一种权重自适应的强化学习云资源调度算法

李成严[1]， 孙 巍[1]， 唐立民[2]
(1. 哈尔滨理工大学 计算机科学与技术学院，哈尔滨 150080；2. 中国航发哈尔滨东安发动机有限公司，哈尔滨 150066)

摘 要：针对云资源调度问题，依据强化学习算法和云资源调度模型，建立了一种同时优化任务完成时间和运行成本的多目标云资源调度模型，并提出了一种权重自适应的强化学习算法对其进行求解。设计基于一维数组的状态空间和动作表示方法，压缩算法的存储开销，以提高算法效率。引入一种权重自适应的动作选择策略，首先根据执行动作所得到的回报值权重自适应，再通过启发式函数选择动作，以提高算法的收敛速度。使用 Cloudsim 仿真平台求解随机生成的数据，对使用权重自适应的 Q 学习算法进行测试。实验结果表明，权重自适应的启发式 Q 学习算法在寻优能力和负载均衡方面比遗传算法和 Q 学习算法好，在收敛速度方面比 Q 学习算法和启发式 Q 学习算法快。

关键词：云资源调度；权重自适应；强化学习；启发式函数
DOI：10.15938/j.jhust.2021.02.003
中图分类号：TP399 **文献标志码**：A **文章编号**：1007-2683(2021)02-0017-09

A Reinforcement Learning of Cloud Resource Scheduling Algorithm Based on Adaptive Weight

LI Cheng-yan[1]， *SUN Wei*[1]， *TANG Li-min*[2]
(1. School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China;
2. AECC Harbin Dongan Engine Co., Ltd, Harbin 150066, China)

Abstract: We considered the cloud computing resource scheduling problem, and proposed a multi-objective optimization mathematical model to optimize task completion time and running cost simultaneously. To solve the model, a reinforcement learning based on adaptive weight algorithm was presented. To improve the efficiency of the algorithm, we designed the state space and the action representation method based on one-dimensional array to compress the storage cost of algorithm. Furthermore, an adaptive weight was introduced in the strategy of action selection to improve the convergence speed, and the weight was updated adaptively according to the return value after each execution of the action, then the action selection strategy was determined by using the heuristic function. Extensive simulation experiments were made with randomly generating data on Cloudsim platform. Results show that the heuristics accelerate Q-Learning based on adaptive weight is better than genetic algorithm and Q-Learning algorithm in optimization ability and load balancing. The convergence rate of the heuristics accelerate Q-Learning based on adaptive weight is faster than Q-Learning algorithm and heuristics accelerate Q-Learning algorithm.

Keywords: cloud computing resource scheduling; adaptive weighting factor; reinforcement learning; heuristic function

收稿日期：2020-01-10
基金项目：国家自然科学基金(61772160)；黑龙江省教育厅科学技术研究项目(12541142).
作者简介：李成严(1972—)，男，教授，硕士研究生导师；
唐立民(1988—)，男，工程师.
通信作者：孙 巍(1995—)，女，硕士研究生，E-mail: sun1260012123@163.com.

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

0 引 言

云计算[1]是当前热门的一种服务模式，为用户提供动态可伸缩的廉价服务，同时减少与服务商之间的交互。用户通过按需付费的方式享受到服务商通过网络为其提供的资源共享池中的资源。

云资源调度[2]是指根据资源使用规则，不同的资源使用者按照规则在云服务平台进行资源调整的过程。在合理的资源调度优化算法对于提高云计算系统的综合性能是至关重要的。

调度中的服务质量约束包括运行成本、完成时间、安全性、可用性等。其中运行成本和完成时间分别是影响运营商和使用者满意度的关键因素。Yao 等[3]为了减少任务的完成时间，提出一种以最小化完成时间为目标的云资源调度模型。Zhang 等[4]为了提高运营商的满意度，研究了如何降低运营商的成本。Xu 等[5]针对在满足多个服务器提供高质量服务的同时，如何降低能耗这一问题，提出了一种新的数据中心调度方案，将提升服务质量与降低能源消耗这二者关系转化为利润与成本之间的关系。Jena 等[6]通过优化任务的等待时间来最大化虚拟机的吞吐量和保持任务优先级之间的平衡。在实际需求中，将减少执行时间和降低运行成本同时作为优化目标对于调度算法来说是至关重要的。本文以减少虚拟机运行成本和任务总完成时间为优化目标，建立了多目标云资源调度模型。

