中国科技期刊卓越行动计划项目入选期刊

控制与决策

CONTROL AND DECISION

基于李雅普诺夫优化和深度强化学习的多任务端边迁移

许驰, 唐紫萱, 金曦, 夏长清

引用本文:
许驰,唐紫萱,金曦,夏长清. 基于李雅普诺夫优化和深度强化学习的多任务端边迁移[J]. 控制与决策, 2024, 39(7): 2457−2464.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2023.1243

基于李雅普诺夫优化和深度强化学习的多任务端边迁移

许 驰[1,2,3†], 唐紫萱[1,2,3,4], 金 曦[1,2,3], 夏长清[1,2,3]

(1. 中国科学院沈阳自动化研究所 机器人学国家重点实验室，沈阳 110016；
2. 中国科学院 网络化控制系统重点实验室，沈阳 110016；
3. 中国科学院 机器人与智能制造创新研究院，沈阳 110169；4. 中国科学院大学，北京 100049)

摘 要: 针对多终端、多边缘服务器场景下异构工业任务的端边协同处理问题,提出一种基于李雅普诺夫优化和深度强化学习的多任务端边迁移算法. 首先,以联合优化任务迁移决策、迁移比例和传输功率为目标,充分考虑计算频率、传输功率、长期能耗和任务截止期等约束,构建系统长期平均开销最小化问题;由于问题中长期目标及约束中变量在不同时隙相互耦合,难以求解,基于李雅普诺夫优化理论,将长期平均开销最小化问题解耦为独立时隙的策略优化问题;通过马尔可夫决策过程建模,并采用双层竞争深度神经网络架构,提出基于深度强化学习的多任务迁移算法. 实验结果表明,所提算法能够稳定收敛,并在长期能耗约束和任务截止期要求下有效降低系统长期平均开销.

关键词: 异构工业任务；任务迁移；李雅普诺夫优化；马尔可夫决策过程；深度强化学习

中图分类号: TP39 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2023.1243

引用格式: 许驰,唐紫萱,金曦,等. 基于李雅普诺夫优化和深度强化学习的多任务端边迁移[J]. 控制与决策, 2024, 39(7): 2457-2464.

Multi-task end-edge offloading based on Lyapunov optimization and deep reinforcement learning

XU Chi[1,2,3†], *TANG Zi-xuan*[1,2,3,4], *JIN Xi*[1,2,3], *XIA Chang-qing*[1,2,3]

(1. State Key Laboratory of Robotics，Shenyang Institute of Automation，Chinese Academy of Sciences，Shenyang 110016，China；2. Key Laboratory of Networked Control Systems，Chinese Academy of Sciences，Shenyang 110016，China；3. Institutes for Robotics & Intelligent Manufacturing，Chinese Academy of Sciences，Shenyang 110169，China；4. University of Chinese Academy of Sciences，Beijing 100049，China)

Abstract: To enable collaborative processing of heterogeneous industrial tasks in the scenario with multiple devices and multiple edge servers, this paper proposes a multi-task end-edge offloading algorithm based on Lyapunov optimization and deep reinforcement learning. First, to jointly optimize task offloading decision, offloading ratio and transmit power, a long-term average system overhead minimization problem is formulated with full consideration of computing frequency, transmission power, long-term energy consumption and task deadline. As variables are coupled among different time slots in the long-term objective and constraints, the problem is difficult to solve. Thus, the long-term average system overhead minimization problem is decoupled into some independent time-slot optimization problems based on the Lyapunov optimization theory. By Markov decision process modelling and employing a double dueling deep neural network architecture, a deep reinforcement learning-based multi-task offloading algorithm is proposed. Experiments show that the proposed algorithm can converge stably, and can effectively reduce the long-term average system overhead under long-term energy consumption constraints and task deadline requirements.

Keywords: heterogeneous industrial tasks；task offloading；Lyapunov optimization；Markov decision process；deep reinforcement learnings

收稿日期: 2023-08-31；**录用日期:** 2023-12-06.

基金项目: 国家自然科学基金项目 (92267108, 62173322, 62133014, 61972389)；辽宁省科技计划项目 (2023JH3/10200004, 2023JH3/10200006, 2022JH25/10100005)；中国科学院青年创新促进会项目 (2019202, 2020207, Y2021062).

责任编委: 邓庆绪.

†通讯作者. E-mail: xuchi@sia.cn.

0 引 言

工业无线网络技术的高速发展,使得海量的工业终端可以实现泛在感知与互联[1]. 当前,工业现场设备面向单一工业任务,普遍存在计算、存储、能量等资源受限问题,为此,边缘计算应运而生. 通过部署边缘服务器,进行任务的动态迁移,可以补充算力资源,提高任务处理实时性[2]. 然而,当大规模异构工业任务并发迁移到边缘服务器时,将会引起网络拥塞和边缘服务器过载等一系列问题,大幅增加任务处理时延及能耗[3]. 因此,必须充分考虑网络随机波动、端边计算资源分布、工业终端电池寿命和传输功率、异构任务截止期等约束,确定有效的任务迁移策略.

