doi:10.12068/j.issn.1005-3026.2024.12.003

# 基于边缘服务器任务迁移的资源分配算法

吴菁晶，张子轩
(东北大学 计算机科学与工程学院，辽宁 沈阳 110169)

**摘　　要**：工业物联网设备会将无法进行本地计算的任务发送至边缘服务器进行处理,但不同设备密度下的覆盖会导致不同边缘服务器的计算任务负载不均衡,进而产生计算时延过大的问题.为了解决这个问题,提出了一种基于改进的深度确定性策略梯度(modified deep deterministic policy gradient,MDDPG)的任务迁移算法,该算法具有基于深度确定性策略梯度的优先经验重放和随机权重平均机制,以寻求最佳的迁移策略,减少任务的计算时延.实验结果表明,MDDPG算法相较于传统的算法有更好的性能.

**关　键　词**：工业物联网;策略梯度;任务迁移;优先经验重放;随机权重平均
**中图分类号**：TN 929.5　　**文献标志码**：A　　**文章编号**：1005-3026(2024)12-1688-08

# Resource Allocation Algorithm Based on Edge Server Task Migration

*WU Jing-jing*, *ZHANG Zi-xuan*
(School of Computer Science & Engineering, Northeastern University, Shenyang 110169, China. Corresponding author: WU Jing-jing, E-mail: wujingjing@cse.neu.edu.cn)

**Abstract**: IIoT (industrial Internet of Things) devices send tasks that cannot be computed locally to edge servers for processing. However, different device densities result in imbalanced computational workloads among various edge servers, leading to significant computation latency. To solve this problem, a task migration algorithm based on modified deep deterministic policy gradient (MDDPG) is proposed. The algorithm has a mechanism of priority empirical replay and random weight averaging based on depth deterministic strategy gradient to find the best migration strategy and reduce the computation delay of the task. Experimental results show that MDDPG algorithm has a better performance than the traditional algorithms.

**Key words**: industrial Internet of Things; policy gradient; task migration; priority experience replay; random weight averaging

边缘计算是工业物联网(industrial Internet of Things,IIoT)中解决任务量增长、减少计算时延和提高服务质量的重要技术之一.工业物联网设备会将无法本地计算的任务发送至边缘服务器进行计算,但不同设备密度下的覆盖会导致不同边缘服务器的计算任务负载不均衡,进而产生计算时延过大的问题[1-2].为了解决这个问题,学者们广泛关注边缘服务器间的协作[3-4],通过有效通信,高负载边缘服务器可以将自身需要处理的计算任务迁移到其他低负载边缘服务器,从而减轻高负载边缘服务器的负担并优化整体时延,而在此过程中需要充分考虑网络时延的影响[5],因此,合理的任务迁移算法是必要的[6].

文献[7]提出了一种博弈论方案来优化计算卸载策略,该方案考虑了计算资源和无线资源,以最小化系统成本.文献[8]制定了一个Stackelberg模型来模拟边缘服务器和用户之间的交互,其中边缘服务器确定提供服务的价格以最大化其收入,用户根据价格做出卸载决策以最大限度地降低自己的成本.文献[9]研究了工业物联网下设

**收稿日期**：2023-07-10
**基金项目**：国家重点研发计划项目(2021YFB3401004).
**作者简介**：吴菁晶(1981-),女,辽宁沈阳人,东北大学副教授,博士生导师.

备的资源分配问题,通过找到纳什均衡来得到最优方案.文献[10]研究在边缘计算系统中有效分配计算任务以降低能耗,对此,作者利用一对多匹配理论进行建模和分析,提出了一种基于启发式交换匹配的算法.文献[11]研究了采用正交多址的边缘计算系统的计算资源和无线资源联合分配问题,文献[12]提出了一种次优匹配算法来解决该问题.物联网设备首先将其计算任务和相应的奖励发布到边缘计算系统.然后,提供计算服务的设备分析了它们通过计算任务可以获得的回报,并向系统提交了投标.边缘服务器之间的集中式协作方法需要收集全局信息,所以它可以获得更好的最优解,并且比分布式方法产生更多的开销.但是,分布式方法比集中式方法更简单、灵活、易于实现和适应动态环境.

从降低时延的角度出发,本文对边缘服务器间任务迁移问题进行全面分析,并提出了基于边缘服务器任务迁移的资源分配算法.首先建立一个边缘服务器间任务迁移的系统模型,同时构造了计算模型和传输模型,明确整体的优化目标.为了使所有任务的平均计算延迟最小化,将问题转化为马尔可夫决策过程(Markov decision processes,MDP).提出了一种基于改进的深度确定性策略梯度(modified deep deterministic policy gradient,MDDPG)的任务迁移算法来解决MDP问题.该算法具有基于深度确定性策略梯度的优先经验重放和随机权重平均机制,以寻求最佳的迁移策略,减少任务的计算时延.具体来说,优先经验重放方案的提出是为了提高经验重放缓冲区的可用性,从而加快训练速度.此外,为了减少训练过程中的噪声,从而稳定奖励,引入了随机权重平均机制来平均权重.

