DOI:10.13992/j.cnki.tetas.2024.06.014

云边端协同下的任务调度与资源分配方法

徐帅帅，苏敏杰，任迅，吴一叶，余润泽，胡涛

（中国移动通信集团设计院有限公司浙江分公司，杭州 310012）

摘　要　为了优化多用户终端、多边缘节点以及云服务器之间的计算资源分配，找到最优的任务调度和资源分配方案。本文在算网融合的基础上提出了一种3层计算资源分配方案，考虑到通信网络带宽的有限性，设立了任务计算优先级和任务上传策略，并在多边缘节点的环境下，制定了多节点并行计算规则，以最大化计算资源的利用率，通过深度强化学习算法进行训练，以学习最优的任务调度和资源分配策略。实验结果表明，本文所提方法在缩短计算任务完成时间方面表现出色，并且在任务数据量增长的情况下，依然表现出良好的鲁棒性。

关键词　任务调度；资源分配；马尔科夫决策过程；深度强化学习

中图分类号　TN915　　**文献标识码**　A　　**文章编号**　1008-5599（2024）06-0050-07

近年来，随着云计算、大数据和人工智能等技术的快速发展，算力需求呈现爆炸式增长[1]。算网融合[2]是一种创新的计算和存储模式，依托高速、移动、安全和泛在的网络连接，整合云、边、端等多层次算力资源，提供数据感知、传输、存储和运算等一体化服务。

在算网融合场景下，云、边、端 3 级的资源分配需要网络和计算资源高度协同，计算单元和计算能力需要嵌入网络，利用泛在闲散算力来缓解算力的潮汐效应，满足高算力和低时延的实时计算场景需求。

1　相关工作

随着自动驾驶、AR/VR、云游戏等新型技术的兴起，具有强大算力和较小延迟的新型应用场景的使用价值将日益上升[3]。

参考文献[4]提出了一种基于移动边缘计算环境下的自适应资源分配算法，以泊松分配形式模拟生成任务流，减少了任务响应时间和总系统能耗。参考文献[5]提出了一种基于二进制编码遗传算法的分布式资源分配策略，适用于多基站协调下的资源分配模式。参考文献[6-8]将任务调度和资源分配问题转化为马尔科夫决策过程，使用深度强化学习训练神经网络，以获得资源分配问题的最优解。

参考文献[9]在分配计算资源时综合考虑时变的信道强度和动态资源需求。参考文献[10]通过将边缘设备之间的资源协调和分配建模为一个潜在的博弈，并提出了一种带有确定性策略梯度算法的多智能体来优化资源分配。参考文献[11]考虑到任务分配过程中数据量大小与位置、通信成本和网络状态等相关的动态信息。参考文献[12-13]以任务的最短完成时间为约束，解决复杂网络之间的任务调度问题。参考文献[14]针对时间任务流模式，制定任务数据缓存决策，满足长期能量约束的同时，有效降低移动设备的整体计算延迟。参考文献[15]将任务分配建模为马尔科夫决策过程，在网络转移状态

