DOI:10.19651/j.cnki.emt.2210891

基于深度强化学习的多目标边缘任务调度研究*

盛 煜[1] 朱正伟[1] 朱晨阳[2] 诸燕平[1]
(1.常州大学微电子与控制工程学院 常州 213000; 2.常州大学计算机与人工智能学院 常州 213000)

摘 要: 针对深度强化学习在边缘计算环境下的多目标任务调度时存在优化效果差等问题,提出了一种新的基于改进的竞争深度双 Q 网络的多目标任务调度算法(IMTS-D3QN)。首先将深度双 Q 网络对目标中的最大操作分解为动作选择和动作评估,以消除过高估计;采用立即奖励经验样本分类方法,对经验样本按照重要性程度分类存储,训练时选取更多重要性程度高的经验样本,提高了实际样本的利用率,加快了神经网络的训练速度。然后,通过引入竞争网络结构对神经网络进行优化。最后,采用软更新方法提高算法的稳定性,并采用动态 ε 贪婪指数递减法寻找最优策略。通过不同线性加权组合得出帕累托最优解,达到响应时间和能耗最小化。实验结果表明,IMTS-D3QN 算法与其他算法相比,在不同任务数下响应时间与能耗上具有明显的优化效果。

关键词: 边缘计算;任务调度;多目标;深度强化学习
中图分类号: TP393 **文献标识码:** A **国家标准学科分类代码:** 520.99

Research on multi-objective edge task scheduling based on deep reinforcement learning

Sheng Yu[1] Zhu Zhengwei[1] Zhu Chenyang[2] Zhu Yanping[1]
(1. School of Microelectronics and Control Engineering, Changzhou University, Changzhou 213000, China;
2. School of Computer Science and Artificial Intelligence, Changzhou University, Changzhou 213000, China)

Abstract: Aiming at the problems of unstable convergence and poor optimization effect in the multi-objective task scheduling of deep reinforcement learning in the edge computing environment, a new multi-objective task scheduling algorithm based on an improved competitive deep double-Q network (IMTS-D3QN) was proposed. First, the selection and calculation of the target Q value are decoupled by the deep double-Q network to eliminate overestimation, the immediate reward experience sample classification method is adopted to extract experience samples from the experience replay unit, which improves the utilization rate of actual samples, which speeds up the training speed of the neural network. Then, the neural network is optimized by introducing competing network structures. Finally, the soft update method is used to improve the stability of the algorithm, and the dynamic ε-greedy exponential decreasing method is used to find the optimal strategy. The Pareto optimal solution is obtained through different linear weighting combinations to minimize the response time and energy consumption. The experimental results show that, compared with other algorithms, the IMTS-D3QN algorithm has obvious optimization effect in response time and energy consumption under different number of tasks.

Keywords: edge computing; task scheduling; multi-objective; deep reinforcement learning

0 引 言

云计算的出现提供了足够的计算资源,但大量的数据量交付到云端会导致网络拥堵和不可预测的能耗,无法满足低延迟低功耗的要求,并降低体验质量。新兴的边缘计算技术克服了云计算的缺点。

边缘计算使得各种物联网应用和服务在网络边缘执行,而不是交付到远程云,从而减少了响应时间与过程中的能耗。然而,边缘服务器的计算、存储和网络资源都是有限的,因此,任务调度对于最大化体验质量至关重要。根据要解决的用户需求的不同,调度过程可描述为一个多目标的问题。因此,本文考虑同时对处理任务的响应时间和能耗进行优化。

收稿日期:2022-07-30
* 基金项目:常州市重点研发计划(CJ20210123)项目资助

针对多目标这类问题,传统的解决方法是采用多种元启发式算法,例如,文献[1-3]提出了一种利用多目标优化和启发式算法解决任务调度的资源分配新模型。启发式算法在某些条件下性能良好,但难以适应复杂多变的边缘云环境。

在此背景下,有研究工作[4]利用强化学习 Q 学习算法的优秀决策能力来解决复杂联邦边缘云环境下的服务调度问题,达到减少服务迁移开销的效果。然而,由于状态空间表示的限制,RL 难以处理高维问题。文献[5]提出了一种基于状态-动作-奖励-状态-动作(state-action-reward-state-action,SARSA)在线学习算法的移动边缘计算资源分配框架。基于 SARSA 算法,他们做出了最小化能源消耗和计算延迟的最优卸载决策。

近年来,深度强化学习在解决大规模复杂问题方面取得了很大的进展,如自然语言处理、游戏和机器人控制。因此,基于深度强化学习的任务调度研究成为一个新的研究方向。文献[6]基于深度 Q 网络(deep Q network,DQN)算法提出了一种云系统中的在线资源调度框架。通过调整不同奖励优化目标的比例,解决了能耗和服务质量两个优化问题。