## 1　问题描述

图1描绘了支持计算资源分配的云边场景,强调了不同边缘服务器之间的交互.该场景由$m$个边缘服务器的集合组成,用$M=\{1,\cdots,|m|\}$表示这个集合,每个边缘服务器(mobile edge computing server,MECS)能够为其覆盖范围内的IIoT设备提供计算服务.IIoT设备将计算量较大的任务迁移至MECS进行计算,可以提高系统性能.MECS之间也可以进行交互,进行数据的发送.采用MECS的CPU频率$f_m$评估MECS $m$的计算能力,并且将全部MECS的CPU频率利用一个集合来表示$F=\{f_1,\cdots,f_M\}$.

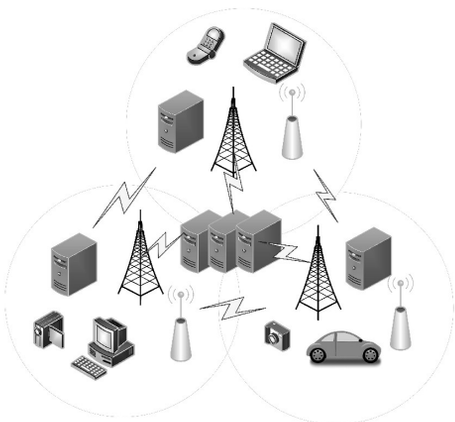

图1　任务迁移网络架构图

Fig. 1　Architecture diagram of task migration network

在实际工业生产中,IIoT设备会根据设备类型生成多种类型的计算任务.为了更好地构建模型,假定IIoT设备产生$k$种类型的计算任务,用集合$K=\{1,2,\cdots,|k|\}$来表示.对于第$k$种任务,其包含两种属性,分别是任务计算量大小$c^k$,代表计算该种任务所需要的CPU周期数;以及任务数据量大小$u^k$,代表该种任务的传输数据量大小.一个服务器能处理的最大任务计算量为$W_m$.

### 1.1　任务迁移

首先,定义2个二值变量$x_m^k,y_j^k$.其中:$x_m^k$表示任务$k$是否在本地服务器处理,$x_m^k=1$表示任务$k$在本地进行处理,否则$x_m^k=0$.$y_j^k=1$表示任务$k$需要迁移到其他服务器上进行处理,否则$y_j^k=0$.综上所述,迁移策略需要满足以下必要条件:

1)唯一性:$x_m^k+y_j^k=1(m\neq j)$.一个任务要么在本地进行处理,要么因本地服务器资源不足而迁移到其他的服务器处理.

2)安全性:MECS $j$能够计算的任务计算量之和不能超过其最大的计算能力,使得接受迁移任务的服务器不会导致系统进入不安全状态.

3)一致性:MECS $m$上的$k$种任务的本地服务器任务计算量与迁移任务的计算量之和必须和该服务器$k$种任务的初始负载量相等.

### 1.2　本地计算模型

利用排队论方法把MECS提供服务的过程建模成一个$M/M/1$的排队系统,所以排队时延是不可以忽视的,对于第$m$个MECS,其排队时延表示为

$$D_m^{\text{queue}} = \frac{\rho_m}{1-\rho_m} \cdot \frac{1}{\lambda_m^k}. \tag{1}$$

其中,在MECS计算任务中会产生一定的额外开销.同时,MECS的性能会受到这个开销的影响.参考文献[13],通过定义一个MECS性能损耗因子$\rho_m$来计算这部分性能的损耗.$\lambda_m^k$为任务到达本地服务器的到达率.

MECS $m$处理一个任务的计算时延为

$$D_m^{\text{proc}} = \frac{c^k}{f_m}. \tag{2}$$

从IIoT发送到MECS总的任务计算时延为

$$D_m^{\text{comp}} = \sum\nolimits_{k \in K} x_m^k (D_m^{\text{queue}} + D_m^{\text{proc}}). \tag{3}$$

### 1.3 任务迁移模型

传输时延代表从一个服务器发送数据到另一个服务器完全接收到这个数据的整体时间.因此,定义在所考虑的模型中,从MECS $m$发送一个数据到MECS $j$的传输延迟为

$$D_{mj}^{\text{trans}} = \frac{u^k}{B_{mj}\text{lb}(1+\text{SNR}_0)}. \tag{4}$$

其中:$B_{mj}$表示从MECS $m$到MECS $j$方向的网络带宽;$\text{SNR}_0$是用户信号强度与接收到的干扰和噪声信号强度的信噪比.