近年来，为求解云资源调度问题，国内外学者提出了许多智能启发式算法，如 Zhang 等[7]提出的蚁群算法，Gawanmeh 等[8]提出的遗传算法等。相较于上述算法，强化学习[9]作为一种与模型无关的具有学习能力的非监督式智能搜索算法，在云资源调度问题上具备较好的学习效果，因此尝试使用强化学习算法解决云资源调度问题。其中，Q 学习算法[10]对于调度问题表现更加稳定。Peng 等[11]设计了一种基于强化学习和排队论的任务调度方案，并在资源约束条件下优化任务调度，将强化学习应用于云工作流调度中利用状态集结技术加快学习进度。针对算法易陷入局部最优这一缺点，Bahrpeyma 等[12]将 Q 学习算法中的动作选择策略选为 ε - greedy 算法，通过 ε 调节探索与利用之间的平衡，Agent 以一定概率重新随机选择动作，跳出局部最优。Wu 等[13]，以解决任务调度时间长，负载不均衡的问题，虽然算法可以寻找到最优解，但仍存在收敛速度过慢问题。Reinaldo 等[14]提出了启发式 Q 学习算法，在传统 Q 学习的基础上加入一个影响行为选择的启发式函数，函数利用先前的经验，指导 Agent 进行动作选择，提高收敛速度，函数只改变动作的选择，并不改变 Q 值。为了进一步提高算法的收敛速度，本文考虑将权重因子与启发式函数相结合，依据 Agent 每次训练后的立即回报值，自动更新不同动作执行后的权重因子，从而确定动作选择策略，提高算法收敛速度。

本文提出一种权重自适应的启发式动作选择策略。在 ε - greedy 算法的基础上，引入结合了权重因子概念的启发式函数，提出基于权重自适应的启发式 Q 学习算法(heuristics accelerate q-Learning based on adaptive weight，WHAQL)，并将它作用于多目标云资源调度模型的求解上。

1 多目标云资源调度模型

多目标优化的云计算资源调度问题的数学模型定义如下:

1) 任务集合定义为 $T=\{t_1,t_2\cdots t_n\}$，共 n 个任务，其中 t_i 表示第 i 个任务，且 $t_i=\{t_i(1),t_i(2)\cdots t_i(p)\}$，表示第 i 个任务包含 p 个不同的属性。

2) 执行任务调度的虚拟机集合定义为 $VM=\{vm_1,vm_2\cdots vm_m\}$，共 m 台虚拟机($m<n$)，其中 vm_j 为第 j 台虚拟机，且 $vm_j=\{vm_j(1),vm_j(2)\cdots vm_j(q)\}$，为第 j 台虚拟机包含 q 个不同的属性。

3) 第 i 个任务分配给第 j 台虚拟机定义为

$$x_{ij}=\begin{cases}1 & t_i \text{ 分配给 } vm_j\\ 0 & \text{其他}\end{cases} \tag{1}$$

4) 第 i 个任务在第 j 台虚拟机的执行时间定义为

$$ect_{ij}=size_i/mip_j \tag{2}$$

其中，$size_i$ 为第 i 个任务的大小; mip_j 为第 j 台虚拟机的处理速度。

5) 第 j 台虚拟机的总运行时间定义为

$$vmT_j=\sum_{i=1}^{n}x_{ij}\times ect_{ij} \tag{3}$$

一个完整的调度方案 P_i 的总执行时间定义为

$$\text{Time}(P_i)=\max_{i=1}^{m}vmT_i \tag{4}$$

执行任务所消耗的总运行成本定义为

$$\text{Cost}(P_i)=\sum_{j=1}^{m}cst_j\times vmT_j \tag{5}$$

其中，cst_j 表示在单位时间内，第 j 台虚拟机执

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

行任务所消耗的资源成本。

多目标云资源调度的目标是使任务总执行时间更短,同时所需的运行成本更低。则云计算资源调度的多目标优化问题可以表示为

$$\min\ [\mathrm{Time}(P_i),\mathrm{Cost}(P_i)] \tag{6}$$

针对多目标优化问题,本文引入一种通过控制权值的方法求解多目标优化问题的函数。考虑到任务的执行时间和运行成本在数据规模上不统一,因此使用取对数的方法对数据进行标准化处理,最终调度 P_i 评价函数定义为

$$\mathrm{est}(P_i)=\omega\log\mathrm{Time}(P_i)+(1-\omega)\log\mathrm{Cost}(P_i) \tag{7}$$

其中: $\omega\in[0,1]$,表示用户对执行时间和运行成本的关注度,通过调整 ω 的大小来满足用户对执行时间和运行成本的不同需求。

虚拟机的最短执行时间与最长执行时间的比值定义为系统的负载均衡函数,公式如下

$$\mathrm{Load}=\frac{\min_{1\leqslant i\leqslant m}vmT_i}{\max_{1\leqslant i\leqslant m}vmT_i} \tag{8}$$

根据式(8)可知,Load 值越接近 1,系统的负载越均衡,对资源的利用率越高。

在改进 Q 学习算法中,将针对多目标云资源调度模型中的优化目标进行合理的算法设计,使改进后的 Q 学习算法更适用于解决此问题模型。

2　WHAQL 算法设计与分析

2.1　强化学习

强化学习(Reinforcement learning,简称 RL)是机器学习领域的一种通用算法,主要思想是 Agent 通过一个"试错"的过程,与环境进行交互得到回报值,以最大化回报值为目标进行学习。