现有的研究工作以优化时延、能耗等为目标,采用深度强化学习(deep reinforcement learning, DRL)方法,开展了大量研究工作[4-8]. 在此基础上,通过将时延与能耗的加权和定义为系统开销,并加以优化,可以根据工业应用的要求动态优化时延和能耗指标. 文献[9]以最小化系统开销为目标,提出了一种区块链确信、数字孪生辅助的任务迁移方法;类似地,文献[10]基于DRL提出了一种关于计算决策、计算能力和传输功率的资源分配策略,以最小系统开销;文献[11]设计了一种基于混合决策的Actor-Critic算法来解决具有连续-离散混合动作空间的动态迁移问题,使系统开销最小;文献[12]将DDPG架构中的神经网络替换为图卷积网络,并进行特征提取,提出了关于迁移比例、计算资源和传输功率的任务迁移与资源分配方法;文献[13]提出了一种时间注意力确定性策略梯度算法来解决计算卸载与资源分配的联合优化问题,以减小系统开销;文献[14]针对具有顺序任务图的单基站-多用户系统,提出了一种基于DRL的卸载决策与资源分配联合优化方法,以最小化系统开销;文献[15]以最小化系统开销和适应动态环境为目标,提出了一种分布-集合式DRL算法;文献[16]提出了一种时延-能量均衡的在线迁移算法,通过自适应调整权重大小来最小化系统开销. 结果表明,与DQN、DDQN等基准DRL算法相比,LyDRL能够在保证系统稳定性的同时,有效降低系统长期开销.

现有研究工作较少考虑系统的长期稳定性. 为此,本文针对长期能耗约束下的多任务端边迁移问题,提出一种基于李雅普诺夫优化和深度强化学习的任务迁移算法(Lyapunov optimization and deep reinforcement learning-based task offloading, LyDRL),以最小化系统长期平均开销.

本文的主要贡献如下:

1) 针对多终端、多边缘服务器场景下的异构工业任务端边迁移问题,充分考虑计算频率、传输功率、长期能耗和任务截止期等约束,建立系统长期平均开销最小化问题,以优化任务迁移决策、迁移比例和传输功率.

2) 针对系统长期平均开销最小化问题中不同时隙间变量的强耦合问题,基于李雅普诺夫优化理论,将原问题重构为独立时隙策略优化问题,实现问题的解耦,保障系统稳定性.

3) 针对独立时隙策略优化问题的非凸性,采用马尔可夫决策过程进行建模,提出基于双层竞争深度神经网络的DRL算法,以获取最佳任务迁移策略.

1 系统模型

如图1所示,考虑一个多终端、多边缘服务器的端边协同任务处理场景,包含 M 个工业终端和 N 个边缘服务器. 其中,工业终端均资源受限,执行异构工业任务,包括控制命令等时延敏感型任务,过程感知和视频监控等计算密集型任务;边缘服务器可以为附近的工业终端提供计算资源,以处理复杂任务,降低任务处理时延.

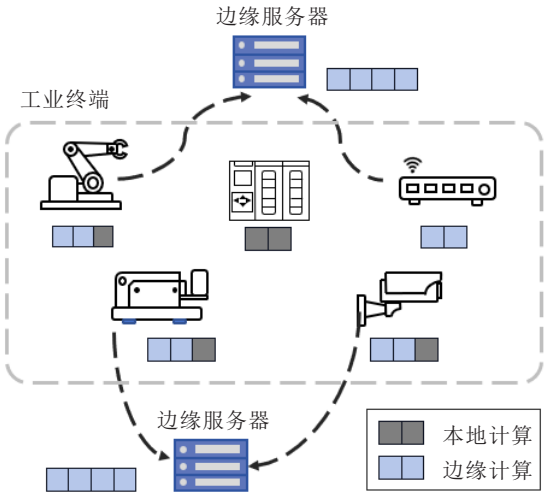

图 1 系统模型

在时隙 t 内,工业终端 m 产生的工业任务数据量为 D_m^t,可以在本地进行处理或迁移到单一边缘服务器进行处理. 以 $o_{m,n}^t \in \{0,1\}$ 表示时隙 t 内工业终端 m 的迁移决策,$o_{m,0}^t = 0$ 表示工业任务 D_m^t 不迁移,$o_{m,n}^t = 1$ 表示工业任务 D_m^t 将迁移到边缘服务器 n 进行处理. 因此,迁移决策应满足

$$\sum_{n=0}^{N} o_{m,n}^t = 1, \tag{1}$$

其中 $n=0$ 表示本地计算.

在迁移决策的基础上,用 $\lambda_{m,n}^t$ 表示工业任务 D_m^t 在时隙 t 内迁移到边缘服务器 n 进行处理的比例,则

任务迁移比例应满足

$$0 \leqslant \lambda_{m,n}^t \leqslant 1. \tag{2}$$

1.1 本地计算模型

当工业终端m在本地处理工业任务时,所有本地计算资源都将用于处理该任务. 此时,本地处理时延计算为

$$T_{m,\,\text{local}}^t = \frac{(1-\lambda_{m,n}^t)D_m^t c_m^t}{f_m}. \tag{3}$$

其中: c_m^t表示计算1比特任务所需计算周期, f_m表示工业终端m的本地计算资源.

相应地,工业任务的本地处理能耗[9-10]可计算为

$$E_{m,\,\text{local}}^t = k(1-\lambda_{m,n}^t)D_m^t c_m^t f_m^2, \tag{4}$$

其中k表示电容开关系数.