在进行任务迁移时,发送方服务器会产生排队开销,将其构建为一个$M/M/1$的排队系统,通过定义损耗因子$\rho_{\text{exo}}$来反映迁移任务时的排队开销.对于第$k$个需要迁移出去的任务,其排队时延表示为

$$D_{mj}^{\text{queue}} = \frac{\rho_{\text{exo}}}{1-\rho_{\text{exo}}} \cdot \frac{1}{\lambda_j^k}. \tag{5}$$

其中,$\lambda_j^k$为任务$k$到达发送服务器要发往其他服务器进行处理的到达率.

因此,从MECS $m$把任务迁移到其他全部MECS的时延为

$$D_{mj}^{\text{emig}} = \sum\nolimits_{k \in K, j \neq m} y_j^k (D_{mj}^{\text{queue}} + D_{mj}^{\text{trans}}). \tag{6}$$

对于第$m$个MECS上到来的处理任务$k$,其总时延由本地计算时延$D_m^{\text{comp}}$和任务迁移时延$D_m^{\text{emig}}$两部分构成,即

$$D_m^{\text{total}} = D_m^{\text{comp}} + D_{mj}^{\text{emig}}. \tag{7}$$

其中,加号左边部分表示MECS $m$在本地计算所有任务所需的时间,加号右边部分表示当MECS $m$迁移任务至全部其他MECS上计算时所需的总延迟.

### 1.4 问题优化

本文主要针对的是服务器间的任务迁移问题,因此对于所有的MECS,它们的目标是找到一个合适的任务迁移决策来最小化总计算任务的时延.对于所有的MECS,提出下述的优化问题进行求解:

$$\text{P1}: \text{minimize} D_m^{\text{total}}, \forall m \in M. \tag{8}$$

$$\left.\begin{aligned} \text{s.t.}\ \ & x_m^k + y_j^k = 1, \forall k \in K, \forall m \in M, m \neq j. \\ & \sum_{k \in K, m \in M} x_m^k c^k \leqslant W_m, \\ & \sum_{k \in K, m \in M} y_j^k c^k \leqslant W_j, \\ & \sum_{k \in K, m \in M} x_m^k c^k + \sum_{k \in K, j \in M} y_j^k c^k = W_m. \end{aligned}\right\} \tag{9}$$

将上述涉及的3个因素作为优化问题的约束.第1个约束满足了唯一性,第2个和第3个约束代表了MECS $m$的总的任务计算量不能超过其自身的计算能力上限,这么做可以实现系统的安全性,第4个约束代表了在进行任务迁移前后总系统状态需要保持一致.

## 2 MDDPG算法

首先将提出的优化问题转化为MDP问题,然后设计了一个基于深度确定性策略梯度(deep deterministic policy gradient, DDPG)的任务迁移算法来求解.一个MDP由一个5元组$(S, A, P, R, \gamma)$组成,其中$S$表示状态空间,$A$表示动作空间,$P$表示状态转移概率,$R$表示奖励函数,$\gamma \in [0, 1]$表示折扣因子.在每个时隙$t$的开始,代理检测当前系统状态$s(t) \in S$,并选择要执行的动作$a(t) \in A$.在动作$a(t)$完成之后,系统状态被更新为$s(t+1)$,并且代理根据环境的反馈获取奖励.

定义决策$\mu$为状态到行为的映射,即$a(t) = \mu(s(t))$.对于给定的决策值,定义一个动作状态值对函数$Q^{\mu}(s(t), a(t))$,其代表在$s(t)$中采用$a(t)$的预期奖励.根据贝尔曼方程,该动作状态值对函数为

$$Q^{\mu}(s(t), a(t)) = E[r(t) + \gamma Q(s(t+1), \mu(s(t+1)))]. \tag{10}$$

DDPG中的评论家网络是近似$Q$值的深度神经网络(deep neural network, DNN),即

$$Q(s(t), a(t)|\theta^Q) \approx Q(s(t), a(t)). \tag{11}$$

其中,$\theta^Q$表示评论家网络的训练参数.类似于评论家网络,演员网络是具有近似最优决策$\theta^{\mu}$的参数$\mu^*$的DNN,即

$$\mu(s(t)|\theta^{\mu}) \approx \mu^*(s(t)). \tag{12}$$

其中,$\mu^*(s(t))$表示最优行为决策.