收稿日期：2024-03-07

未知的情况下实现资源协同计算。

与现有工作相比，本文考虑任务计算优先级和任务截止时间对任务分配和资源调度的影响因素，并结合深度强化学习方法，优化任务最短完成时间。

2　系统模型

2.1 系统架构设计

综合考虑任务调度与资源分配问题中的终端数量、边缘节点数量和云节点等因素构建的云边端协同架构如图 1 所示。

本文构建具有 M 个终端，N 个边缘计算节点和 1 个云服务器的云边端协同架构。终端产生的计算任务可以上传到边缘节点或者云服务器，且具有任务处理的截止时间。

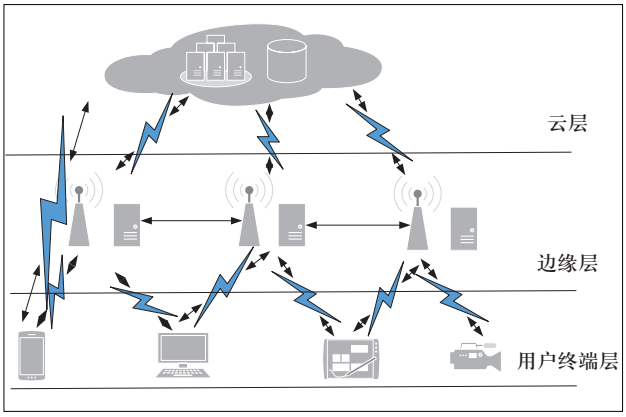

图1　云边端架构示意图

2.2 通信模型

本文终端任务上传到边缘节点或者云服务器均考虑以无线链路方式传输，根据香农公式，可以得到其传输速率如下。

$$R_{i,k}=B_{i,k}\times lb(1+\frac{p_i g_{i,k}}{\sigma^2})$$

其中，$i\in\{1,2,3,\cdots,n\}$ 是任务的个数；$B_{i,k}$ 是链路带宽；lb 是以 2 为底的对数；p_i 是设备发射功率；$g_{i,k}$ 是信道增益，与数据传输距离有关，可表示为 $g_{i,k}$ 等于 $D_{i,k}^{-\omega}$；ω 是路径损失指数；σ^2 是噪声功率。

2.3 计算模型

2.3.1 终端计算模型

任务以任务流的方式调度到终端设备、边缘设备和云服务器进行计算。计算任务在终端设备处理时，所需时间的计算如下。

$$t_i^{terminal}=\frac{d_i c}{f_i^{terminal}}$$

其中，$f_i^{terminal}$ 表示终端设备的计算能力（处理器主频）；c 表示处理 1 位数据的周期；d_i 表示任务 i 的数据量大小。

2.3.2 边缘节点计算模型

将任务上传到边缘节点或者云服务器上的时延如下。

$$t_i^s=\frac{d_i}{R_{i,k}}$$

考虑边缘节点的异构性，将边缘节点的计算能力定义为 f_i^{edge} 等于 $\{f_1^{edge},f_2^{edge},\cdots,f_n^{edge}\}$，边缘设备上任务的计算时间如下。

$$t_i^{edge}=\frac{d_i c}{f_i^{edge}}$$

其中，f_i^{edge} 表示边缘节点设备的计算能力（处理器主频）。

2.3.3 云服务器计算模型

当任务上传到云节点，云节点处理任务所需计算时间如下。

$$t_i^{cloud}=\frac{d_i c}{f_i^{cloud}}$$

其中，f_i^{cloud} 表示云服务器提供的计算能力（处理器主频）。

结合终端计算、边缘计算和云计算过程可得，云边端协同计算任务完成总时间的目标函数如下。

$$min\sum_{i=1}^{n}[t_i^{terminal}+(1-\sum_{j=1}^{m}x_i^j)(t_i^{edge}+st_i^{edge})+$$

$$\sum_{j=1}^{m}x_i^j(t_i^{cloud}+st_i^{cloud})]$$

其中，st_i^{egde} 和 st_i^{cloud} 表示任务 i 在边缘设备或云服务器执行计算的开始时间；$\sum_i^j(i=1,2,\cdots,j=1,2,\cdots,n)$ 表示任务是在边缘设备执行或是在云服务器上执行，即当 $\sum_{j=1}^{m}x_i^j$ 等于 0 时，任务执行在边缘设备。

为避免同一个任务流在边缘节点和云服务器上同时运行而造成的资源浪费问题，约束同一个任务流只能在边缘节点或者云服务器上执行，并确保任务在截止时间内完成，必须满足如下。

$$\sum_{j=1}^{m}x_i^j \le 1, \forall i \in \{1,2,...,n\}$$

$$t_i^{terminal}+(1-\sum_{j=1}^{m}x_i^j)(t_i^{edge}+st_i^{edge})+\sum_{j=1}^{m}x_i^j(t_i^{cloud}+st_i^{cloud}) \le T_i$$