1.2 边缘计算模型

当工业终端m将工业任务迁移到边缘服务器n进行处理时,任务迁移的速率可根据香农定理计算为

$$R_{m,n}^t = w_{m,n}^t \log_2\left(1+\frac{o_{m,n}^t P_m^t g_{m,n}}{w_{m,n}^t N_0}\right), \tag{5}$$

其中$w_{m,n}^t$表示工业终端m进行任务迁移时的带宽,由系统总带宽W和迁移决策结果$o_{m,n}^t$决定,计算为$w_{m,n}^t = W\Big/\sum_{m=1}^{M} o_{m,n}^t$. 注意,当工业终端$m$进行本地计算,即$n=0$时,$w_{m,0}^t = 0$. $g_{m,n}$表示工业终端m与边缘服务器n之间的信道功率增益. N_0表示噪声功率密度, P_m表示工业终端m传输功率. 显然, P_m受到自身硬件最大传输功率$P_{\max}$的限制,即

$$0 \leqslant P_m \leqslant P_{\max}. \tag{6}$$

当工业终端m产生的工业任务被迁移到边缘服务器n进行处理时,传输时延计算为

$$T_{m,\,\text{edge}}^t = \frac{\lambda_{m,n} D_m^t}{R_{m,n}^t}. \tag{7}$$

相应地,传输能耗计算为

$$E_{m,\,\text{edge}}^t = P_m^t T_{m,\,\text{edge}}^t. \tag{8}$$

2 问题建模

假设边缘服务器具有海量的计算资源,边缘计算时延可忽略. 同时,由于边缘计算结果较小,边缘计算结果的反馈时延亦可忽略. 在时隙t内,处理工业任务D_m^t的时延可计算为本地计算时延与边缘传输时延的最大值,即

$$T_m^t = \max(T_{m,\,\text{local}}^t, T_{m,\,\text{edge}}^t). \tag{9}$$

与此同时,时隙t内,处理工业任务D_m^t的能耗为本地计算能耗与边缘传输能耗之和,即

$$E_m^t = E_{m,\,\text{local}}^t + E_{m,\,\text{edge}}^t. \tag{10}$$

综合考虑系统的时延和能耗指标,定义时隙t内系统开销为时延与能耗的加权和,即

$$\text{cost}_m^t = \omega T_m^t + (1-\omega)E_m^t. \tag{11}$$

其中: ω表示时延的权重, $1-\omega$表示能耗的权重.

在此基础上,建立系统长期平均开销最小化问题P_1:

$$\min_{\boldsymbol{\mathcal{O}},\boldsymbol{\lambda},\boldsymbol{\mathcal{P}}} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M} \text{cost}_m^t. \tag{12}$$

s.t. 式(1)、(2)、(6);

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\text{E}[E_m^t] \leqslant e_t,\ t\in T; \tag{13}$$

$$T_m^t \leqslant \gamma_t,\ t\in T. \tag{14}$$

其中: $\boldsymbol{\mathcal{O}} = \{o_{m,n}^t\}_{M\times N}$, $\boldsymbol{\lambda} = \{\lambda_{m,n}^t\}_{M\times N}$和$\boldsymbol{\mathcal{P}} = \{P_m^t\}_M$分别表示迁移决策、迁移比例和传输功率的集合; E表示期望. 约束(13)为长期平均能耗约束,其中假设所有工业终端在每个时隙的能耗阈值均相同,记为e_t. 约束(14)为时延截止期约束,其中γ_t表示时延阈值,满足时延截止期约束(14)代表工业任务处理成功.

3 基于李雅普诺夫优化的问题解耦

在异构工业任务数据随机到达和无线信道状态时变的情况下,未来系统环境及状态是未知的. 与此同时,不同时隙的系统决策是互相影响的,很难满足长期能耗约束(13),且问题P_1的目标也是长期目标,故问题P_1不能直接求解. 因此,本文基于李雅普诺夫优化理论[17]将原问题解耦为独立时隙策略优化问题,以保证系统的稳定性.

首先,引入M个能量队列$\{Q_m(t)\}_{m=1}^M$,重构长期能耗约束(13). 其中,能量队列定义为每个时隙内能耗的队列积压,初始队列积压为$Q_m(0)=0$. 能量队列的更新过程为

$$Q_m(t+1) = \max\{Q_m(t)-e_t, 0\} + E_m^t, \tag{15}$$

其中能耗阈值e_t即为队列服务速率. 当能量队列稳定时,长期平均能耗不会超过e_t,可满足约束条件(13).

然后,引入李雅普诺夫函数

$$L(\boldsymbol{Q}(t)) = \frac{1}{2}\sum_{m=1}^{M} Q_m(t)^2, \tag{16}$$

其中$L(\boldsymbol{Q}(t))$的大小反映队列的状态. 通过式(16)可以看出,如果队列积压过多, $L(\boldsymbol{Q}(t))$会变得越来越大. 因此,只有当所有工业终端的队列积压都很小时, $L(\boldsymbol{Q}(t))$才会得到一个较小的值.

为保证能量队列的稳定性,引入李雅普诺夫漂移函数$\Delta L(\boldsymbol{Q}(t))=L(\boldsymbol{Q}(t+1))-L(\boldsymbol{Q}(t))$来表示从时隙$t$到时隙$t+1$的李雅普诺夫函数的增长,其上界由如下引理推导得出.