为了提高评论家网络对$Q$值的逼近精度,通

过Adam随机梯度下降优化算法来最小化损失函数更新参数$\theta^Q$,即

$$L(\theta^Q)=E[\left(Q(s(t),a(t)|\theta^Q)-\delta(t)\right)^2], \quad (13)$$

$$\delta(t)=r(t)+\gamma Q'(s(t+1),\mu'(s(t+1)|\theta^{\mu'})|\theta^{Q'}). \quad (14)$$

训练使用的样本数据来自有限大小的重放经验缓冲$R$.在每个时隙$t$中,样本被存储在$R$中.在每个时隙$t$中,评论家网络和演员网络通过从$R$中随机选择小批量样本来更新网络参数.因此,式(13)可以改写为

$$L(\theta^Q)=\frac{1}{M}\sum_{i=1}^{M}[\left(Q(s(i),a(i)|\theta^Q)-\delta(i)\right)^2]. \quad (15)$$

演员网络使用性能目标来衡量策略$\mu$的性能.为了探索潜在的更好决策,在决策机制中引入了随机噪声.将引入随机噪声的独特行为策略定义为

$$\beta=\mu(s(t)|\theta^{\mu})+n(t). \quad (16)$$

其中$n(t)$是随机噪声.为了提高训练速度和探索效率,引入了Ornstein Uhlenbeck(OU)过程作为与时间相关的随机噪声.因此,$\beta$的性能目标函数$J_\beta(\mu)$表示为

$$J_\beta(\mu)=E[Q^{\mu}(s(t),\mu(s(t)|\theta^{\mu}))]. \quad (17)$$

最优行为决策是通过最大化性能函数得到的,即$\mu^*(s(t))=\text{argmax}J_\beta(\mu)$.为了最大化$J_\beta(\mu)$,还需要通过调整$J_\beta(\mu)$的梯度来训练参数$\theta^{\mu}$.基于小批量样本数据的梯度为

$$\nabla_{\theta^{\mu}}J_\beta(\mu)=$$
$$\frac{1}{M}\sum_{i=1}^{M}[\nabla_{a(i)}(Q^{\mu}(s(i),a(i)|\theta^Q)\nabla_{\theta^{\mu}}\mu(s(i)|\theta^{\mu}))]. \quad (18)$$

优先经验回放(prioritized experience replay,PER)机制是用于频繁地重放性能较差的经验或者极其成功的行为的经验.因此,经验的定义是一个至关重要的问题.在强制学习算法中,研究者经常使用时间差分(temporal difference,TD)误差来更新值函数的估计值,因此TD误差的值可以作为其相关的评价值.需要注意的是,TD是强制学习算法之一,它充分利用了MDP结构来实现一个更高的效率.此外,代理可以通过TD误差从过去的经验中进行学习.TD误差越大,相关性越正.因此,较高的TD误差的经验与其成功的行为有关,并有可能获得更高的价值.因为代理的行为会受到较大的负值约束,所以TD误差值也是有限的.频繁重放上述经验,既能达到在对应状态下较差的行为的真实效果,又能避免这种行为的发生.因此,糟糕的学习经验的价值可以产生一个很好的效果.为了评价经验的价值,可以选择绝对TD误差$|\Psi|$作为指标.$y$的TD误差为

$$\Psi_y=r(t)+\gamma Q'(s(t+1)),\mu'(s(t+1)|\theta^{\mu'}|\theta^{Q'}). \quad (19)$$

通过这个值来对经验进行改进.本文将经验样本$y$的概率定义为

$$P(y)=\frac{H_y^{\zeta}}{\sum_y H_y^{\zeta}}. \quad (20)$$

其中:$H_y$是$\Psi_y$的排名;$\zeta$控制优先顺序的范围.

由于较低的TD误差也具有重放的概率,在选择经验的过程中,为了确保样本的多样性,使用采样概率作为随机因素的一种方式.值得注意的是,上述提到的采样方法可以避免神经网络过度拟合,但重复使用高TD误差经验可能导致状态访问频率的不稳定性.这种变化会导致神经网络的训练出现振荡或发散现象.为了处理这个问题,在网络权重变化过程中使用随机权重平均(stochastic weight averaging,SWA),即

$$\theta_{\text{SWA}}^Q=\frac{\theta_{\text{SWA}}^Q n_{\text{SWA}}+\theta^{Q'}}{n_{\text{SWA}}+1}. \quad (21)$$

其中:$\theta_{\text{SWA}}^Q$使用$\theta^{Q'}$初始化权重;$n_{\text{SWA}}$是权重的数量.

改进的DDPG算法是一种离线算法,在算法训练过程中作出的所有决策都是由代理自主执行的.因此,该算法需要通过不断迭代来实现最优决策.DDPG算法结构如图2所示.