其中，$\sum_{j=1}^{m}x_i^j$ 表示任务执行的计算节点；T_i 表示任务的截止时间。

为确保计算任务被上传到边缘节点和云服务器，对于任务 i 在边缘设备或云服务器执行计算的开始时间 st_i^{egde} 和 st_i^{cloud}，必须满足如下。

$$st_i^{edge} \ge t_i^s \sum_{j=1}^{m}x_i^j, i \in \{1,2,...,n\}$$

$$st_i^{cloud} \ge t_i^s \sum_{j=1}^{m}x_i^j, i \in \{1,2,...,n\}$$

2.4 任务优先级

任务调度过程中，以任务的到达时间、处理时间和截止时间决定其优先级。任务的计算量越大或者任务的时延越小，任务的优先级越高。任务的优先级状态如下。

$$p_i^s = \alpha \frac{d_i c}{t_i^{\max}}$$

其中，$t_i^{\max}$ 是任务完成的最大时间；α 是一个缩放系数。

3 任务调度与资源分配机制

本文将云边端协同任务调度与资源分配问题建模为马尔科夫决策过程，并采用深度强化学习的方法训练神经网络，具体流程如图 2 所示。

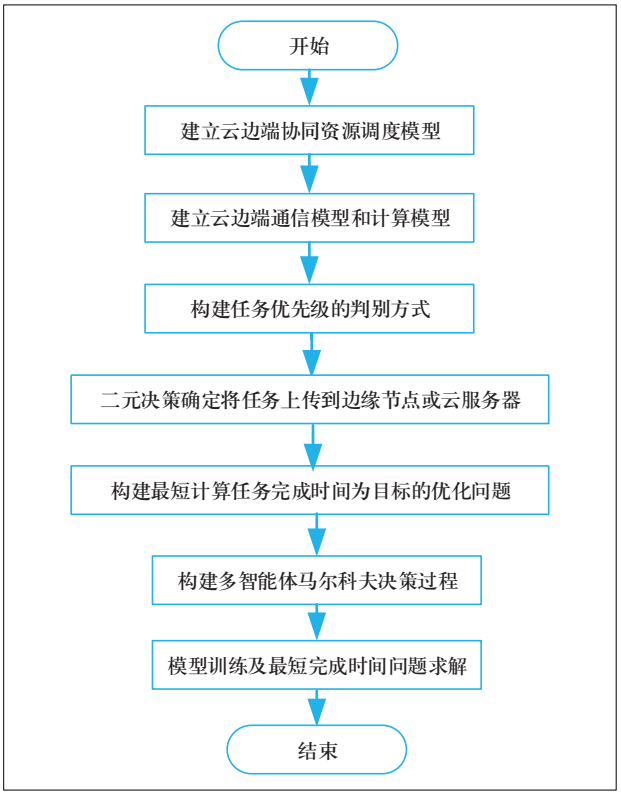

图2 云边端协同任务调度与资源分配流程图

3.1 多智能体马尔科夫决策过程

任务的优先级决策和任务上传方式可以看作两步顺序决策问题，需要两个决策代理。本文将任务优先级和任务上传方式定义为两个智能体，将云边端任务调度与资源分配问题描述为多智能体马尔科夫决策过程，其系统状态、动作空间和奖励函数描述如下。

3.1.1 系统状态

将云边端任务调度与资源分配的多智能体马尔科夫决策过程设计为任务优先级和任务上传决策两个代理，系统状态定义如下。

$$S_t = (S_t^p, S_t^u)$$

其中，S_t^p 表示任务优先级系统状态；S_t^u 表示任务上传决策系统状态。

任务优先级在时间 t 下的系统状态 S_t^u 如下。

$$S_t^p = [et_1 \cup F_1 \cup I_1, ..., et_n \cup F_n \cup I_n]$$

其中，$et_i, i \in \{1,2,\cdots,n\}$ 表示任务的预计完成时间；F_i 表示在时间步长为 t 的约束内，任务的剩余时间；I_i 记录任务是否被成功调度。

任务上传决策在时间 t 下的系统状态 S_t^u 如下。

$$S_t^u = [bt_{i0} \cup pt_{i0}, bt_{i1} \cup pt_{i1}, bt_{i2} \cup pt_{i2}]$$

其中，bt_{i0} 和 pt_{i0} 表示任务在终端设备的开始时间和处理时间；bt_{i1} 和 pt_{i1} 表示任务在边缘节点的开始时间和处理时间；bt_{i2} 和 pt_{i2} 表示任务在云服务上的开始时间和处理时间。