Agent 通过执行动作与环境进行交互,当 Agent 执行一个动作后,会使得状态按某种概率转移到另一个状态; 同时,环境会根据回报函数反馈给 Agent 回报值。过程如图 1 所示,其中,t 为时间($t=0,1,2,3\ldots$); $S_t\in S$,S 为状态空间; $A_t\in A(S_t)$,$A(S_t)$ 为在状态 S_t 时的动作集; R_t 为 t 时刻的立即回报。

2.2　云资源调度的马尔科夫决策模型

2.2.1　马尔科夫决策优化方法

云资源调度问题可以用马尔科夫决策过程[15](Markov decision process,MDP)来描述。MDP 用五元组可以表示为$\{S,A,Q,R,\gamma\}$。

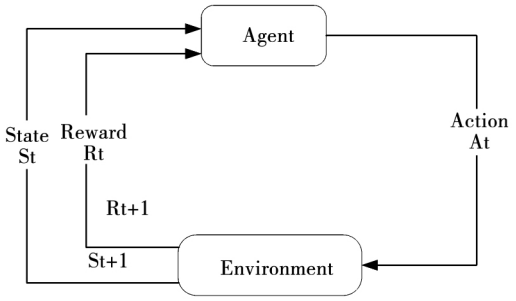

图 1　Agent 与环境交互图

Fig. 1　Interaction between agent and environment

S: 表示状态集,状态空间。

A: 表示动作集,动作空间。

Q: 表示状态转移函数,$Q(s_t,a_t)$ 表示在 t 时刻执行动作 a_t 后,状态在 $t+1$ 时刻由 s_t 转移为 s_{t+1} 所得到的 Q 值函数。

R: 表示立即回报函数。$r(s_t,a_t)$ 表示在状态 s_t 下执行动作 a_t 得到的立即回报值。

γ: 表示折扣因子,用于权衡长期回报与立即回报之间的重要程度。其中 $\gamma\in[0,1]$,当 $\gamma=0$ 时,代表只考虑立即回报,不考虑长期回报; 当 $\gamma=1$ 时,代表将长期回报与立即回报看得同等重要。

2.2.2　云资源调度的 MDP 模型

传统的云资源调度问题中,状态空间中的每一个状态会定义为一个任务匹配矩阵 $Matrix_{m\times n}$; 动作空间中,每个状态对应的动作与任务数相对应。比如说,第 i 个任务分配给第 j 台虚拟机,在任务匹配矩阵中,$m_{ij}=1$。

由于上述状态是定义为矩阵的形式,因此存在搜索空间过大问题。为了减少算法搜索空间,本文将状态空间中的每一个状态定义为数组形式,提高算法性能。本文云资源调度的 MDP 模型定义如下:

1) 状态空间由不同的状态 s 构成,由一个动态数组表示,其中状态 s 用一维数组表示,s 的下标表示任务序号,s 的值表示虚拟机序号,数组的维数为任务的个数,数组数值的最大值为虚拟机序号的最大值。比如 5 个任务分配 3 台虚拟机,则是一个维数为 5 的整形数组,每个元素的值表示任务分配到哪个虚机上执行。

2) 动作空间。将动作定义为整型变量,当执行将第 i 个任务分配给第 j 台虚拟机这一动作时,则将整型变量 j 赋值给状态 s 数组中第 i 个值。

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

例如一维数组 [1, 0, 0, 2, 1]，则表示第 0 个任务分配给 1 号虚拟机，第 1 个任务分配给 0 号虚拟机，完整对应关系如下:

TaskID = 0 VmID = 1

TaskID = 1 VmID = 0

TaskID = 2 VmID = 0

TaskID = 3 VmID = 2

TaskID = 4 VmID = 1

3) 立即回报定义了一种立即启发式回报函数，能够较精确的评价动作的好坏，为学习系统直接及时地提供回报信息，从而引导强化学习算法更快的学会最优策略。此处将回报函数定义为

$$r = \omega r\_ect + (1 - \omega)\, r\_cst \tag{9}$$

其中，r_ect 和 r_cst 分别定义为

$$r\_ect = Ect - T_i \tag{10}$$

$$r\_cst = Cst - C_i \tag{11}$$

T_i 和 C_i 分别表示当前状态下已经分配的任务的总执行时间和执行任务的总成本。Ect 和 Cst 都表示较大常数，此处将 Ect 设置为所有任务在所有虚拟机上的总执行时间，Cst 设置所有任务在所有虚拟机上的总成本。用 T_i 和 C_i 来评价当前状态下任务分配给第 i 台虚拟机这一动作的好坏。式(10)和式(11)将 T_i 和 C_i 最小化问题转化为回报值最大化问题，将任务调度的目标最小化完成时间与 Q 学习中最大化 Q 值函数联系起来。

4) 状态转移函数通过 2.3 节中 Q 学习算法中的 Q 值更新公式进行计算。

2.3 云资源调度的 Q 学习算法

Q 学习算法作为一种基于值函数的离线学习算法，其核心是建立一个 Q 表，表的行和列分别表示状态和动作，Q 表的 Q 值是用来衡量在当前状态下执行该动作的价值。主要通过迭代下述步骤进行训练学习。