引理1 李雅普诺夫漂移函数$\Delta L(\boldsymbol{Q}(t))$的上界为

$$\Delta L(\boldsymbol{Q}(t))\leqslant C+\sum_{m=1}^{M}Q_m(t)(E_m^t-e_t),\tag{17}$$

其中

$$C\triangleq\sum_{m=1}^{M}\frac{(E_m^t)^2+t(e_t)^2}{2}.$$

证明 对于非负实数a,b,c,d及其不等式关系$a\leqslant\max(b-c,0)+d$,存在不等式关系$a^2\leqslant b^2+c^2+d^2+2b(d-c)$. 因此,根据式(15),可以得到

$$\begin{aligned}&(Q_m(t+1))^2=\\&(\max\{Q_m(t)-e_t,0\}+E_m^t)^2\leqslant\\&(Q_m(t))^2+e_t^2+(E_m^t)^2+2Q_m(t)(E_m^t-e_t),\end{aligned}$$

那么

$$\begin{aligned}&\Delta L(\boldsymbol{Q}(t))=\\&\frac{1}{2}\sum_{m=1}^{M}(Q_m(t+1))^2-\frac{1}{2}\sum_{m=1}^{M}(Q_m(t))^2\leqslant\\&\frac{1}{2}\sum_{m=1}^{M}[e_t^2+(E_m^t)^2+2Q_m(t)(E_m^t-e_t)]=\\&C+\sum_{m=1}^{M}Q_m(t)(E_m^t-e_t).\end{aligned}$$

□

在此基础上,定义条件李雅普诺夫漂移函数为$\Delta\boldsymbol{Q}(t)=\mathrm{E}\{\Delta L(\boldsymbol{Q}(t))|\boldsymbol{Q}(t)\}$,表示时隙$t$下所有能量队列积压的期望,由当前时隙$t$的任务迁移策略决定. 为了在保证能量队列稳定性的同时系统长期平均开销最小化,引入李雅普诺夫漂移加惩罚项,定义为

$$\Delta\boldsymbol{Q}(t)+\beta\sum_{m=1}^{M}\mathrm{E}\{\mathrm{cost}_m^t|\boldsymbol{Q}(t)\},\tag{18}$$

其中惩罚系数β是一个正参数.

根据引理1,式(18)的上界为

$$\begin{aligned}&\Delta\boldsymbol{Q}(t)+\beta\sum_{m=1}^{M}\mathrm{E}\{\mathrm{cost}_m^t|\boldsymbol{Q}(t)\}\leqslant\\&C+\sum_{m=1}^{M}Q_m(t)\mathrm{E}[(E_m^t-e_t)|\boldsymbol{Q}(t)]+\\&\beta\sum_{m=1}^{M}\mathrm{E}\{\mathrm{cost}_m^t|\boldsymbol{Q}(t)\}.\end{aligned}\tag{19}$$

在每个时隙t内,观察队列积压情况,并最小化漂移加惩罚项. 也就是说,在最小化队列积压的同时,最小化式(19). 去除式(19)中的常数项,将原始问题P_1转化为独立时隙策略优化问题P_2:

$$\min_{\boldsymbol{\mathcal{O}},\boldsymbol{\lambda},\boldsymbol{\mathcal{P}}}\sum_{m=1}^{M}Q_m(t)E_m^t+\beta\sum_{m=1}^{M}\mathrm{cost}_m^t;\tag{20}$$

s.t. 式(1)、(2)、(6)、(14).

其中E_m^t与β可分别视为所有工业终端能量队列与系统开销的惩罚系数,以动态平衡工业终端系统开销与能量队列. 直观地,问题P_2的目标(20)为在时隙t内,最小化所有工业终端系统开销的同时,减小能量队列积压.

4 LyDRL算法

4.1 马尔可夫决策过程建模

在问题P_2中,约束条件(1)、(2)、(6)、(14)中既有离散变量又有连续变量. 因此,问题P_2是一个混合整数非线性规划问题,属于典型的NP难问题,无法在多项式时间内求解[18]. 为此,本文采用马尔可夫决策过程对问题P_2进行建模,并基于DRL算法进行求解.

1) 状态空间: 在时隙t内,状态$\boldsymbol{s}(t)$由工业任务数据量、所需计算周期数、工业终端与边缘服务器的距离以及能量队列积压组成,定义为

$$\boldsymbol{s}(t)=\{\boldsymbol{D}(t),\boldsymbol{C}(t),\boldsymbol{l}(t),\boldsymbol{Q}(t)\}.\tag{21}$$

其中

$$\begin{aligned}&\boldsymbol{D}(t)=\{D_m^t\}_M,\ \boldsymbol{C}(t)=\{c_m^t\}_M,\\&\boldsymbol{l}(t)=\{l_{m,n}\}_{M\times N},\ \boldsymbol{Q}(t)=\{Q_m(t)\}_M,\end{aligned}$$

分别表示工业终端产生的工业任务数据量、处理工业任务所需计算周期、工业终端m到边缘服务器n的距离以及工业终端m能量队列积压的集合.

