时每种方法的成功率

Fig. 6　Success rate of each method at different λ

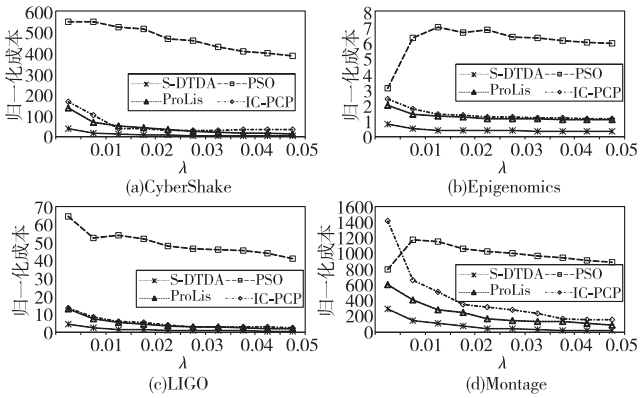


图 7　不同 λ 时每种方法的成本

Fig. 7　Cost of each method at different λ

图 8 给出了 λ 从 0.1 变化至 0.5 时各算法的归一化执行成本。由实验结果可知,S-DTDA、ProLis 和 IC-PCP 随着 λ 的增加变化不大,但是 S-DTDA 比 ProLis 和 IC-PCP 具有明显优势。PSO 在 Epigenomics 和 LIGO 模型下的表现最差。随着 λ 的增加,PSO 在 Cybershake 和 Montage 模型下的成本逐渐低于 IC-PCP,但是与 S-DTDA 和 ProLis 相比,PSO 仍处于明显劣势。

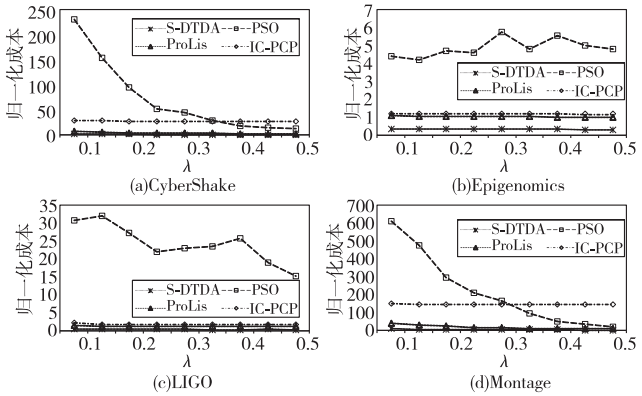


图 8　不同 λ 时每种方法的成本

Fig. 8　Cost of each method at different λ

5　结束语

本文提出了截止时间动态分配的工作流调度成本优化算法 S-DTDA。该算法利用粒子群算法对截止时间进行动态分配,弥补了三步列表调度算法的缺陷。在虚拟机选择阶段,该算法在候选资源中增加了竞价实例,大大降低了执行成本。对于一个给定的工作流、最后期限约束和云模型,该算法可以找到一个满足截止时间约束且优化了执行成本的调度方案。本文利用 Amazon EC2 提供的真实数据建立了模型并进行实验仿真。实验结果表明,相较于 ProLis、PSO 和 IC-PCPC 三个经典算法,S-DTDA 在实验成功率和执行成本上具有明显优势。

接下来,笔者会尝试改进本文算法或者探索新的算法解决本文所研究的问题,提高调度成功率并降低执行成本。除此之外,还会将研究目标扩展到多目标的工作流调度领域中。

参考文献:

[1] Juve G, Chervenak A, Deelman E, *et al*. Characterizing and profiling scientific workflows[J]. Future Generations Computer Systems, 2013, 29(3): 682-692.

[2] Wang Zijia, Zhan Zhihui, Yu Weijie, *et al*. Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling[J]. IEEE Trans on Cybernetics, 2020, 50(6): 2715-2729.