1) 观察当前状态 s_t，选择合适的动作 a_t。

2) 状态 s_t 转移到下一状态 s_{t+1}，同时更新立即回报值 $r(s_t, a_t)$。

3) 在状态 s_t 下执行动作 a_t 的 Q 值函数定义为 $Q(s_t, a_t)$，更新公式如下:

$$Q(s_t, a_t) = (1 - \alpha)\, Q(s_t, a_t) + \alpha\left(r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a)\right) \tag{12}$$

其中，$\alpha \in (0, 1)$，表示学习速率。

Q 学习算法动作选择策略一般选用 ε - greedy 算法，Agent 以 ε 的概率随机选取动作，即探索过程，以 $1 - \varepsilon$ 的概率选择 Q 值最大的动作，即利用过程。策略概率分布定义为

$$\pi(s_t) = \begin{cases} \arg\max_{a_t} Q(s_t, a) & \text{if} \quad p \leqslant q \\ a_{\text{random}} & \text{otherwise} \end{cases} \tag{13}$$

其中，a_{random} 表示随机选择一个动作; $p, q \in [0, 1]$，p 值决定 Agent 进行探索概率，p 值越大，Agent 进行探索的概率就越小。

Q 学习算法的目标是形成一个策略 $\pi: S \rightarrow A$，通过 *Agent* 反复的训练过程实现 Q 值的最大化。

Q 学习算法(q-learning, QL)伪码如算法 1 所示。

算法 1 Q 学习算法伪码

① Initialize $\omega, \varepsilon, \alpha, \gamma$

② Initialize Q table

③ Repeat(for each episode):

④ Initialize state s_t

⑤ Repeat(for each step)

⑥ Select an action a using policy(13)

⑦ Execute action a_t observe r and s_{t+1}

⑧ Update the values of $Q(s_t, a_t)$ according to equation(12)

⑨ $s = s_{t+1}$

⑩ Until s is terminal

⑪ Until some stopping criterior is reached

2.4 WHAQL 算法设计

2.4.1 启发式 Q 学习

针对 Q 学习算法收敛的速度慢这一问题，启发式学习算法在传统 Q 学习算法的基础上，通过提供先验知识去指导 Agent 进行动作选择，动作选择策略如下所示:

$$\pi(s_t) = \begin{cases} \arg\max_a \left[Q(s_t, a) + H(s_t, a)\right] & \text{if} \quad p \leqslant q \\ a_{\text{random}} & \text{otherwise} \end{cases} \tag{14}$$

其中启发式函数 $H(s_t, a)$ 的更新公式定义为

$$H(s_t, a) = \begin{cases} \max_a Q(s_t, a) - Q(s_t, a_t) + \Delta & a_t = \pi^H(s_t) \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

其中: $\pi^H(s_t)$ 表示状态为 s_t 时，在启发式函数 H 的指导下选择的最优动作; Δ 表示启发系数。

启发式 Q 学习算法(heuristics accelerate

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

Q-learning,HAQL) 伪码如算法 2 所示。

算法 2　启发式 Q 学习算法伪码

① Initialize $\omega,\varepsilon,\alpha,\gamma$

② Initialize Q table, H table

③ Repeat(for each episode) :

④ Initialize state s_t

⑤ Repeat(for each step)

⑥ Update the values of $H(s_t,a_t)$ according to equation (15)

⑦ Select an action a using policy (16)

⑧ Execute action a_t observe r and s_{t+1}

⑨ Update the values of $Q(s_t,a_t)$ according to equation (12)

⑩ $s=s_{t+1}$

⑪ Until s is terminal

⑫ Until some stopping criterior is reached

在启发式 Q 学习算法中,Δ 为固定值,不随着不同的动作进行自动更新,为了进一步加强不同动作反馈得到的回报值对动作选择的指导,进而达到提高算法的收敛速度的目的,本文采用一种权重自适应的启发式动作选择策略。

2.4.2　权重自适应的启发式动作选择策略

权重自适应的启发式动作选择策略设计如下。

设计 G 表存储有权重因子的相关数据,元素为四元组 $\langle s_i,a_i,f(s_i,a_i),r_{\max}\rangle$。$s_i$ 和 a_i 分别表示需要更新权重因子的状态和动作;$f(s_i,a_i)$ 表示在状态 s_i 下执行动作 a_i 的权重因子;$r_{\max}$ 表示状态 s_i 下的最大回报值。更新规则为:

$$f(s_i,a_i)=\begin{cases}\dfrac{f(s_i,a_i)\,r_{\max}}{r_t} & \text{if}\quad a_i\neq a_t\\ 1 & \text{if}\quad a_i=a_t\end{cases}\tag{16}$$