2) 动作空间: 在时隙t内,动作$\boldsymbol{a}(t)$由任务迁移决策、任务迁移比例与传输功率组成,定义为

$$\boldsymbol{a}(t)=\{\boldsymbol{O}(t),\boldsymbol{\lambda}(t),\boldsymbol{P}(t)\}.\tag{22}$$

其中

$$\begin{aligned}&\boldsymbol{O}(t)=\{o_{m,n}^t\}_{M\times N},\\&\boldsymbol{\lambda}(t)=\{\lambda_{m,n}^t\}_{M\times N},\\&\boldsymbol{P}(t)=\{P_m^t\}_M,\end{aligned}$$

分别表示工业终端的任务迁移决策、任务迁移比例与传输功率的集合.

3) 奖励空间: 奖励$r(t)$表示在当前状态$\boldsymbol{s}(t)$下采取动作$\boldsymbol{a}(t)$时产生的动作奖励. 根据目标函数(20),在时隙t内,工业终端m的奖励r_m^t定义为

$$r_m^t=-Q_m(t)E_m^t-\beta\mathrm{cost}_m^t.\tag{23}$$

显然,工业终端的系统开销及能量队列越小,得到的奖励越大.

基于此,所有工业终端获得的奖励之和计算为

$$r(t)=\sum_{m=1}^{M} r_m^t, \tag{24}$$

累积奖励可计算为给定时长T内所有工业终端得到的奖励之和,即

$$R(t)=\sum_{t=0}^{T} \gamma^t r(t), \tag{25}$$

其中γ为折扣系数,表示历史奖励对当前奖励的影响.

4.2　基于DRL的算法设计

为求解马尔可夫决策过程中的累积奖励最大化问题,充分考虑无线网络环境的动态多变以及异构任务高并发迁移造成的求解维度灾难问题[19],本文提出LyDRL算法,基本思路如图2所示. 单个工业终端作为智能体与环境进行交互,产生当前时隙的状态$\boldsymbol{s}(t)$、动作$\boldsymbol{a}(t)$、奖励$r(t)$及下一时隙状态$\boldsymbol{s}(t+1)$,并存入经验池中. 每个迭代周期,智能体从经验池中对数据进行随机采样,训练神经网络,并更新超参数. 智能体输出动作$\boldsymbol{a}(t)$到环境中.

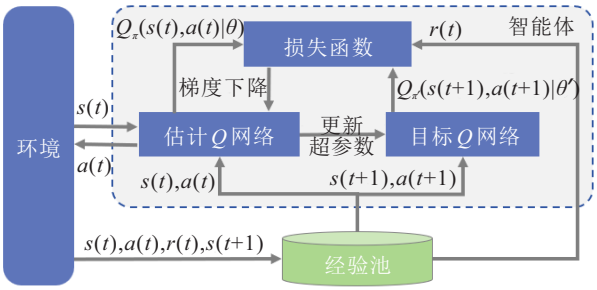

图 2　LyDRL算法结构

具体来说,本文提出采用如图3所示的双层竞争深度神经网络架构. 其中,设计两个结构相同但超参数不同的深度神经网络,即估计Q网络与目标Q网络. 估计Q网络根据当前时隙t的状态$\boldsymbol{s}(t)$近似估计Q值$Q_\pi(\boldsymbol{s}(t),\boldsymbol{a}(t)|\theta)$. 目标$Q$网络根据下一时隙$t+1$的状态$\boldsymbol{s}(t+1)$近似目标$Q$值$R(t)+\gamma\max Q_{\pi'}(\boldsymbol{s}(t+1),\boldsymbol{a}(t+1)|\theta')$. 估计$Q$网络的超参数$\theta$是实时更新的,而目标$Q$网络的超参数$\theta'$需要迭代一定的次数才会更新. 也就是说,估计$Q$网络与目标$Q$网络的超参数异步更新,进而可以避免振荡与发散.

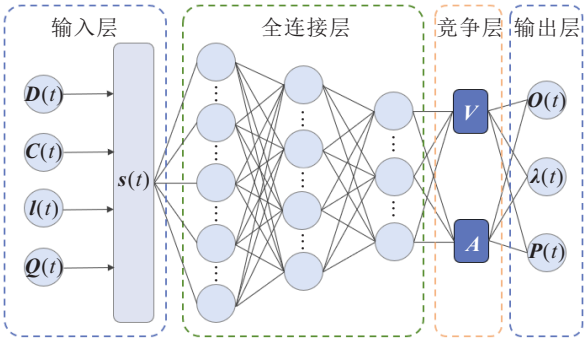

图 3　神经网络结构

与此同时,在深度神经网络中引入竞争层,将全连接层的输出分为两条路径. 其中,路径V和路径A分别输出当前时隙t下与状态$\boldsymbol{s}(t)$相关的标量$V(\boldsymbol{s}(t)|\theta)$和与动作$\boldsymbol{a}(t)$相关的向量$\boldsymbol{A}(\boldsymbol{s}(t),\boldsymbol{a}(t)|\theta)$. 标量$V(\boldsymbol{s}(t)|\theta)$用来估计状态值函数,向量$\boldsymbol{A}(\boldsymbol{s}(t),\boldsymbol{a}(t)|\theta)$用来估计动作优势函数相比于当前状态值函数的优势,表明当前状态下各个动作相对的好坏程度. 通过引入竞争架构选择当前状态下合适的动作,可以避免Q值高估问题.