[3] Zhan Zhihui, Liu Xiaofeng, Gong Yuejiao, *et al*. Cloud computing resource scheduling and a survey of its evolutionary approaches[J]. ACM Computing Surveys, 2015, 47(4): article No. 63.

[4] Su Sen, Li Jian, Huang Qingjia, *et al*. Cost-efficient task scheduling for executing large programs in the cloud[J]. Parallel Computing, 2013, 39(4-5): 177-188.

[5] Wu Fuhui, Wu Qingbo, Tan Yusong. Workflow scheduling in cloud: a survey[J]. Journal of Supercomputing, 2015, 71(9): 3373-3418.

[6] Pinedo M. Scheduling: theory, algorithms, and systems[M]. Berlin: Springer, 2012.

[7] Wu Quanwang, Ishikawa F, Zhu Qingsheng, *et al*. Deadline-constrained cost optimization approaches for workflow scheduling in clouds[J]. IEEE Trans on Parallel and Distributed Systems, 2017, 28(12): 3401-3412.

[8] Cao Shujin, Deng Kefeng, Ren Kaijun, *et al*. An optimizing algorithm for deadline constrained scheduling of scientific workflows in IaaS clouds using spot instances[C]// Proc of IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking. Piscataway, NJ: IEEE Press, 2019: 1421-1428.

[9] Kwok Y K, Ishfaq A. Static scheduling algorithms for allocating directed task graphs to multiprocessors[J]. ACM Computing Surveys, 1999, 31(4): 406-471.

[10] Jia Yu, Rajkumar B, Kotagiri R. Workflow scheduling algorithms for grid computing[J]. Studies in Computational Intelligence, 2008, 146: 173-214.

[11] Wu Zhangjun, Ni Zhiwei, Gu Lichuan, *et al*. A revised discrete particle swarm optimization for cloud workflow scheduling[C]// Proc of International Conference on Computational Intelligence and Security. 2010: 184-188.

[12] Wang Pengwei, Lei Yinghui, Promise R, *et al*. Makespan-driven workflow scheduling in clouds using immune-based PSO algorithm[J]. IEEE Access, 2020, 8: 29281-29290.

[13] Abrishami S, Naghibzadeh M, Epema D H J. Cost-driven scheduling of grid workflows using partial critical paths[J]. IEEE Trans on Parallel and Distributed Systems, 2012, 23(8): 1400-1414.

[14] Saeid A, Mahmoud N, Dick H. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds[J]. Future Generation Computer Systems, 2013, 29(1): 158-169.

[15] Chen Weihong, Xie Guoqi, Li Renfa, *et al*. Execution cost minimization scheduling algorithms for deadline-constrained parallel applications on heterogeneous clouds[J]. Cluster Computing, 2021, 24: 701-715.

[16] Che Haiying, Wang Xiaolong, Wang Hong, *et al*. Scheduling with deadline constraint of healthcare applications on cloud-based workflow [J]. Journal of Medical Imaging and Health Informatics, 2020, 10(10): 2430-2438.

(下转第 184 页)

计算机技术与发展,2018,28(11):48-51. (Liu Chengjian, Luo Jie. A control method of traffic signals based on parameter fusion of Q-learning [J]. Computer Technology and Development, 2018, 28(11): 48-51.)

[8] 黄子蓉. 基于深度强化学习的多智能体协同研究[D]. 太原: 太原理工大学,2021. (Huang Zirong. Research of multi-agent cooperation based on deep reinforcement learning [D]. Taiyuan: Taiyuan University of Technology, 2021.)

[9] Grandinetti P, Canudas-de-Wit C, Carin F. Distributed optimal traffic light design for large-scale urban networks [J]. IEEE Trans on Control Systems Technology, 2019, 27(3): 950-963.