本文对以上研究的基础上对边缘云计算的任务调度问题进行了详细的研究。提出一种新的基于改进的竞争深度双 Q 网络的边缘云任务调度与分配框架,然后采用深度强化学习(deep reinforcement learning,DRL)策略实现了边缘云任务调度优化,实现了任务响应时间和服务器能耗的多目标优化,最后通过线性加权法调整不同组合的目标权重来平衡任务响应时间与处理任务过程中的能耗。

1　系统模型和问题描述

1.1　系统模型

物联网应用产生的数量庞大且计算密集型的任务,很难在本地设备上执行,被交付给部署在靠近终端设备的网络边缘的服务器,标记为 $\{S_1,S_2,\cdots,S_i\}(i=1,2,\cdots,m)$。边缘服务器上配置了虚拟机 VM,记为 $VM=\{vm_1,vm_2,\cdots,vm_j\}(j=1,2,\cdots,n)$。各虚拟机之间独立运行。

在本文中,任务之间具有依赖关系,此任务类型通常使用有向无环图(directed acyclic graph,DAG)任务表示,定义为 $Task=\{t_1,t_2,\cdots,t_l\}(l=1,2,\cdots,k)$。图 1 为 5 个任务的简单 DAG 模型,在任务调度和资源配置过程中,系统对 DAG 任务进行解耦,打破了子任务之间的依赖关系。满足条件的子任务被调度到调度队列中等待。系统按照先到先得的策略为等待队列中的任务配置虚拟机资源。

1.2　问题描述

DAG 任务中的子任务表示为:

$$t_l=<mi_l,cpu_l,mips_l> \tag{1}$$

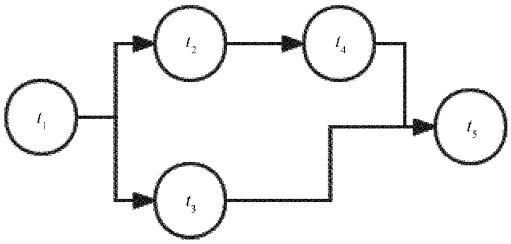


图 1　DAG 任务模型

其中,mi_l 表示子任务的长度(即数百万条指令的个数),cpu_l 表示子任务的 CPU 占用率,$mips_l$ 表示子任务的指令数。

边缘服务器 S_i 中配置的虚拟机表示为:

$$vm_j^i=<mips_j^i,cpu_j^i> \tag{2}$$

式中:$mips_j^i$ 表示虚拟机上每秒执行多少条指令,cpu_j^i 表示虚拟机上 CPU 的大小。

系统中每个服务器可以加载不同数量的虚拟机,单个服务器不能加载超过 4 个虚拟机。假设系统将任务 t_1 在等待时间 $T_{wa}=0$ 时分配给拥有 2 个虚拟机的边缘服务器 vm_1 上。当任务 t_2 也在 $T_{wa}=0$ 到达时,此时它有两个调度选择,第 1 个选择是等待 t_1 完成后进行部署,对应的等待时间为 t_1 的执行时间 T_{ex},第 2 个选择是将 t_2 部署到 vm_2 上,无等待时间。若所有虚拟机都被占用,则寻找目标最优的虚拟机进行等待部署。

因此,子任务 t_l 分配给 vm_j^i,子任务的执行时间[7]和等待时间表示为式(3)和(4):

$$T_{ex_{lj}^i}=\frac{mi_l}{cpu_l\times mips_j^i} \tag{3}$$

$$T_{wa_{lj}^i}=\sum_{k=1}^{C}T_{ex_{kj}} \tag{4}$$

其中,C 表示 vm_j^i 中等待队列中的任务数。在这里,任务的响应时间[8]被定义为式(5):

$$T_{re_{lj}^i}=T_{ex_{lj}^i}+T_{wa_{lj}^i} \tag{5}$$

这些虚拟机是服务器资源的基本单位。服务器的能量消耗不是线性的,最优能量利用率在 70%[9]左右,然后显著增加。为了尽量减少服务器的消耗,服务器端利用率应稳定在最优值附近。因此,建立了能耗模型,并通过服务器当前的使用状态来估算当前的能耗。

服务器的资源利用率定义如式(6)所示。

$$U_m=\frac{cpu_{total}^l}{cpu_{total}^{S_i}} \tag{6}$$

其中,分母 $cpu_{total}^{S_i}$ 表示服务器上虚拟机的总 CPU 大小,分子 cpu_{total}^l 表示当前在服务器上运行的任务总 CPU 大小。

服务器的功率由两部分组成:静态功率 P_{wr_S} 和动态功率 P_{wr_D}。如式(7)所示。其中,静态功率在服务器运行时是固定的,动态功率随着利用率的变化而变化[10]。

$$P_{wr_S}=\begin{cases}0(U_m=0)\\const(U_m>0)\end{cases}$$
$$P_{wr_D}=\begin{cases}\alpha\times U_m(U_m<\tilde{U}_m)\\\alpha\times\tilde{U}_m+(U_m-0.7)^2\beta(U_m\geqslant\tilde{U}_m)\end{cases}\tag{7}$$