3.1.2 动作空间

当某个任务产生时，应结合其具体情况，如传输时延和算力需求，决定其是被上传到边缘节点或是云服务器。该过程同样由两个代理组成，定义动作 A_t 如下。

$$A_t = (a_p, a_u)$$

其中，a_p 为 $(1,2,\cdots,n)$ 表示任务优先级动作空间；a_u 为 $(0,1)$ 表示任务上传决策动作空间。

3.1.3 奖励函数

本文构建的云边端协同任务调度与资源分配问题的优化目标为完成计算任务的最短时间。根据动作空间和计算时间确定任务的完成时间如下。

$$R_t(S_t, A_t) = ct_i$$

其中，ct_i 是任务 i 的完成时间。该奖励函数可以用来评价动作的好坏，并指导神经网络更新参数。

3.2 基于深度强化学习的任务调度与资源分配方法

本文提出了一种基于深度强化学习的云边端协同任务计算方法，基于 DDQN 构建的神经网络示意如图 3 所示。首先，任务流被分成两个智能体代理，即任务优先级和任务上传决策。任务优先级和任务上传决策通过两个全连接层(FC)输出状态动作对的 Q_m 值。全连接层由一对FC组成，负责学习状态动作对 Q 值的映射关系，并使用 ReLU 激活函数做线性整流。输出层输出有效动作的 Q_m 值。

强化学习目的是将 t 时刻系统状态长期效用最优化，定义 t 时刻的累计回报函数如下。

$$U_t = \sum_{i=0}^{n} \gamma^i (R_{t+i})$$

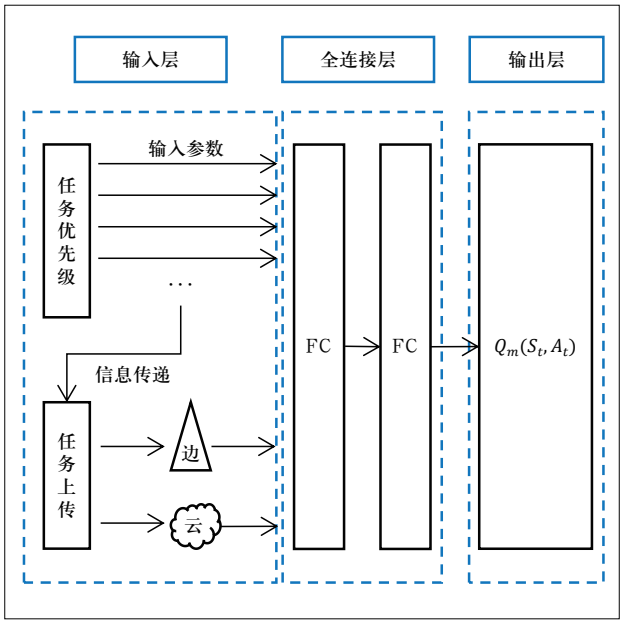

图3　神经网络示意图

其中，γ 是折扣因子，用来平衡未来的奖励和当前的奖励。

通过马尔科夫决策和求期望的方式得到只与当前时隙的状态和动作有关的价值函数。

$$Q^*(s_t, a_t) = \mathrm{E}[U_t \mid S_t = s_t, A_t = a_t]$$

其中，E 表示期望运算符。

解决强化学习问题需要找到一个最优策略 Q 值。

$$a_t = argmax Q^*(s_t, a_t)$$

$$\gamma[Q_\pi(s_{t+1}, a_{t+1})] \approx \gamma \mathrm{E}[Q_\pi(s_{t+1}, a_{t+1})]$$

由上式可得到 t 时隙和 t+1 时隙的动作价值的关系如下。

$$\begin{aligned} Q_\pi(s_t, a_t) &= R_t + \gamma \mathrm{E}[Q_\pi(s_{t+1}, a_{t+1})] \\ &= R_t + \gamma[Q_\pi(s_{t+1}, a_{t+1})] \end{aligned}$$