[10] 朱依婷,闫云,何兆成. 公交与社会车辆混合的中观交通建模与仿真[J]. 系统仿真学报,2022,34(9): 2019-2027. (Zhu Yiting, Yan Yun, He Zhaocheng. Mesoscopic modeling and simulation of mixed traffic flow of buses and vehicles [J]. Journal of System Simulation, 2022, 34(9): 2019-2027.)

[11] 邹长杰,郑皎凌,张中雷. 基于 GAED-MADDPG 多智能体强化学习的协作策略研究 [J]. 计算机应用研究,2020,37(12): 3656-3661. (Zou Changjie, Zheng Jiaoling, Zhang Zhonglei. Research on collaborative strategy based on GAED-MADDPG multi-agent reinforcement learning [J]. Application Research of Computers, 2020, 37(12): 3656-3661.)

[12] Merrouche F, Baha N. Depth camera based fall detection using human shape and movement [C]// Proc of IEEE International Conference on Signal and Image Processing. Piscataway, NJ: IEEE Press, 2016: 586-590.

[13] Newman M E J, Girvan M. Finding and evaluating community structure in networks [J]. Physical Review E, 2004, 69(2): 026112.

[14] Newman M E J. Fast algorithm for detecting community structure in networks [J]. Physical Review E, 2004, 69(6):06613.

[15] Wei Hua, Xu Nan, Zhang Huichu, *et al.* CoLight: learning network-level cooperation for traffic signal control [C]// Proc of the 28th ACM International Conference on Information and Knowledge Management. New York: ACM Press, 2019: 1913-1922.

[16] Urbanik T, Tanaka A, Lozner B. Signal timing manual [M]. Washington DC: National Academies Press, 2015.

[17] 顾夫挺. 基于多传感器的车辆检测方法及其可靠性分析[D]. 杭州: 浙江工业大学,2019. (Gu Futing. Vehicle detection method base on multi-sensors and its reliability [D]. Hangzhou: Zhejiang University of Technology, 2019.

[18] Robertson D I. TRANSYT: a traffic network study tool [EB/OL]. (1990-11). https://trid.trb.org/view/1199173.

[19] Mnih V, Badia A P, Mirza M. Asynchronous methods for deep reinforcement learning [EB/OL]. (2016-06-16). https://arxiv.org/pdf/1602.01783.pdf.

[20] Lillicrap T P, Hunt J J, Pritzel A. Continuous control with deep reinforcement learning [EB/OL]. (2019-07-05). https://arxiv.org/pdf/1509.02971v6.pdf.

[21] Littman M L. Markov games as framework for multi-agent reinforcement learning [C]// Proc of the 11th International Conference on Machine Learning. San Francisco, CA: Morgan Kaufmann Publishers, 1994: 157-163.

[22] Leone A, Rescio G, Garoppo A. A wearable EMG-based system pre-fall detector [J]. Procedia Engineering, 2015, 120: 455-458.

[23] Gupta S, Hazra R, Dukkipati A. Networked multi-agent reinforcement learning with emergent communication [EB/OL]. (2020-04-09). https://arxiv.org/pdf/2004.02780.pdf.

[24] Miller A J. Settings for fixed-cycle traffic signals [J]. Journal of the Operational Research Society, 1963, 14(4): 373-386.

[25] Arachi S, Khosravi A, Creighton D, *et al.* Optimal fuzzy traffic signal controller for an isolated intersection [C]// Proc of IEEE International Conference on Systems, Man, and Cybernetics. Piscataway, NJ: IEEE Press, 2014: 435-440.

[26] 朱明浩. 城市道路干线绿波协调控制研究及效果评价 [D]. 北京:北京工业大学,2016. (Zhu Minghao. Research and effect evaluation on green wave coordinated control of urban trunk roads [D]. Beijing: Beijing University of Technology, 2016.)

[27] Abdulhai B, Pringle R, Karakoulas G. Reinforcement learning for true adaptive traffic signal control [J]. Journal of Transportation Engineering, 2016, 129(3): 278-258.