其中，U_m 为服务器 S 的利用率，$\alpha=0.5$，$\beta=10$，$const=1$，$\tilde{U}_m$ 最佳利用率是 0.7。服务器 S 的总能耗 $P_{wr_{Ser}}$ 为静态功率 P_{wr_S} 和动态功率 P_{wr_D} 之和，如式(8)所示。

$$P_{wr_{Ser}}=P_{wr_S}+P_{wr_D}\tag{8}$$

问题描述的符号如表 1 所示。

表 1　符号表

符号	说明
m	边缘服务器数量
n	虚拟机数量
k	任务数量
C	vm_j^i 中等待队列中的任务数
S_i	边缘服务器 $(i=1,2,\cdots,m)$
t_l	子任务 $(l=1,2,\cdots,k)$
mi_l	子任务 t_l 的长度
cpu_l	子任务 t_l 的 CPU 占用率
$mips_l$	子任务 t_l 的指令数。
vm_j^i	边缘服务器 S_i 中配置的虚拟机 vm_j
$mips_j^i$	vm_j^i 每秒执行多少条指令
cpu_j^i	vm_j^i 上 CPU 的大小
VM	虚拟机资源，$\{vm_1,vm_2,\cdots,vm_j\}(j=1,2,\cdots,n)$
$T_{ex_{lj}^i}$	子任务 t_l 分配给 vm_j^i 的执行时间
$T_{wa_{lj}^i}$	表示 vm_j^i 等待队列中的时间
$T_{re_{lj}^i}$	执行时间与等待队列中的时间之和
cpu_{total}^l	在服务器 S_i 上运行的任务所占 CPU 总大小
$cpu_{total}^{S_i}$	服务器 S_i 上虚拟机的总 CPU 大小
U_m	服务器的资源利用率
P_{wr_S}	服务器静态功率
P_{wr_D}	服务器动态功率
$P_{wr_{Ser}}$	服务器总能耗

2　深度强化学习

2.1　传统 DQN 算法

DQN 算法[11]是深度强化学习的经典模型，它将深度卷积神经网络与传统 Q 学习算法相结合。学习代理在所提供的环境中采取行动，以最大限度地优化能耗与响应时间。代理在完全未知的环境中通过反复尝试寻找最优行为。根据所选动作获得奖励，在这个奖励功能的帮助下以交互方式学习。DQN 算法[7]的关键技术如下：

1)经验回放：在 Q 学习中，从状态、行动、奖励和下一个状态中学习，做出决定，然后丢弃它们。然而，可能会有许多罕见的经验，可能对模型非常重要。在经验回放中，把学到的东西储存在记忆缓冲器中，试着回忆那些学到的东西并从中学习。

2)目标网络：DQN 中使用两个结构相同但是参数不同的网络，预测 Q 估计的评估网络使用最新的参数，而预测 Q 现实的目标网络参数使用的是之前评估网络的输出，用来评估当前状态动作对的值函数，每经过一定次数的迭代，将评估网络的参数复制给目标网络。一定程度上提高了算法的稳定性。

3)ε 贪婪指数：学习代理以 ε 的概率从所有动作中的随机抽取一个动作，在整个训练过程中，代理会逐渐增加选择最高价值的动作概率，利用好的策略返回高额奖励。

DRL 代理的目标是最大化预期累积折扣报酬。该模型采用深度卷积神经网络拟合最优行为价值函数，如式(9)所示。

$$Q(s,a)=E_{s'}[r+\gamma\max_{a'}Q(s',a')\mid s,a]\tag{9}$$

损失函数如式(10)所示，使用策略梯度下降来更新参数 θ 获得最优 Q 值。

$$L_i(\theta_i)=E[(r+\gamma\max_{a'}Q(s',a';\theta_i^-)-Q(s,a;\theta_i))^2]\tag{10}$$

2.2　IMTS-D3QN 算法

IMTS-D3QN 算法在 DQN 算法基础上进行了更深层次的提高，引入了如下 5 个部分。

1)双深度 Q 网络

DQN 中通过最大运算符来选择和评估动作。这使得它更有可能选择高估的值，从而导致过度乐观的价值估计。

在双 Q 学习[12]中，通过随机分配经验来学习两个值函数，从而更新两个值函数中的一个，得到两组权重。并且对于每次更新，一组权重用于确定贪心策略，另一组用于确定其值。为了清楚比较，可以解开 Q 学习中的选择和评估，将 DQN 的目标重写则可表示为双 Q 学习误差如式(11)所示，损失函数如式(12)所示。

$$Y_t^{DDQN}=R_{t+1}+\gamma Q(s_{t+1},\mathrm{argmax}Q(s_{t+1},a;\theta_t);\theta_t')\tag{11}$$

$$L(\omega)=E[(Y_t^{DDQN}-Q(s,a,\theta_t))^2]\tag{12}$$

双 Q 学习的思想是通过将目标中的最大操作分解为动作选择和动作评估来减少高估。虽然没有完全解耦，但 DQN 架构中的目标网络提供了第 2 个价值函数的自然候选者，而无需引入额外的网络。总之，第 2 个网络的权重 θ_t' 替换为目标网络的权重，用于评估当前的贪心策略。