其中: a_t 表示 Agent 在当前周期中在状态 s_i 下选择的动作; r_t 表示当前周期在状态 s_i 下执行动作 a_t 反馈的回报值。$f(s_i,a_i)$ 的值由 $r_{\max}$ 和 r_t 共同决定。当 $r_t>r_{\max}$ 时,即当前动作的回报值更大,该动作为目前最优选择,因此按式(17) 对权重因子 $f(s_i,a_i)$ 进行更新。通过对权重因子的不断更新,记录下不同动作的重要性。

为了使权重因子的大小对 Agent 的动作选择做出进一步的指导,将 $f(s_i,a_i)$ 与启发式函数相结合,将启发式函数的更新规则定义如下:

$$G(s_t,a)=\begin{cases}\max\limits_a Q(s_t,a)-Q(s_t,a_t)+U\dfrac{f(s_t,a_t)}{\sum_k f(s_t,a_k)} & a_t=\pi^f(s_t)\\ 0 & \text{otherwise}\end{cases}\tag{17}$$

其中: $\pi^f(s_t)$ 表示状态为 s_t 时,在权重因子函数 f 的指导下选择的最优动作; $\dfrac{f(s_t,a_t)}{\sum_k f(s_t,a_k)}$ 表示最大的权重因子与总权重因子的比值,通过这一数值的大小表示该动作的重要程度; U 值的大小表示权重因子对动作选择的影响程度的强弱,U 越大,权重因子对动作选择的指导性越强。

改进后的 Q 学习动作选择机制定义为

$$\pi(s_t)=\begin{cases}\arg\max\limits_{a_t}\left[Q_t(s_t,a_t)+G_t(s_t,a_t)\right] & \text{if}\quad q\leqslant p\\ a_{\text{random}} & \text{otherwise}\end{cases}\tag{18}$$

2.5　WHAQL 算法伪码

算法 3　改进 Q 学习算法伪码

① Initialize $\omega,\varepsilon,\alpha,\gamma,U$

② Initialize Q table, G table

③ Repeat(for each episode) :

④ Initialize state s_t

⑤ Repeat(for each step)

⑥ Select an action a using policy (18)

⑦ Execute action a_t observe r and s_{t+1}

⑧ Update the values of $Q(s_t,a_t)$ according to equation (12)

⑨ Update the values of $f(s_t,a_t)$ according to equation (16)

⑩ Update the values of $G(s_t,a_t)$ according to equation (17)

⑪ $s=s_{t+1}$

⑫ Until s is terminal

⑬ Until some stopping criterior is reached

2.6　WHAQL 算法收敛性分析

本节针对在 WHAQL 算法启发式动作选择策略的收敛性进行分析。

假设动作 a_1 是在状态 s^* 下记录的初始最优动作,Agent 通过学习得到了具有更大回报值的动作 a_2,根据式(16) 可知:

$f(s^*,a_1)<f(s^*,a_2)$

则 $\pi^f(s^*)=\max\limits_a f(s^*,a)=a_2$

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

根据式(17)可知:

情况 1: 当 $a=a_2$ 时,

$$G(s^*,a_2)=\max_a Q(s^*,a)-Q(s^*,a_2)+U\frac{f(s^*,a_2)}{\sum_k f(s^*,a_k)} \tag{19}$$

情况 2: 当 $a=a'$ 时,其中 a' 表示包括 a_1 在内,但 $a_1\neq a_2$ 的其他动作。

$$G(s^*,a')=0 \tag{20}$$

根据式(18)和式(19)可知:

$$Q(s^*,a_2)+G(s^*,a_2)=Q(s^*,a_2)+\max_a Q(s^*,a)-Q(s^*,a_2)+U\frac{f(s^*,a_2)}{\sum_k f(s^*,a_k)}=\max_a Q(s^*,a)+U\frac{f(s^*,a_2)}{\sum_k f(s^*,a_k)} \tag{21}$$

其中 $U\frac{f(s^*,a_2)}{\sum_k f(s^*,a_k)}>0$

根据式(21)可知:

$$Q(s^*,a')+G(s^*,a')=Q(s^*,a')+0=Q(s^*,a') \tag{22}$$

对比式(21)和式(22),显然

$$\max_a Q(s^*,a)+U\frac{f(s^*,a_2)}{\sum_k f(s^*,a_k)}>Q(s^*,a')$$

即

$$Q(s^*,a_2)+G(s^*,a_2)>Q(s^*,a')+G(s^*,a')$$

根据式(18)可知

$$\pi(s^*)=a_2$$

通过上述证明可知,基于权重自适应的启发式动作选择策略收敛在权重因子大的策略;再通过魏英姿等[16]已证明过的启发式 Q 学习算法的最优策略不变性以及 Q 值迭代收敛性,可以证明 WHAQL 算法最终必将收敛于最优策略。