本文模型的训练包含主网络和目标网络两个神经网络，二者具有相同的神经网络结构。主网络选出下一个状态 Q 的最大动作 a^* 等于 $argmax Q_\pi(s_{t+1}, a_{t+1}, \omega)$，目标网络获得动作 a^* 的 Q，即 $Q_\pi^*(s_{t+1}, a^*, \omega^*)$。$\omega$ 和 ω^* 表示主网络和目标网络具有不同的参数。

$$Loss = \frac{1}{q} \sum_{i=1}^{q} [R_i + \gamma \max Q_\pi(s_{i+1}, a_{i+1}, \omega) - Q_\pi(s_i, a_i, \omega)]^2$$

其中，q 表示经验池中采样的样本数量。

4　实验结果与分析

本节通过仿真实验对所提算法性能进行分析，并从任务流的数量、边缘节点并行计算个数、任务数据量的大小等方面对本文算法进行评估。

4.1 实验参数

本文实验设置为具有 30 台终端设备、8 个边缘节点和 1 个云服务器的分布式网络环境。具体参数见表 1。

4.2 对比实验

为证明本文所提算法的效果，将与以下算法进行对比分析。

（1）ALO。一种多终端设备的任务调度算法，将终端任务上传到最近的边缘节点或者进一步上传到云服务器[16]。

（2）全部本地计算。任务全部在本地终端设备上进行计算。

（3）边缘计算。任务全部上传到边缘节点进行计算。

表1　参数与参数取值

参数	取值
任务的数量（个）	10 ~ 15
单个任务的数据大小（kB）	1000 ~ 2000
任务完成要求的截止时间（秒）	0.5 ~ 2
边缘节点并行计算的数量（个）	[2,4,5,6,8]
处理器处理 1 位数据的周期（纳秒）	500
终端设备的计算能力（GHz）	2 ~ 4
边缘节点的计算能力（GHz）	5 ~ 10
云服务器分配给终端设备的计算能力（GHz）	15 ~ 20
终端到边缘节点或云的距离（km）	10 ~ 20
边缘节点到云的距离（km）	1 ~ 10
通信带宽（GHz）	2
噪声功率（dBm）	−100
终端设备的传输功率（dBm）	30
边缘节点设备的传输功率（dBm）	100
路径损失指数	4
折扣因子	0.9

（4）云计算。任务全部上传到云服务器进行计算。

算法在不同学习率（LR）上的收敛情况如图 4 所示，学习率即每次迭代更新向损失函数最小值移动的步长。当 LR 等于 0.000 010 0 时，模型的收敛情况虽快，但起伏波动较大；当 LR 等于 0.000 001 0 时，模型的收敛效果较好；当 LR 等于 0.000 000 1 时，模型收敛情况较慢，且波动情况较多。综上所述，本文选择模型的 LR 等于 0.000 001 0。