2)竞争网络

竞争网络[13]自动生成状态值函数和优势函数的单独估计，无需任何额外监督。直观地说，竞争架构可以了解哪些状态是有价值和没有价值的，而不必了解每个状态的每个动作的效果，可以更准确的评估 Q 值。

竞争架构用一个单一的深度模型来表示价值函数 $V(s)$ 和优势函数 $A(s,a)$,其输出将两者结合起来产生一个状态-动作值 $Q(s,a)$。定义如式(13)所示。

$$A^{\pi}(s,a)=Q^{\pi}(s,a)-V^{\pi}(s) \tag{13}$$

价值函数 V 衡量的是处于特定状态 s 的好坏。然而,Q 函数测量在此状态下选择特定操作的价值。现在,使用优势的定义,可能会尝试如式(14)的聚合模块:

$$Q(s,a;\theta,\alpha,\beta)=V(s;\theta,\beta)+A(s,a;\theta,\alpha) \tag{14}$$

其中,θ 表示卷积层的参数,而 α 和 β 是两个全连接层流的参数。

由于网络直接输出 Q 值,因此无法知道状态值 V 和动作 A,为了反映这种可识别性,集中了动作 A,以确保性能,同时提高优化的稳定性,其中 a' 是所有可能的动作,$\text{avg}A(s,a;\theta,\alpha)$ 是所有动作优势函数的平均值。修正后的 Q 值用式(15)表示。

$$Q(s,a;\alpha,\beta)=V(s;\theta,\beta)+(A(s,a;\theta,\alpha)-\text{avg}A(s,a';\theta,\alpha)) \tag{15}$$

3)目标网络的软更新

传统的 DQN 通常使用硬更新,即每 C 步将评估网络权重复制给目标网络。目标网络更新间隔 C 越大,算法就会越稳定,目标网络更新频率越慢,算法收敛速度会越慢。本文提出了一种替代的软更新思想,也可以称为指数平均移动(EMA)。即引入一个学习率 τ,将旧的目标网络参数 θ_i' 和新的对应网络参数 θ_i 做加权平均,然后赋值给目标网络,即式(16)。确保目标网络在每次迭代中都会更新,相当于网络更新间隔为 1。

$$\theta_{i+1}'=(1-\tau)\theta_i'+\tau\theta_i=\theta_i'+\tau(\theta_i-\theta_i'),0<\tau\ll 1 \tag{16}$$

这样,目标网络的参数变化较小,计算出的目标标签值变化相对平稳。这样,即使目标网络在每次迭代时都被更新,算法也能保持一定程度的稳定性。软区间更新系数 τ 越小,算法越稳定。适当的软区间更新系数可以使 DQN 算法训练既稳定又快速。

4)动态 ε 贪婪指数递减法

在本文中,算法采用动态 ε 贪婪策略[14]进行选择。在训练阶段,传统的强化学习算法随机选择概率为 ε 的动作,选择概率为 $1-\varepsilon$ 的最优动作,其中 ε 为固定的值。如果太小,环境将无法得到充分的探索,导致过度探索和探索不足。如果太大,环境虽然得到了充分的探索,但是会导致随机选择行为概率很大,选择最优行为的概率就会很小。因此,提出了一种指数下降动态贪婪方法来解决这个问题,如式(17)所示。该方法解决了深度强化学习所面临的探索困境,确保了由于环境的不确定性,当训练的数量很大且价值函数已经可以很好地逼近时,需要更大的探索概率。

$$\varepsilon=\frac{(t+\delta)^3}{x}\mathrm{e}^{\sqrt{t+\delta}} \tag{17}$$

式中:t 表示训练次数,δ 表示设计偏移量,x 是一个随环境变化的变量。

5)立即奖励经验样本分类

神经科学研究表明啮齿动物在清醒或睡眠期间海马体中会重播先前经历的序列,与奖励相关的序列会更频繁地被重播。受到该观点启发,使用两个经验缓冲区对经验样本进行分类[15-16]。立即奖励值大的经验样本重要性程度更高,将立即奖励值大的经验样本放入高回报经验缓冲池其余存入经验缓冲池中。