3 仿真实验

为测试本文所提出改进 Q 学习算法在解决本文所设计的多目标云资源调度模型的效率,将模型与算法在 Cloudsim 仿真平台进行实验。

利用 Cloudsim 仿真平台随机生成数据集,将任务大小定义在区间 [60000,120000] 之间,虚拟机的处理速度定义在 [400,1200] 之间,通过式(2)可计算得到任务在不同虚拟机上的执行时间,虚拟机单位时间内的运行成本通过随机生成的虚拟机处理速度进行规则计算得到,在根据式(5)到虚拟机的运行成本。

本文测试的任务规模从 10 个开始依次递增 10 个,最多达到 50 个;虚拟机的数量设置为 5 个。实验中的主要参数设置如表 1 所示。

表 1 实验参数设置

Tab. 1 Experimental parameter setting

参数	取值
任务大小	60 000 ~ 120 000
任务数	10 ~ 50
虚拟机数	5
虚拟机处理速度	400 ~ 1 200
虚拟机内存(RAM)	2 048 MB
虚拟机带宽	10 000
数据中心数	1
主机数	1

WHAQL 算法的参数设置如表 2 所示。

表 2 算法参数设置

Tab. 2 Algorithm parameter setting

参数	取值
α	0.1
γ	0.9
ε	0.1
ω	0 ~ 1
U	1.5

在相同数据集和算法参数设置下,本文从 3 个方面对改进后的算法 WHAQL 在求解多目标云资源调度问题上进行验证。

3.1 算法寻优能力

在寻优能力方面,比较了以下 4 种算法:

1) 按顺序执行的调度方案,将任务按顺序依次分配在每个虚拟机上,即第一个任务分配给第一台虚拟机,第二个任务分配给第二台虚拟机等,用 Equ 表示。

2) 遗传算法[18](GA)。

3) Q 学习算法[17](QL)。

4) 基于自动更新权重的启发式 Q 学习算法(WHAQL)。

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

图 2 ~ 图 4 分别表示当 $\omega = 0.5$、$\omega = 1$、$\omega = 0$ 时，使用上述 4 种算法对不同任务规模下的模型进行多次求解取平均值的结果。

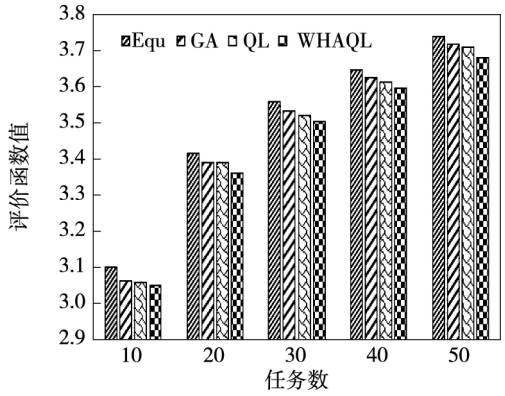

图 2　$\omega = 0.5$ 时，算法寻优能力对比图

Fig. 2　$\omega = 0.5$ Comparison chart of algorithm´s optimization ability

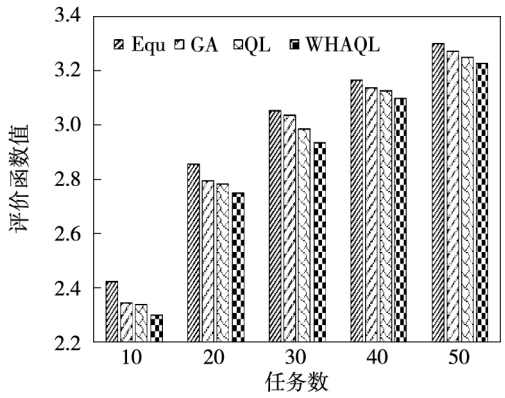

图 3　$\omega = 1$ 时，算法寻优能力对比图

Fig. 3　$\omega = 1$ Comparison chart of algorithm´s optimization ability

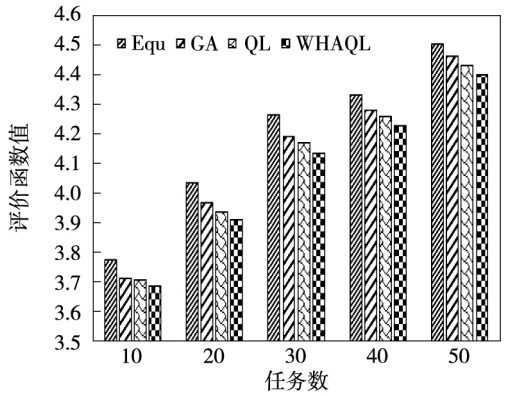

图 4　$\omega = 0$ 时，算法寻优能力对比图

Fig. 4　$\omega = 0$ Comparison chart of algorithm´s optimization ability

其中，横坐标表示任务规模，纵坐标表示评价函数值，评价函数值越小，算法的寻优能力越强。

通过图可以看出，当时间因子和成本因子均为 0.5 时，WHAQL 所得到的调度方案可以得到最小化的评价函数; 而当单独考虑时间或成本时，WHAQL 也能够在时间或成本的最小化上获得更好的结果。