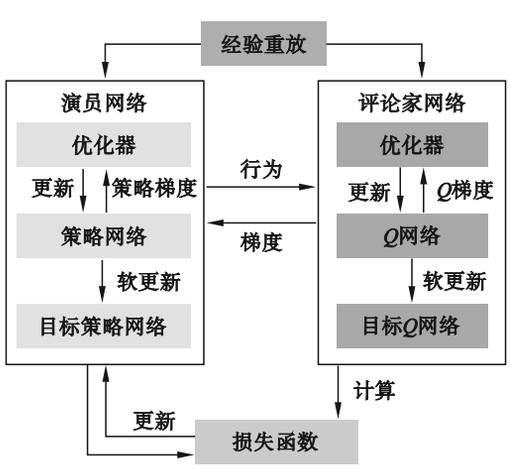


图2　DDPG算法体系结构

Fig. 2　Architecture of DDPG algorithm

对于每一次迭代,在每个时隙$t$的开始,代理根据不同的行为策略$\beta$为每个边缘服务器进行任务分配.然后,收集当前时隙$t$中所有任务的计算时延.在每个时隙$t$结束时,代理基于所有任务的平均计算时延来计算$a(t)$的$r(t)$.同时,代理收集

$s(t+1)$. 然后，将 $s(t)$, $a(t)$, $r(t)$ 和 $s(t+1)$ 存储在 $R$ 中. 代理通过从重放缓存 $R$ 中随机选择一小批样本 $\{s(i),a(i),r(i),s(i+1)\}$ 来训练演员和评论家网络. 具体地，主网络和目标网络的元素分别包括一名演员和一名评论家. 主网络的演员网络执行决策. 评论家网络使用梯度方法来帮助演员网络学习更好的决策. 作为主网络的目标网络可以产生数值来训练神经网络. 这里，按优先级排列的经验重放缓冲器可以存储当前网络状态、奖励和下一状态的经验. 可以使用演员网络或评论家网络的过去经验来更新参数.

DDPG 使用 DNN 作为近似来逼近策略函数. 具体地，演员网络的输入是 $s(i)$，输出是 $\mu(s(i)|\theta^{\mu})$. 对于评论家网络，输入是 $s(i)$, $a(i)$ 和 $\mu(s(i)|\theta^{\mu})$，输出是 $Q(s(i),a(i)|\theta^{Q})$ 和 $Q^{\mu}(s(i),\mu(s(i))|\theta^{\mu})$，其中 $Q(s(i),a(i)|\theta^{Q})$ 用于计算损失函数，$Q^{\mu}(s(i),\mu(s(i))|\theta^{\mu})$ 用于更新式(17)的参数. 与演员网络类似，演员目标网络的输入为 $s(i+1)$，输出为 $\mu'(s(i+1)|\theta^{\mu'})$. 对于评论家目标网络，输入为 $s(i+1)$ 和 $\mu'(s(i+1)|\theta^{\mu'})$，输出为 $Q'(s(i+1),\mu'(s(i+1)|\theta^{\mu'}))$. 通过上述迭代训练，DNN 对决策函数和值函数的逼近精度逐渐提高. 当演员网络和批评家网络的训练完成时，目标网络参数更新为

$$\left.\begin{aligned}\theta^{Q'}&=\phi\theta^{Q'}+(1-\phi)\theta^{Q'},\\ \theta^{\mu'}&=\phi\theta^{\mu}+(1-\phi)\theta^{\mu'}.\end{aligned}\right\} \tag{22}$$

其中，$\varphi<1$ 用于限制目标值的变化率，可以提高神经网络训练的稳定性.

算法 1 是 DDPG 算法的伪代码.

算法 1 DDPG 算法

初始化网络参数 $\theta^{Q}$ 和 $\theta^{\mu}$
初始化目标网络参数 $\theta^{Q'}$ 和 $\theta^{\mu'}$
初始化重放缓存 $R$
**for** episode=1 to 1000 **do**
  初始化随机噪声 $n(t)$
  初始化工业互联网环境获得初始状态 $s(0)$
  for $t$=1 到 $T$ do
    执行行为 $a(t)$
    获得相应的奖励 $r(t)$
    获得下一个状态 $s(t+1)$
    将元组 $\{s(t),a(t),r(t),s(t+1)\}$ 存储重放缓存 $R$ 中
    从重放缓存 $R$ 中随机进行小批次抽样元组 $\{s(i),a(i),r(i),s(i+1)\}$
    计算 $\delta(i)$
    通过最小化损失函数训练评论家网络
    通过梯度训练演员网络
    更新相应的目标网络参数
  **end**
**end**

## 3 仿真和分析

本文采用一个包含 10 个边缘服务器的网络，通过基站与 IIoT 设备连接，IIoT 设备数量为 40~50. IIoT 设备生成计算任务的过程服从泊松分布. 每个基站上的任务到达率为每秒 1~5 个. 标准计算量和标准数据量的值本文分别设置为 1 M 和 1 MB. 实际任务的数据大小则为单位任务的整数倍. 为了保证网络传输和计算速度的良好性能，实验选择了 10 千兆位以太网，并设置网络带宽为 20 Gb/s，边缘服务器的计算能力为 6 GHz.