提出的 IMTS-D3QN 算法描述如算法 1 所示。

算法 1:改进的竞争深度双 Q 网络的多目标任务调度算法

输入:任务中的所有任务

输出:响应时间和能耗的结果

1. 初始化 Q 网络和目标 Q 网络,高回报经验缓冲池 D_{HM},经验缓冲池 D_H,折扣因子 γ,更新参数 τ

2. For 每批 t_l 在时间 $Time=1,\cdots,T$ do

3. 观察状态 s_t 并通过动态 ε 选择 a_t,观察下一状态 s_{t+1},获取奖励 r_1,r_2

4. 计算 $r=\alpha\times r_1+(1-\alpha)\times r_2$

5. 存储经验 (s_t,a_t,r_t,s_{t+1}) 以奖励 r 分类分别存入 D_H 与 D_{HM} 中

6. 从 D_H 中选取部分经验样本,D_{HM} 中选取剩余部分样本

7. 计算目标值

$y=r+\gamma Q(s_{t+1},\text{argmax}_a Q(s_{t+1},a;\theta_t);\theta_t')$

8. 使用均方误差损失函数进行反向传播,以更新 Q 网络参数

$L(\omega)=E[(y-Q(s,a,\theta_t))^2]$

9. 软更新目标网络 $\theta_{i+1}'=\theta_i'+\tau(\theta_i-\theta_i')$

10. end for

2.3 算法设计

1)状态空间:任务调度过程中,代理将得到每个边缘服务器当前的资源状态信息和任务状态信息,所以状态空间是上述两种信息组成的。如式(18)所示。

$$state=\{mi_t,cpu_t,mips_t,mips_j^i,cpu_j^i,Twa_{ij}^i,U_m\} \tag{18}$$

2)行为空间:代理需要从当前的所有服务器中选择一个服务器,因此,行为空间被定义为式(19):

$$action=\{S_1,S_2,\cdots,S_i\} \tag{19}$$

3)奖励函数:为了能够通过深度 Q 学习在实时输入下自动生成长期最优解,需要通过奖励函数反馈决策的每一步。通过考虑最小化任务响应时间和能耗为优化目标。其函数定义如式(20)与(21)所示。

$$R_t=ART_{IJ}=\frac{\sum_{i=1}^{I}T_{re_{ij}^i}}{I} \tag{20}$$

$$R_e = \frac{P_{wr_{Ser}^{t}} - P_{wr_{Ser}^{t-1}}}{P_{wr_{Ser}^{t}}^{1}} \tag{21}$$

式(20)表示平均响应时间，其中 I 表示任务数，式(21)比较了在时间 t 和 $t-1$ 时的能耗。

将式(20)与(21)标准化，使这两个回报值介于 0 和 1 之间。使用最小最大归一化方法对两种奖励进行归一化。构造奖励函数如式(22)所示。

$$R = \alpha \times R_t + (1-\alpha) \times R_e (\alpha \in [0,1]) \tag{22}$$

3　仿真实验及结果分析

本节通过大量的实验对所提出的任务调度算法进行了评价。所有的实验都是在 Ubuntu 18.04LTS 操作系统、Python3.6、Pytorch 1.0 运行得到的。进一步比较了提出的算法与其他算法。

3.1　实验内容与参数设置

本文使用 Alibaba-Cluster-trace-v2018 作为测试案例。文件 batch_instance.csv 中记录(即任务)的内容包括提交时间、任务长度、CPU 利用率等。为了简化实验，选择了其中几个小时的数据，筛选出需要的信息。每次最多有 30 个任务提交到系统中。分别测试了任务数为 5 000、10 000、15 000 的情况。

为了评价 IMTS-D3QN 算法的性能，将其与随机调度(random)算法、轮询调度(round-robin，RR)算法、异构最早完成时间 (heterogeneous earliest finish time，HEFT)算法和传统 DQN 算法进行了比较。Random 是一种常用的调度算法，将用户提交的任务随机分配给虚拟机。RR 是一种将任务依次分配给各个虚拟机的有效调度算法。HEFT 是一种静态调度算法，将所有任务都安排在能够使它最早完成的虚拟机上执行。本实验中传统 DQN 算法引入了深度学习的神经网络，目标网络和经验回放机制。

IMTS-D3QN 算法中参数如表 2 所示。

表 2　算法参数

参数	值
折扣因子 γ	0.7
经验池大小	5 000
高回报经验池大小	5 000
小批量大小	32
学习率	0.003
更新参数 τ	0.05
设计偏移量 δ	25