3.2　算法收敛速度

在算法收敛速度方面，本文将基于权重自适应的启发式 Q 学习算法(WHAQL) 与 Q 学习算法[17] (QL) 和启发式 Q 学习算法[18] (HAQL) 进行对比。

图 5 表示当任务规模为 20，$\omega = 0.5$ 时，3 种算法迭代过程的对比图。总迭代次数设置为5 000次，每迭代 500 次为一个学习阶段，记录一次结果，共 10 个学习阶段。其中，横坐标表示任务规模，纵坐标表示评价函数值。

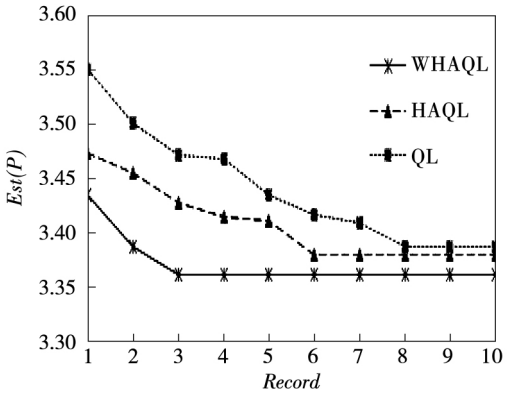

图 5　$\omega = 0.5$ 20task – 5vm 收敛过程对比图

Fig. 5　$\omega = 0.5$ 20task – 5vm Comparison of convergence process

观察图 5 中 3 种算法的迭代曲线可知，3 种算法在经历不同的迭代次数后均可达到收敛; WHAQL 和 HAQL 算法在启发式函数指导下，两者都较 QL 算法学习能力更强，更快达到收敛，而 WHAQL 算法在启发式函数中引入自动更新的权重因子，对动作的指导能力得到加强，使其在 3 个算法中收敛速度最快; 此外，WHAQL 在引入自动更新的权重因子后，将每次动作反馈的回报值更好地用于指导下一次动作的选择，使 Agent 更好地权衡了不同动作的重要程度，因此 WHAQL 相较于 HAQL 和 QL 算法，收敛到的评价函数值也更小。可见，改进后的算法 WHAQL 寻找最优解的能力也更强。

3.3　算法负载均衡

在算法负载均衡方面，将 WHAQL 算法与 Equ、

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

GA[8]和QL算法[17]进行对比。

图6表示当 $\omega = 0.5$ 时，不同任务规模的情况下，4种算法的负载均衡对比图。

其中，横坐标表示任务规模，纵坐标表示负载均衡值，负载均衡值越接近1，系统的负载越均衡。

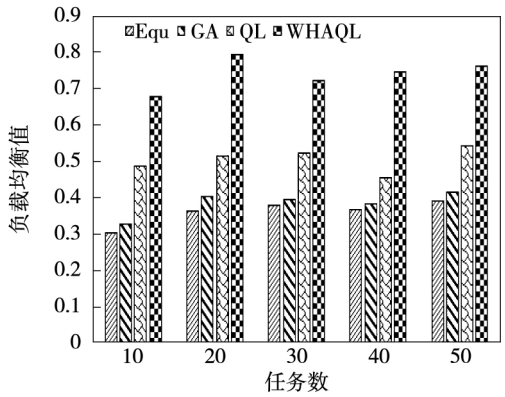

图6 不同任务规模下的负载均衡程度

Fig. 6 Load balancing degree under different task scales

由图6可知，WHAQL的负载均衡程度相较于其他3种算法效果更好，这证明WHAQL不仅对资源有更高的利用率，还可以有效减轻虚拟机的工作负载。

通过上述实验可知，本文提出的WHAQL在寻优能力上优于其他几种算法，相较于QL和HAQL也具备更快的收敛速度。将WHAQL应用于多目标云资源调度模型，使任务的完成时间更短、虚拟机的运行成本更低，同时有效减轻虚拟机工作负载。从整体上提高了云资源调度的综合性能。

4 结 论

本文将任务的完成时间和虚拟机的运行成本同时作为优化目标，建立了多目标云计算资源调度模型，提出了一种基于权重自适应的启发式Q学习算法(WHAQL)。WHAQL在启发式强化学习的基础上引入了权重因子，进一步加强了对Agent动作选择的指导，提高算法的收敛速度的同时，也提高了算法的寻优能力。实验证明WHAQL有效地提高了云资源调度的整体性能。

在未来研究工作中，主要研究如何对Q学习算法中的状态空间进行整合优化，使其更适用于解决未来更大规模的云资源调度问题。

参 考 文 献:

[1] SINGH B, DHAWAN S, Arora A, et al. A View of Cloud Computing [J]. International Journal of Computers & Technology, 2013, 4(2): 387.

[2] ZHAN Zhihui, LIU Xiaofang, GONG Yuejiao, et al. Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches [J]. Acm Computing Surveys, 2015, 47(4): 1.

[3] YAO Guangshun, DING Yongsheng, JIN Yaochu et al. Endocrine-based Coevolutionary Multi-swarm for Multi-objective Workflow Scheduling in a Cloud System [J]. Soft Computing, 2017, 21(15): 4309.