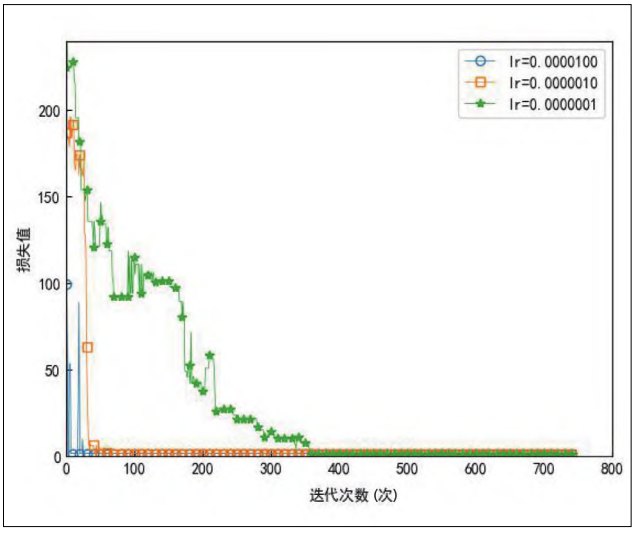

图4　不同学习率下的模型收敛示意图

不同算法产生的任务完成时间对比结果如图 5 所

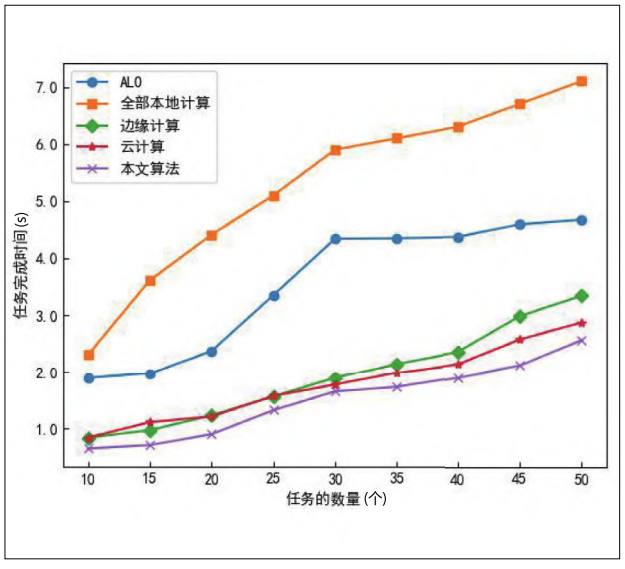

图5　不同任务数量的算法对比结果

示。ALO 算法的表现并不突出，结合 ALO 算法特性分析，该算法在处理任务时，未考虑任务的优先级，且以最近的边缘节点或者云服务器为上传节点，会造成边缘节点或者云服务器资源分配不均问题，从而影响整体的任务完成时间。本文所提将任务拆成多个任务流的方法，在优化任务总完成时间上具有突出表现。通过对比分析，本文算法考虑任务优先级，且在云边端之间协同任务计算，从而在任务完成时间上优于其它算法。

如图 6 所示，在任务总数据量不变的情况下，随着边缘节点可以进行并行计算数量的增加，使得边缘设备的利用率提高，任务的完成时间也逐渐减少。但是，由于任务上传到不同距离边缘设备的传输时间也随之增加，故本文算法在处理任务的完成时间会随着边缘节点数量的增多逐渐趋于平缓。

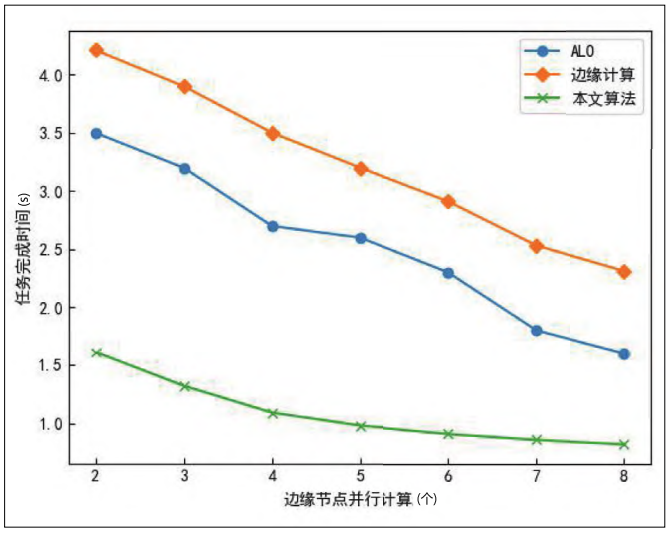

图6　边缘节点并行计算的算法对比结果

任务的不同数据大小下算法的性能对比如图 7 所示，随着任务数据量的增加，任务完成的时间也都随之增加。全部本地算法随着任务数据量的增加，任务完成时间速率增长速度比任务数据量增长速率更快，因为过多的数据量会造成固定计算资源的性能折扣。ALO 算法随着任务数据量的增加，任务计算完成时间呈现较稳定增长。边缘计算和云计算随着任务数据量的增加，呈现不同的增长