[28] Zhang Kaiqing, Yang Zhuoran, Liu Han, *et al.* Fully decentralized multi-agent reinforcement learning with networked agents [C]// Proc of the 35th International Conference on Machine Learning. 2018: 5872-5881.

[29] Foerster J, Farquhar G, Afouras T. Counterfactual multi-agent policy gradients [C]// Proc of the 32nd AAAI Conference on Artificial Intelligence. Pola Alto, CA: AAAI Press, 2018.

[30] 曹洁,张玲. 自适应遗传算法的 multi-agent 交通信号优化控制 [J]. 计算机工程与应用,2016,52(13): 265-270. (Cao Jie, Zhang Ling. Multi-agent traffic signal control based on adaptive genetic algorithm [J]. Computer Engineering and Applications, 2016, 52(13): 265-270.)

(上接第 177 页)

[17] 朱宇宁,何利力. 基于启发式算法的混合云工作流调度算法[J]. 软件工程与应用,2021,10(6): 746-756. (Zhu Yuning, He Lili. Hybrid cloud workflow scheduling algorithm based on heuristic algorithm[J]. Software Engineering and Applications, 2021, 10(6): 746-756.)

[18] 马学森,许雪梅,蒋功辉,等. 混合自适应粒子群工作流调度优化算法[J/OL]. 计算机应用. (2022-04-07). https://kns.cnki.net/kcms/detail/51.1307.tp.20220406.1232.010.html. (Ma Xuesen, Xu Xuemei, Jiang Gonghui, *et al.* Hybrid adaptive particle swarm optimization algorithm for workflow scheduling [J/OL]. Journal of Computer Applications. (2022-04-07). https://kns.cnki.net/kcms/detail/51.1307.tp.20220406.1232.010.html.)

[19] Khan M, Jehangiri A I, Ahmad Z, *et al.* An exploration to graphics processing unit spot price prediction[J]. Cluster Computing, 2022, 22(5): 3499-3515.

[20] Veena K, Anand K C, Chandra P G. Amazon EC2 spot price prediction using regression random forests [J]. IEEE Trans on Cloud Computing, 2020, 8(1): 59-72.

[21] Liu Wenqiang, Wang Pengwei, Meng Ying, *et al.* Cloud spot instance price prediction using KNN regression[J]. Human-Centric Computing and Information Sciences, 2020, 10: article No. 34.

[22] Luan T, Luciana A, Pierre S, *et al.* Scheduling bag-of-tasks in clouds using spot and burstable virtual machines[J/OL]. IEEE Trans on Cloud Computing. (2021-11-08). https://doi.org/10.1109/TCC.2021.3125426.

[23] Brum R C, Sousa W P, Melo A C M A, *et al.* A fault tolerant and deadline constrained sequence alignment application on cloud-based spot GPU instances[C]// Proc of the 27th International Conference on Parallel and Distributed Computing. 2021: 317-333.

[24] Zohra A, Moulay Y H. Cloud spot price prediction approach using adaptive neural fuzzy inference system with chaos theory[J]. International Journal of Networking and Virtual Organisations, 2022, 26(1-2): 23-46.

[25] 王子健,卢政昊,潘纪奎,等. 最后期限动态分配的三步云工作流调度算法[J/OL]. 小型微型计算机系统. (2022). https://kns.cnki.net/kcms/detail/21.1106.TP.20211213.1445.018.html. (Wang Zijian, Lu Zhenghao, Pan Jikui, *et al.* A three-step cloud workflow scheduling algorithm based on dynamic deadline distribution [J/OL]. Journal of Chinese Mini-Micro Computer Systems. (2022). https://kns.cnki.net/kcms/detail/21.1106.TP.20211213.1445.018.html.)

[26] Topcuoglu H, Hariri S, Wu Minyou. Performance-effective and low-complexity task scheduling for heterogeneous computing [J]. IEEE Trans on Parallel and Distributed Systems, 2002, 13(3): 260-274.