在此基础上,估计Q网络与目标Q网络之间的损失函数可计算为

$$\begin{aligned}L(\theta)=\mathrm{E}[(&R(t)+\gamma\max Q_{\pi'}(\boldsymbol{s}(t+1),\\&\boldsymbol{a}(t+1)|\theta')-Q_\pi(\boldsymbol{s}(t),\boldsymbol{a}(t)))^2].\end{aligned} \tag{26}$$

为实现深度神经网络的异步更新,利用随机梯度下降法更新当前神经网络参数θ,即

$$\begin{aligned}&\nabla_\theta L(\theta)=\\&\mathrm{E}[(R(t)+\gamma\max Q_{\pi'}(\boldsymbol{s}(t+1),\boldsymbol{a}(t+1)|\theta')-\\&Q_\pi(\boldsymbol{s}(t),\boldsymbol{a}(t)))\delta\nabla_\theta Q_\pi(\boldsymbol{s}(t),\boldsymbol{a}(t)|\theta)].\end{aligned} \tag{27}$$

4.3　算法训练

在算法训练过程中,为避免陷入局部最优,采用ϵ-greedy策略选择动作,以平衡探索新动作与利用最大Q值动作之间的关系. 其中,ϵ值随着迭代次数增加而逐渐减小. 与此同时,加入经验回放机制,构建一个经验池来存储智能体与环境交互获得的经验,即当前状态、动作、奖励与下一时隙状态. 算法训练时,从经验池中随机采样一批次经验数据来更新神经网络,其中经验池大小为U.

LyDRL算法的训练过程如算法1所示.

算法1　LyDRL算法训练过程.

输入: 状态$\boldsymbol{s}(t)$;

输出: 动作$\boldsymbol{a}(t)$.

1) 初始化: 估计Q网络参数θ,目标Q网络参数θ',迭代次数K,神经网络更新步数S;

2) for $k=0,1,\ldots,K$ do

3)　从经验池中随机采样L个经验作为训练数据;

4)　将$\boldsymbol{s}(t)$输入到估计Q网络,得到$Q_\pi(\boldsymbol{s}(t),\boldsymbol{a}(t))$;

5)　根据ϵ-greedy策略选择动作$\boldsymbol{a}(t)$,执行动作$\boldsymbol{a}(t)$获得奖励$r(t)$,并到达下一状态$\boldsymbol{s}(t+1)$;

6)　将$(\boldsymbol{s}(t),\boldsymbol{a}(t),r(t),\boldsymbol{s}(t+1))$存储到经验池;

7)　将$\boldsymbol{s}(t+1)$输入到目标Q网络,得到$\boldsymbol{a}(t+1)$;

8) 根据式(27)更新估计Q网络参数θ;
9) 将状态$\boldsymbol{s}(t)$转换为$\boldsymbol{s}(t+1)$;
10) if $k \geqslant S$ then
11) 将估计Q网络参数θ更新为θ';
12) end if
13) end for

5 实验分析

为验证LyDRL算法的有效性和优势,在Intel i7-13700k CPU与NVIDIA RTX4090-24G GPU工作站上部署TensorFlow-GPU-1.14.0与Python3.7环境,开展实验验证与分析. 实验基本参数如表1所示[3,7]. 此外,选择DDQN与DQN作为基准算法进行对比实验.

表1 仿真参数

参数名称	参数符号	参数数值
工业终端数量	M	5～30
边缘服务器数量	N	3
控制数据大小	D_m^t	[30, 100) kb
感知数据大小	D_m^t	[100, 500) kb
视频数据大小	D_m^t	[500, 1 000) kb
系统总带宽	W	20 MHz
本地计算资源	f_m	2 GHz/s
电容开关系数	k	10^{-27}
最大传输功率	$P_{\max}$	300 mW
噪声功率密度	N_0	10^{-11} mW
长期能耗阈值	e_t	2 W
时延阈值	γ_t	0.5 s
惩罚系数	β	10
迭代次数	K	30 000
学习速率	α	0.001
折扣系数	γ	0.9
经验池大小	U	10 000
批量采样经验数	L	128
神经网络更新步数	S	100

图4描述了不同算法下的归一化奖励. 可以看出,随着迭代次数的增加,所有算法最终都能收敛到一个稳定的值,说明了算法的有效性. 其中,采用LyDRL算法得到的奖励相比DDQN与DQN得到的奖励分别高出了近15 %与20 %,即本文所提LyDRL算法的奖励最高,并且更加稳定.

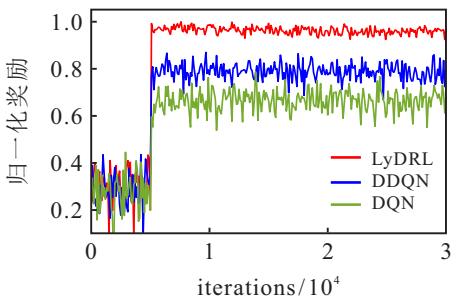

图4 归一化奖励

如图5所示,随着迭代次数的增加,所有算法得到的平均时延都会从一个较高的值收敛到一个较低的值. 其中,DDQN与DQN算法得到的平均时延均高于0.5 s,不满足任务的截止期要求. 相比之下,LyDRL算法的平均时延最低,且满足任务截止期约束.