本文进行了一系列实验来确定算法中重要超参数的最佳值. 首先分析了学习率对于所提出的算法收敛的影响，图 3 描述了所提算法在不同的学习率下对收敛性能的影响，通过图 3 可以观察到，当学习率为 0.001 和 0.000 1 时，所提算法收敛较快且收敛效果较好. 但是，当学习率为 0.1 和 0.01 时，所提出的算法收敛后达到的性能较差. 这是由于较大的学习率将使演员网络和评论家网络都采用很快的训练速度去更新，伴随着较快的收敛速度. 当收敛速度过大时，不能保证此时的收敛值是最优的；同时，当学习速率很小时，也就是学习率为 0.000 01 时，算法在收敛过程产生较大的波动，这是由于模型在每次更新参数时只会微调参数的值，因此需要更多的迭代次数才能达到最优解，这将导致收敛速度变得非常缓慢. 因此，在选择学习率时，需要平衡收敛速度和误差，综合考虑之后将网络的最佳学习率设置为 0.001.

图 4 描述了不同的折扣因子对所提出算法的收敛性能的影响，折扣因子是控制长期奖励与短期奖励之间权重的重要参数. 从图中可以看出，当折扣因子为 0.001 时，训练好的资源分配决策具有最好的性能. 当折扣因子较大时，网络会将整个时间段的采集数据视为长期数据，导致不同时间段的泛化能力较差. 需要注意的是，折扣因

子并非通用的超参数,其最优取值可能受到任务、环境和算法等多方面因素的影响.在不同的场景下,需要对折扣因子进行调整和优化,以最大程度地提升算法性能.一个较为合适的折扣因子将提高训练策略的最终性能,所以在实验中折扣因子设定为0.001.

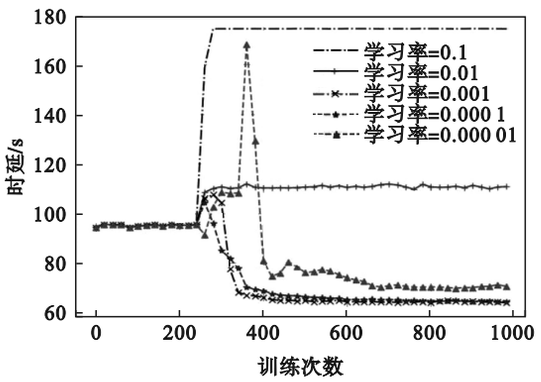


图3　学习率对算法收敛性能的影响

Fig. 3　Impact of learning rate on convergence performance

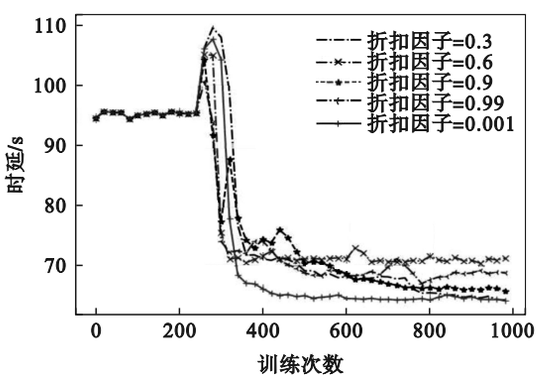


图4　折扣因子对算法收敛性能的影响

Fig. 4　Impact of discount factor on convergence performance

图5描述了不同探索率对所提算法的收敛性能的影响.当算法探索率为0.1时,收敛后的最优延迟在75 s左右波动.当探索率变大时,生成的随机噪声分布的空间也会变大,这可能会提供一个更大的空间范围让网络进行更好的探索.对于探索率为0.001的情况,较小的探索率会导致网络陷入局部最优解,网络在训练320次达到收敛状态.所以,较大的探索参数和较小的探索参数都无法收敛到最优值,因此,选取的探索率为0.01.

图6描述了所提算法与其他算法之间的性能比较.针对强化学习网络,进行了1 000次的训练.从图中看到,即使训练次数增加,A3C算法的性能很难收敛到一个平稳值.而DQN和MDDPG算法都可以达到收敛.这是由于在A3C算法中,演员网络和评论家网络的更新是同时进行的.而演员网络进行行为选择时是根据评论家网络的价值函数,但是评论家网络一般较难实现收敛.因此,会存在某些情况导致A3C算法较难收敛.而DQN和MDDPG都使用一个双重的网络结构,其中包含了评估网络和目标网络.这种结构的优势在于能够切断训练时所有数据样本之间的相关性,使每个数据样本独立,最终提升网络的训练效果,找到更优的行为策略.