虚拟机类型以及相关参数如表 3 所示。

边缘服务器中配置的不同虚拟机如表 4 所示。

3.2　仿真结果与分析

IMTS-D3QN 算法对多目标优化问题进行线性加权求解时，调整不同的权重组合，权重组合从 0.1～0.9 共 9 组，

表 3　虚拟机参数

虚拟机类别	CPU 核心	处理速度/MIPS
1	1	2 500
2	1	2 000
3	1	1 500
4	1	1 000
5	1	800

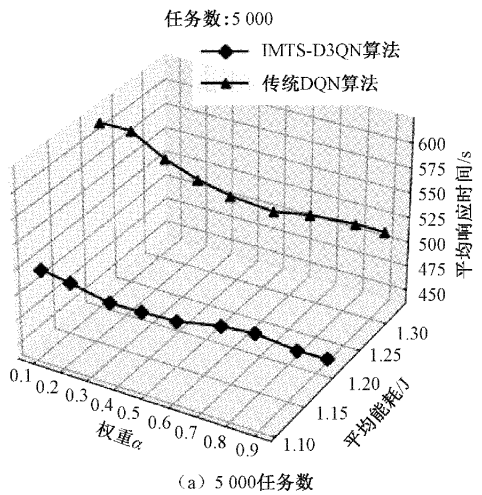
表 4　边缘服务器配置

边缘服务器类别	虚拟机类别组成
1	1,2,3,3
2	2,3
3	1,4,5
4	1,2,3,4
5	1,5

分别测试不同任务数时的平均响应时间与平均能耗的变化情况。在对响应时间和能耗同时进行优化时，如果对其中一个目标赋予相对过大的权重，则另一个目标会因此产生劣解，表明响应时间和能耗是一对相冲突解。图 2 表示不同权重组合下的不同任务数帕累托最优曲线，每组权重组合分别找出最优解，生成帕累托最优曲线。可以看出传统 DQN 算法对能耗与响应时间进行同时优化时优化效果较差，如图 2(b)和(c)随着任务数的不断增加，传统 DQN 算法波动性较图 2(a)增大，且相邻权重下的优化效果较接近。IMTS-D3QN 算法在优化的稳定性上较传统 DQN 算法有一定改善，随着任务数的不断增加，曲线偏向于平滑。可以看到 IMTS-D3QN 算法在任何权重组合下，能耗与响应时间的优化都比传统 DQN 算法要好。

图 3 选取了在权重 $\alpha=0.1$、0.9 时，IMTS-D3QN 算法与其他 4 种算法的比较情况。

随机调度算法与轮询调度算法在任务调度过程中消耗了大量的响应时间与能耗，且随着任务数的不断增加，响应



(a) 5 000任务数

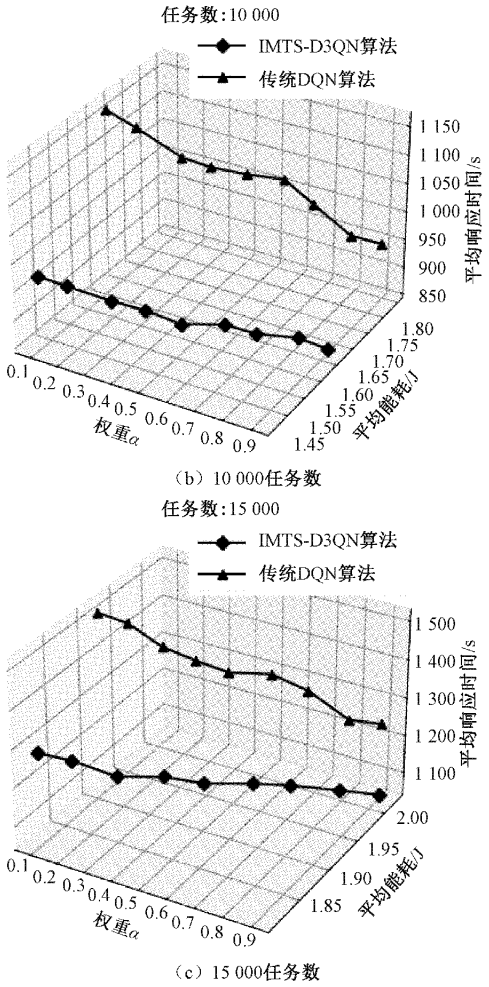
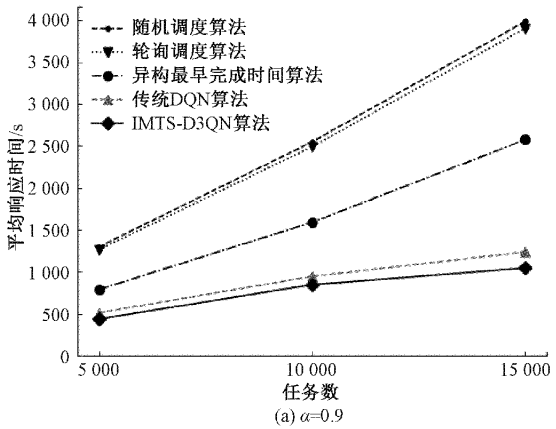