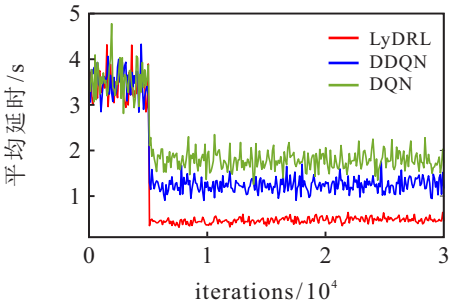

图5 不同算法的平均时延对比

类似地,如图6所示,随着迭代次数的增加,所有算法得到的平均能耗都会下降,并收敛到一个稳定的值. 相比于DDQN与DQN,LyDRL算法的平均能耗下降最多,且满足长期能耗约束.

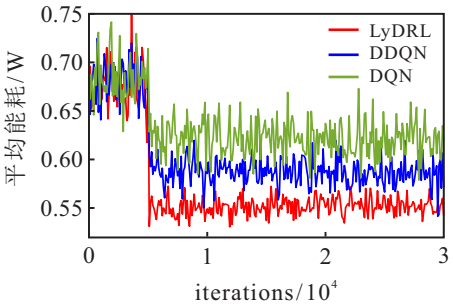

图6 不同算法的平均能耗对比

图7描述了在不同工业终端数量的情况下,不同算法获得的系统长期平均开销. 可以看出,随着工业终端数量的增加,所有算法得到的系统长期平均开销均增大. 这是由于工业终端数量增加的同时,异构工业任务高并发迁移,使得同一边缘服务器下工业终端分得带宽减小,造成时延和能耗增加,从而导致系统长期平均开销变大. 对于相同数量的工业终端,LyDRL算法得到的系统长期平均开销总是最小的,表明了LyDRL算法可有效降低系统长期平均开销.

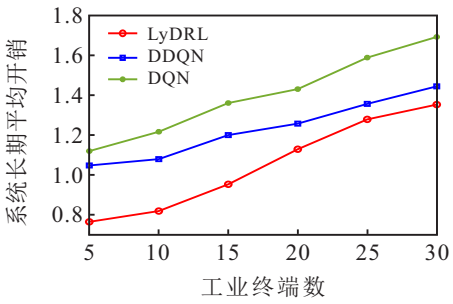

图7 不同工业终端数量时的系统长期平均开销

图 8 描述了不同本地计算资源 f_m 情况下, 系统长期平均开销随着训练次数的变化趋势. 从图 8 可以看出, 本地计算资源设置为 2 GHz/s 时, 系统长期平均开销最小, 优于本地计算资源分别设置为 1 GHz/s 和 3 GHz/s 的情况. 这是由于当本地计算资源过小时, 工业终端的本地处理时延增大, 导致系统长期平均开销增大. 相反, 当本地计算资源过大时, 尽管可以降低本地处理时延, 但是根据本地能耗计算公式 (4), 本地计算资源增加时, 本地能耗呈指数增加, 导致系统长期平均开销也变大. 因此, 只有设置合适的本地计算资源, 才能有效平衡时延与能耗的加权和, 减小系统长期平均开销.

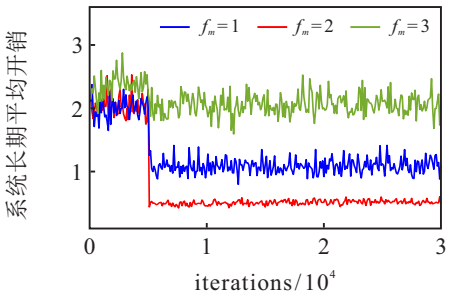

图 8　不同本地计算资源对系统长期平均开销的影响

图 9 给出了不同长期能耗阈值 e_t 约束下, 所有工业终端能量队列之和随着训练次数的变化趋势. 可以看出, 长期能耗阈值 e_t 越大, 能量队列积压越小. 其原因主要在于: 根据李雅普诺夫优化理论, 当 e_t 变大时, 离开能量队列的能耗变大, 从而减小队列积压, 使长期平均能耗满足约束, 保证系统的长期稳定性. 具体地, 当 $e_t = 2$ 时, 能量队列积压随着训练次数增加而下降, 且最终收敛在一个较小的值. 当 $e_t = 1.5$ 时, 能量队列积压也会随着训练次数增加而下降并收敛, 但最终收敛到的值比 $e_t = 2$ 时收敛值大. 当 $e_t = 1$ 时, 长期能耗约束较为严格, 能量队列积压随着训练次数增加振荡, 并一直处于一个较高的值.

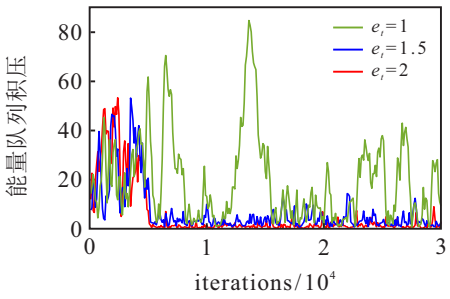

图 9　不同长期能耗阈值对能量队列积压的影响

6　结　论

为实现多终端、多边缘服务器场景下异构工业任务的端边协同处理, 本文提出了一种基于李雅普诺夫优化和深度强化学习的任务迁移算法. 首先, 充分考虑长期能耗约束, 建立一个以优化任务迁移决策、任务迁移比例与传输功率为目标的系统长期平均开销最小化问题. 然后, 基于李雅普诺夫优化理论将该问题转化为独立时隙策略优化问题. 进一步, 通过马尔可夫决策过程将问题转化为累积奖励最大化问题, 并提出基于双层竞争深度神经网络的 LyDRL 算法, 以获取最优任务迁移策略. 实验结果表明, 与 DDQN 和 DQN 等基准算法相比, LyDRL 算法能够快速收敛, 可以在长期能耗约束下最小化系统长期平均开销, 保证系统的长期稳定性.