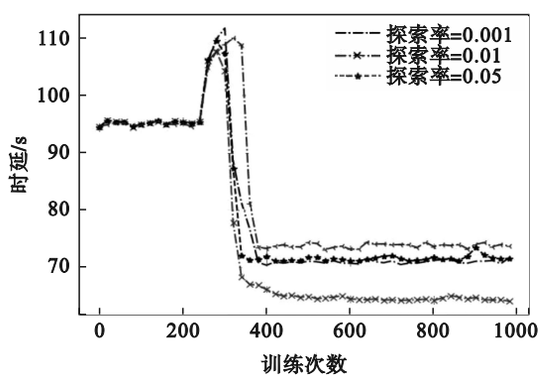


图5　探索率对算法收敛性能的影响

Fig. 5　Impact of exploration rate on convergence performance

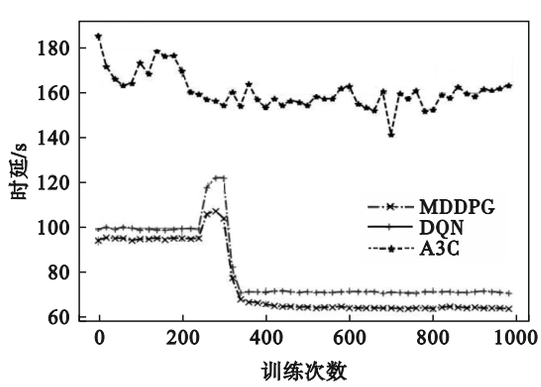


图6　不同算法收敛性能的比较

Fig. 6　Convergence performance comparison of different algorithms

图7描述了本文提出的MDDPG算法与其他算法不同任务数据大小对时延的影响.从图中可以观察到,在相同的任务数据大小的情况下,MDDPG算法的延迟是所有算法中最低的.出现这种情况的主要原因是,在DQN中可用行为空间中存在离散行为缺乏对一部分空间探索的情况,导致DQN无法找到最优资源分配策略.但是,MDDPG算法是对一个连续的行为空间进行完整的探索,最终收敛到一个最优的资源分配策略.因此,随着任务数据量的增加,MDDPG算法

的计算时延的增长速度比DQN慢,这表明了所提算法的优势.

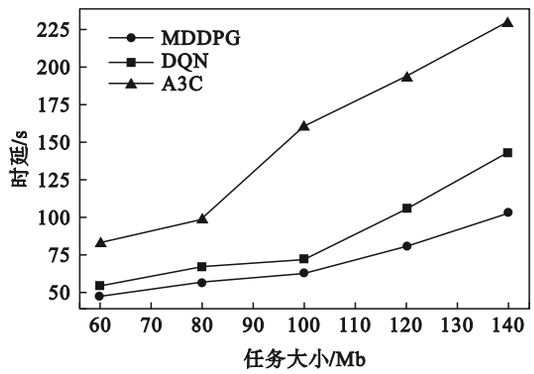


图7 任务数据量对时延的影响

Fig. 7 Impact of task data size on delay

图8描述了不同的服务器计算能力下,本文提出的算法与其他算法性能比较.从图中可以看出,在不同的边缘服务器计算能力下,本文算法获得了最低的计算时延.这主要是因为MDDPG能够输出多个连续的行为,而DQN只能在有限的离散行为集中进行输出.服务器计算能力越强,计算任务越快,需要排队等待计算的任务数量越少,服务器的负载就越少.而当服务器计算能力较差时,就会导致需要排队计算的任务数量增加,从而使计算时延增加.而提出的算法在服务器计算能力较差时,计算延迟相较于计算能力较强时的情况增加不多,从而展现出所提算法性能较优.

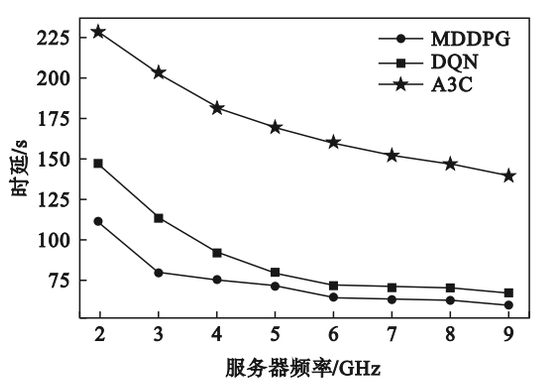


图8 服务器计算能力对时延的影响

Fig. 8 Impact of computing capacity in the server on delay

图9描述了服务器数量对不同算法时延的影响.假设在不同的边缘服务器数量下,在一定时间范围内需要计算相同数量的任务.如图所示,所提算法的计算时延随边缘服务器数量的变化而小幅变化.随着边缘服务器数量的增加,DQN方案的计算时延会产生波动,这是因为在使用不同数量的边缘服务器的情况下,使用DQN方案所得到的行为输出范围存在显著差异.网络在进行训练时,由于数据样本的原因可能会偏向于输出较大的值,尤其是针对多维行为输出的MDDPG算法.为保证MDDPG算法的收敛和稳定性,需要将网络输入数据限制在相同的范围内.此外,所提出的MDDPG算法具有最低的时延,这是因为此算法能够在连续行为下找到最优值,并实现最优行为策略.