[4] ZHANG Yi, SUN Jin. Novel Efficient Particle Swarm Optimization Algorithms for Solving QoS-demanded Bag-of-tasks Scheduling Problems with Profit Maximization on Hybrid Clouds [J]. Concurrency and Computation-Practice & Experience, 2017, 29(21): 424.

[5] XU Song, LIU Lei, CUI Lizhen, et al. Resource Scheduling for Energy-Efficient in Cloud-Computing Data Centers [C] // 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE, 2018: 774.

[6] JENA U K, DAS P K, KABAT M R. Hybridization of Meta-heuristic Algorithm for Load Balancing in Cloud Computing Environment [J]. Journal of King Saud University-Computer and Information Sciences, 2020, 2(1): 1.

[7] ZHANG Liumei, WANG Yichuan, ZHU Lei, et al. Towards Energy Efcient Cloud: an Optimized Ant Colony Model for Virtual Machine Placement [J]. Journal of Communications and Information Networks, 2016, 1(4): 116.

[8] GAMANMEH A, PARVIN S, ALWADI A. A Genetic Algorithmic Method for Scheduling Optimization in Cloud Computing Services [J]. Arabian Journal for Science & Engineering, 2017(5): 1.

[9] 陈学松，杨宜民. 强化学习研究综述 [J]. 计算机应用研究，2010，27(8)：2834.
CHEN Xuesong, YANG Yimin. Reinforce-ment Learning: Survey of Recent Work. [J]. Application Research of Computers, 2010, 27(8): 2834.

[10] KLIDBARY S H, SHOURAKI S B, KOURABBASLOU S S. Path Planning of Modular Robots on Various Terrains Using Q-learning Versus Optimization Algorithms [J]. Intelligent Service Robotics, 2017, 10(2): 121.

[11] PENG Zhiping, CUI Jingling ZUO Qirui, et al. Random Task Scheduling Scheme Based on Reinforcement Learning in Cloud Computing [J]. Cluster Computing, 2015, 18(4): 1595.

[12] BAHRPEYMA F, HAGHIGHI H, ZAKEROLHOSSEINI A. An Adaptive RL Based Approach for Dynamic Resource Provisioning in Cloud Virtualized Data Centers [J]. Computing, 2015, 97(12): 1209.

[13] WU Jiahao, PENG Zhiping, CUI Delong, et al. A Multi-object Optimization Cloud Workflow Scheduling Algorithm Based on Re-

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

inforcement Learning [J]. Intelligent Computing Theories and Application, 2018: 550.

[14] BIANCHI R A C, RIBEIRO C H C, COSTA A H R. Heuristically Accelerated Q-Learning: A New Approach to Speed Up Reinforcement Learning [C] // Advances in Artificial Intelligence-SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29-October 1, 2004: 245.

[15] SHAH S M, BORKAR V S. Q-learning for Markov Decision Processes with a Satisfiability Criterion [J]. Systems & Control Letters, 2018, 113: 45.

[16] 魏英姿,赵明扬. 强化学习算法中启发式回报函数的设计及其收敛性分析 [J]. 计算机科学,2005(3):190.
WEI Yingzi, ZHAO Mingyang. Design and Convergence Analysis of a Heuristic Reward Function for Reinforcement Learning Algorithms [J]. Computer Science, 2005(3): 190.

[17] CHOI J, REALFF M J, LEE J H. A Q-Learning-based Method Applied to Stochastic Resource Constrained Project Scheduling with New Project Arrivals [J]. International Journal of Robust & Nonlinear Control, 2010, 17(13): 1214.

[18] YANG Bohong, LU Hong, LI Baogen, et al. A Novel Experience-based Exploration Method for Q-Learning [J]. Communications in Computer and Information Science, 2018, 901: 225.

[19] 李成严, 曹克翰, 冯世祥, 等. 不确定执行时间的云计算资源调度 [J]. 哈尔滨理工大学学报, 2019, 24(1): 85.
LI Chengyan, CAO Kehan, FENG Shixiang, et al. Resource Scheduling with Uncertain Execution Time in Cloud Computing [J]. Journal of Harbin University of Science and Technology, 2019, 24(1): 85.

[20] ZHONG Shan, LIU Quan, ZHANG Zongzhang, et al. Efficient Reinforcement Learning in Continuous State and Action Spaces with Dyna and Policy Approximation [J]. Frontiers of Computer Science, 2019 (1): 106.

[21] 罗智勇,朱梓豪,尤波,等. 基于串归约的时间约束下工作流精确率优化算法 [J]. 哈尔滨理工大学学报,2018,23(5):68.
LUO Zhiyong, ZHU Zihao, YOU Bo, et al. Optimization Algorithm of Workflow's Accuracy Based on Serial Reduction Under Constraint time [J]. Journal of Harbin University of Science and Technology, 2018, 23(5): 68.

(编辑: 王　萍)

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net