图 2　不同任务数下不同权重的最优解

时间与能耗增长很快。异构最早完成时间算法对比前面两种算法,在响应时间的优化方面表现的较不错,但是能耗的优化却不能达到预期的效果,可以看出此类算法更适用于单个目标的优化。对此使用了传统 DQN 算法对比前面 3 种算法在目标的优化上得出了不错的效果,如图 3(a)响应时间 $\alpha=0.9$ 对比图 3(c)响应时间 $\alpha=0.1$,清楚地看出权重大的优化效果越明显。图 3(b)和(d)能耗亦是如此,因



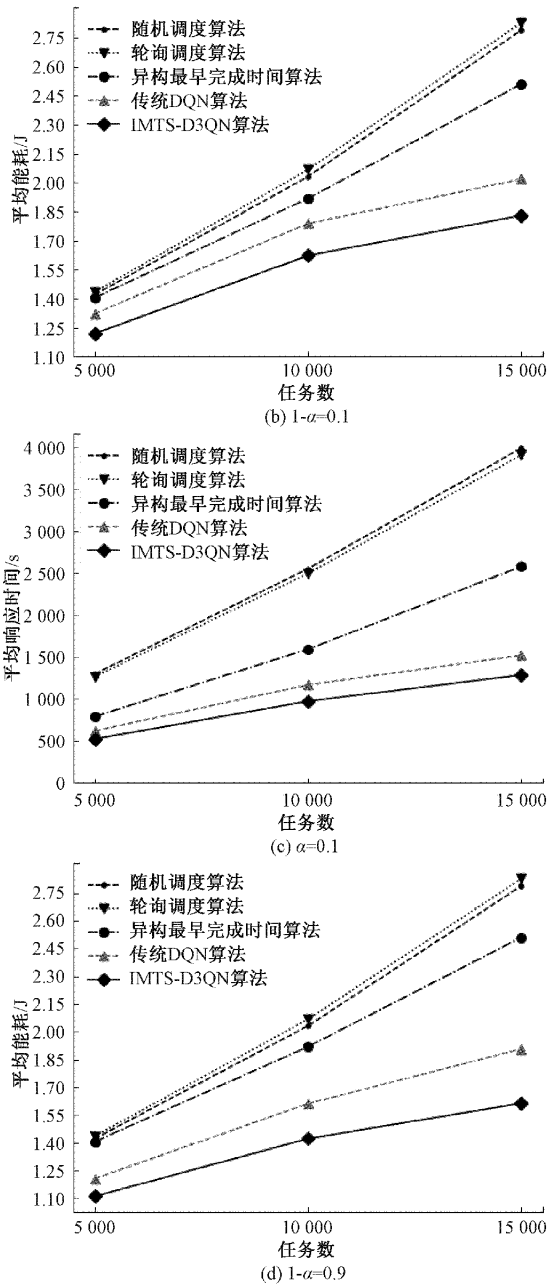


图 3　$\alpha=0.1$、0.9 时响应时间与能耗的结果

此深度强化学习算法可以通过调整不同目标奖励函数的权重,有效地权衡响应时间和能耗。提出的 IMTS-D3QN 算法在不同任务数的情况下都比其余算法具有更好的优化效果。

表 5 与 6 列出不同算法在不同任务下的平均响应时间和平均能耗的具体优化数值。

当权重 $\alpha=0.9$,任务数分别为 5 000、10 000、15 000 时,IMTS-D3QN 算法比传统 DQN 算法在响应时间上改善了 14.47%、10.45%、15.49%,在能耗上节约了 7.96%、11.79%、15.41%。

当权重 $\alpha=0.1$,任务数分别为 5 000、10 000、15 000

表 5　响应时间

算法	平均响应时间/s		
	5 000	10 000	15 000
随机调度算法	1 298.3	2 551.1	3 989.3
轮询调度算法	1 264.7	2 493.9	3 906.5
异构最早完成时间算法	791.1	1 592.1	2 578.6
传统 DQN 算法(α=0.9)	513.3	947.9	1 239.5
IMTS-D3QN 算法(α=0.9)	439.0	848.8	1 047.4
传统 DQN 算法(α=0.1)	620.8	1 168.3	1 520.4
IMTS-D3QN 算法(α=0.1)	523.0	973.8	1 285.6

表 6　能耗

算法	平均能耗/J		
	5 000	10 000	15 000
随机调度算法	1.421	2.032	2.790
轮询调度算法	1.435	2.068	2.829
异构最早完成时间算法	1.405	1.919	2.511
传统 DQN 算法(α=0.9)	1.205	1.611	1.907
IMTS-D3QN 算法(α=0.9)	1.109	1.421	1.613
传统 DQN 算法(α=0.1)	1.323	1.789	2.019
IMTS-D3QN 算法(α=0.1)	1.219	1.623	1.831

时，IMTS-D3QN 算法比传统 DQN 算法在响应时间上改善了 15.75%、16.64%、15.44%，在能耗上节约了 7.86%、9.27%、9.31%。