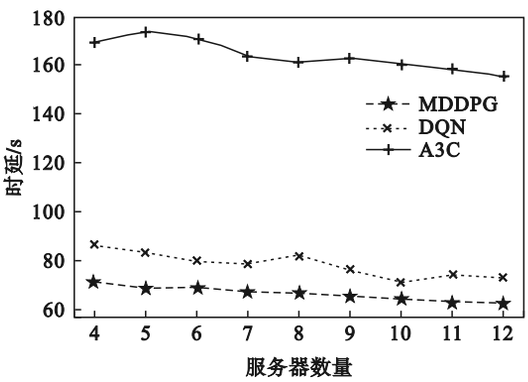


图9 服务器数量对时延的影响

Fig. 9 Impact of servers quantity on delay

## 4 结 语

本文提出了一种基于MDDPG的服务器间任务迁移算法.目的是改善IIoT边缘计算场景下的服务器任务迁移的时延最小化问题,来获得最优的分配方案.通过使用优先经验重放提高经验重放缓冲器的利用率以及使用随机权重平均对网络进行更好的训练.实验结果表明,MDDPG比其他算法具有更好的性能,期望MDDPG算法可以在IIoT中实施以优化资源分配.

**参考文献:**

[1] Zhang N, Fang X J, Wang Y, et al. Physical-layer authentication for Internet of Things via WFRFT-based Gaussian tag embedding [J]. *IEEE Internet of Things Journal*, 2020, 7(9): 9001-9010.

[2] Sisinni E, Saifullah A, Han S, et al. Industrial Internet of Things: challenges, opportunities, and directions [J]. *IEEE Transactions on Industrial Informatics*, 2018, 14(11): 4724-4734.

[3] Aazam M, Zeadally S, Harras K A. Deploying fog computing in industrial Internet of Things and Industry 4.0 [J]. *IEEE Transactions on Industrial Informatics*, 2018, 14(10): 4674-4682.

[4] Zhang N, Wu R Y, Yuan S L, et al. RAV: relay aided

vectorized secure transmission in physical layer security for Internet of Things under active attacks[J].*IEEE Internet of Things Journal*,2019,6(5):8496-8506.

[5] Wu Y, Ni K J, Zhang C, et al.Noma-assisted multi-access mobile edge computing:a joint optimization of computation offloading and time allocation[J]. *IEEE Transactions on Vehicular Technology*,2018,67(12):12244-12258.

[6] Kuang Z F,Li L F,Gao J,et al.Partial offloading scheduling and power allocation for mobile edge computing systems[J]. *IEEE Internet of Things Journal*,2019,6(4):6774-6785.

[7] He Y J,Yang M N,Guizani H M.Resource allocation based on digital twin-enabled federated learning framework in heterogeneous cellular network[J]. *IEEE Transactions on Vehicular Technology*,2023,72(1):1149-1158.

[8] Liu M Y, Liu Y. Price-based distributed offloading for mobile-edge computing with computation capacity constraints[J]. *IEEE Wireless Communications Letters*, 2018,7(3):420-423.

[9] Lin H,Hu J,Wang X D,et al.Towards secure data fusion in industrial IoT using transfer learning[J].*IEEE Transactions on Industrial Informatics*,2021,17(10):7114-7122.

[10] Zhou Z Y,Liao H J,Gu B,et al.Resource sharing and task offloading in IoT fog computing: a contract-learning approach[J]. *IEEE Transactions on Emerging Topics in Computational Intelligence*,2020,4(3):227-240.

[11] Chai R,Song X,Chen Q B.Joint task offloading,CNN layer scheduling, and resource allocation in cooperative computing system[J].*IEEE Systems Journal*,2020,14(4): 5350-5361.

[12] Zhang D Y, Tan L, Ren J, et al. Near-optimal and truthful online auction for computation offloading in green edge-computing systems[J].*IEEE Transactions on Mobile Computing*,2019,19(4):880-893.

[13] Ang C W,Tham C K.Analysis and optimization of service availability in a HA cluster with load-dependent machine availability[J].*IEEE Transactions on Parallel and Distributed Systems*,2007,18(9):1307-1319.