参考文献 (References)

[1] Ahlen A, Akerberg J, Eriksson M, et al. Toward wireless control in industrial process automation: A case study at a paper mill[J]. IEEE Control Systems Magazine, 2019, 39(5): 36-57.

[2] Porambage P, Okwuibe J, Liyanage M, et al. Survey on multi-access edge computing for Internet of Things realization[J]. IEEE Communications Surveys & Tutorials, 2018, 20(4): 2961-2991.

[3] 刘晓宇, 许驰, 曾鹏, 等. 面向异构工业任务高并发计算卸载的深度强化学习算法[J]. 计算机学报, 2021, 44(12): 2367-2381.
(Liu X Y, Xu C, Zeng P, et al. Deep reinforcement learning-based high concurrent computing off loading for heterogeneous industrial tasks[J]. Chinese Journal of Computers, 2021, 44(12): 2367-2381.)

[4] Wang J, Hu J, Min G Y, et al. Dependent task offloading for edge computing based on deep reinforcement learning[J]. IEEE Transactions on Computers, 2022, 71(10): 2449-2461.

[5] 李燕君, 蒋华同, 高美惠. 基于强化学习的边缘计算网络资源在线分配方法[J]. 控制与决策, 2022, 37(11): 2880-2886.
(Li Y J, Jiang H T, Gao M H. Reinforcement learning-based online resource allocation for edge computing network[J]. Control and Decision, 2022, 37(11): 2880-2886.)

[6] Liu C B, Tang F, Hu Y K, et al. Distributed task migration optimization in MEC by extending multi-agent deep reinforcement learning approach[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(7): 1603-1614.

[7] Xu C, Tang Z X, Yu H B, et al. Digital twin-driven collaborative scheduling for heterogeneous task and edge-end resource via multi-agent deep reinforcement learning[J]. IEEE Journal on Selected Areas in Communications, 2023, 41(10): 3056-3069.

[8] Tuong V D, Noh W, Cho S. Delay minimization for NOMA-enabled mobile edge computing in industrial Internet of Things[J]. IEEE Transactions on Industrial Informatics, 2022, 18(10): 7321-7331.

[9] Liu T, Tang L, Wang W L, et al. Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network[J]. IEEE Internet of Things Journal, 2022, 9(2): 1427-1444.

[10] Liu X Y, Xu C, Yu H B, et al. Multi-agent deep reinforcement learning for end-edge orchestrated resource allocation in industrial wireless networks[J]. Frontiers of Information Technology & Electronic Engineering, 2022, 23(1): 47-60.

[11] Zhang J, Du J, Shen Y, et al. Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach[J]. IEEE Internet of Things Journal, 2020, 7(10): 9303-9317.

[12] Chen J, Wu Z L. Dynamic computation offloading with energy harvesting devices: A graph-based deep reinforcement learning approach[J]. IEEE Communications Letters, 2021, 25(9): 2968-2972.

[13] Chen J, Xing H L, Xiao Z W, et al. A DRL agent for jointly optimizing computation offloading and resource allocation in MEC[J]. IEEE Internet of Things Journal, 2021, 8(24): 17508-17524.

[14] Yan J, Bi S Z, Zhang Y J A. Offloading and resource allocation with general task graph in mobile edge computing: A deep reinforcement learning approach[J]. IEEE Transactions on Wireless Communications, 2020, 19(8): 5404-5419.

[15] Qiu X Y, Zhang W K, Chen W H, et al. Distributed and collective deep reinforcement learning for computation offloading: A practical perspective[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(5): 1085-1101.

[16] Jiao X L, Ou H J, Chen S G, et al. Deep reinforcement learning for time-energy tradeoff online offloading in MEC-enabled industrial internet of things[J]. IEEE Transactions on Network Science and Engineering, 2023, 10(6): 3465-3479.

[17] Neely M J. Stochastic network optimization with application to communication and queueing systems[M]. Switzerland: Springer, Cham, 2010.

[18] Kannan R, Monma C L. On the computational complexity of integer programming problems[C]. Berlin, Heidelberg: Springer, 1978: 161-172.

[19] Luong N C, Hoang D T, Gong S M, et al. Applications of deep reinforcement learning in communications and networking: A survey[J]. IEEE Communications Surveys & Tutorials, 2019, 21(4): 3133-3174.

作者简介

许驰 (1987−), 男, 研究员, 博士, 从事工业控制系统、工业无线网络、5G、边缘计算等研究, E-mail: xuchi@sia.cn;

唐紫萱 (1999−), 女, 硕士生, 从事工业无线网络与人工智能等研究, E-mail: tangzixuan@sia.cn;

金曦 (1983−), 女, 研究员, 博士, 从事工业网络调度、5G等研究, E-mail: jinxi@sia.cn;

夏长清 (1985−), 男, 副研究员, 博士, 从事工业网络调度、边缘计算等研究, E-mail: xiachangqing@sia.cn.