4　结　　论

本文针对深度强化学习在边缘环境下的多目标任务调度时存在优化效果差的情况，为了更高效地最小化能耗与响应时间，提出了一种 IMTS-D3QN 算法。首先将深度双 Q 网络对目标中的最大操作分解为动作选择和动作评估，以消除过高估计；采用立即奖励经验样本分类方法，对经验样本按照重要性程度分类存储，训练时选取更多重要性程度高的经验样本，提高了实际样本的利用率，加快了神经网络的训练速度。然后，通过引入竞争网络结构对神经网络进行优化。最后，采用软更新方法提高算法的稳定性，并采用动态 ε 贪婪指数递减法寻找最优策略。通过线性加权将能耗与响应时间表示为奖励函数，通过调节不同的权重平衡能耗与响应时间。通过对比随机调度算法、轮询调度算法、异构最早完成时间算法、传统 DQN 算法，发现 IMTS-D3QN 算法具有更好的性能。此外，还进行了大量模拟，以评估能耗与响应时间的消耗。结果表明，该算法优于其他基准方法。未来的工作将继续边缘计算的研究，考虑增加其他指标的优化。

参考文献

[1]　马学森，谈杰，陈树友，等. 云计算多目标任务调度的优化粒子群算法研究[J]. 电子测量与仪器学报，2020，34(8)：133-143.

[2]　ALFAKIH T，HASSAN M M，AI-RAZGAN M. Multi-objective accelerated particle swarm optimization with dynamic programing technique for resource allocation in mobile edge computing[J]. IEEE Access，2021，9(167)：503-520.

[3]　ABBASI M，MOHAMMADI P E，KHOSRAVI M R. Workload allocation in iot-fog-cloud architecture using a multi-objective genetic algorithm[J]. Journal of Grid Computing，2020，18(1)：43-56.

[4]　JEONG Y，MARIA E，PARK S. Towards energy-efficient service scheduling in federated edge clouds[J]. Cluster Computing，2021，338(9)：1-13.

[5]　ALFAKIH T，HASSAN M M，GUMAEI A，et al. Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA[J]. IEEE Access，2020，8(1)：54074-54084.

[6]　PENG Z，LIN J，CUI D，et al. A multi-objective trade-off framework for cloud resource scheduling based on the deep Q-network algorithm[J]. Cluster Computing，2020，23(4)：2753-2767.

[7]　RAN L，SHI X，SHANG M. SLAs-aware online task scheduling based on deep reinforcement learning method in cloud environment[C]. 2019 IEEE 21st International Conference on High Performance Computing and Communications，2019：1518-1525.

[8]　SHENG S，CHEN P，CHEN Z，et al. Deep reinforcement learning-based task scheduling in IoT edge computing[J]. Sensors，2021，21(5)：1666-1684.

[9]　CHENG M，LI J，BOGDAN P，et al. H_2O-cloud：A resource and quality of service-aware task scheduling framework for warehouse-scale data centers[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems，2019，39(10)：2925-2937.

[10]　GAO Y，WANG Y，GUPTA S K，et al. An energy and deadline aware resource provisioning，scheduling and optimization framework for cloud systems[C]. 2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS)，IEEE，2013：1-10.

[11]　MNIH V，KAVUKCUOGLU K，SILVER D，et al. Human-level control through deep reinforcement learning[J]. Nature，2015，518(7540)：529-533.

[12]　VAN HASSELT H，GUEZ A，SILVER D. Deep reinforcement learning with double Q-learning[C]. Proceedings of the AAAI Conference on Artificial

Intelligence,2016: 2094-2100.

[13] WANG Z, SCHAUL T, HESSEL M, et al. Dueling network architectures for deep reinforcement learning[C]. International Conference on Machine Learning, PMLR, 2016:1995-2003.

[14] ZHU Z, HU C, ZHU C, et al. An improved dueling deep double-Q network based on prioritized experience replay for path planning of unmanned surface vehicles[J]. Journal of Marine Science and Engineering, 2021, 9(11):1267-1281.

[15] 时圣苗,刘全.采用分类经验回放的深度确定性策略梯度方法[J]. 自动化学报,2022,48(7):1816-1823.

[16] 向卉,刘建明.结合 DDPG 与优先数据剪枝的样本处理方法[J]. 计算机仿真,2021,38(6):428-433.

作者简介

盛煜,硕士研究生,主要研究方向为强化学习。

E-mail: 20080902023@smail.cczu.edu.cn

朱正伟,教授,主要研究方向为智能检测技术及应用、嵌入式系统及应用。

E-mail: zhuzw@cczu.edu.cn

朱晨阳(通信作者),讲师,主要研究方向为机器学习、强化学习。

E-mail: zcy@cczu.edu.cn

诸燕平,副教授,主要研究方向为无线传感器网络、数值优化、移动网络计算。

E-mail: zhuyanping@cczu.edu.cn