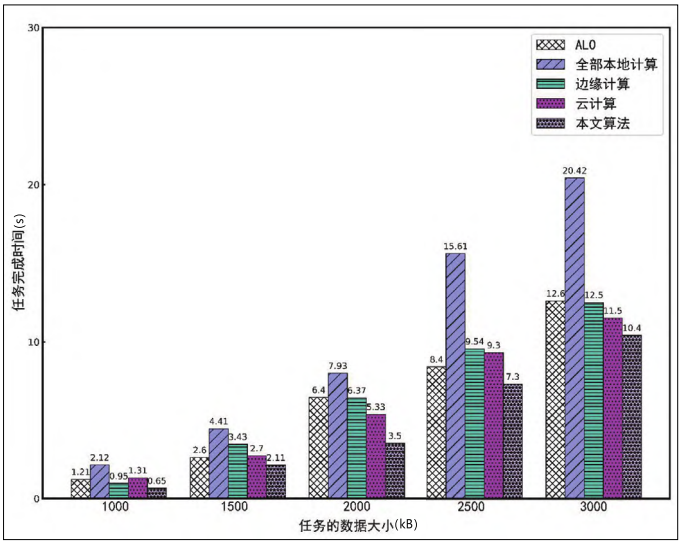

图7　不同任务数据大小的算法对比

幅度，当任务数据量较小时，边缘计算更有优势。当任务数据量较大时，云计算更有优势。对比以上算法，本文算法在数据量增加的情况下，仍然具有较好的鲁棒性。

5　结束语

本文在多用户终端、多边缘节点和云服务器的 3 层计算资源架构下，提出了一种云边端协同的任务调度和资源分配策略，将任务优先级和任务上传策略引入云边端协同模型中，联合系统状态、动作空间和奖励机制构建多智能体马尔科夫决策过程，使用深度强化学习方法训练神经网络，以任务最短完成时间为优化目标，得到任务完成所需时间的最优决策。通过实验对比分析，本文算法在云边端 3 层计算资源架构下的任务完成时间优化上具有良好表现。在下一步工作中，将进一步结合计算资源能耗、计算资源成本和通信网络负载等影响因素，构建更加符合实际场景的通信模型。

参考文献

[1] 曹畅，唐雄燕．算力网络关键技术及发展挑战分析[J]．信息通信技术与政策，2021(3).
[2] 雷波，刘增义，王旭亮，等．基于云,网,边融合的边缘计算新方案:算力网络[J]．电信科学，2019(9).

[3] ISLAM A. A survey on task offloading in multi-access edge computing[J]. Journal of Systems Architecture, 2021(4).
[4] TONG Z, DENG X, YE F, et al. Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment[J].Information Sciences: An International Journal, 2020(1).
[5] LIAO Z, PENG J, XIONG B, et al. Adaptive offloading in mobile-edge computing for ultra-dense cellular networks based on genetic algorithm[J]. Journal of Cloud Computing, 2021(1).
[6] LEI K, GUO P, ZHAO W, et al.A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem[J]. Expert Systems with Applications, 2022(11).
[7] DONG T, XUE F, XIAO C, et al. Task scheduling based on deep reinforcement learning in a cloud manufacturing environment[J]. Concurrency and Computation: Practice and Experience, 2020(11).
[8] LU H, GU C, LUO F, et al. Optimization of task offloading strategy for mobile edge computing based on multi-agent deep reinforcement learning[J]. IEEE Access, 2020(8).
[9] SEID A M, BOATENG G O, LU J, et al. Multi-agent drl for task offloading and resource allocation in multi-uav enabled iot edge network[J]. IEEE Transactions on Network and Service Management, 2021(4).
[10] NGUYEN D C, DING M, PATHIRANA P N, et al. Cooperative task offloading and block mining in blockchain-based edge computing with multi-agent deep reinforcement learning[J]. IEEE Transactions on Mobile Computing, 2021(4).
[11] ALKHALAILEH M, CALHEIROS R N, NGUYEN Q V, et al. Data-intensive application scheduling on mobile edge cloud computing[J]. Journal of Network and Computer Applications, 2020(10).
[12] SUN J, YIN L, ZOU M, et al. Makespan-Minimization workflow scheduling for complex networks with social groups in edge computing[J]. Journal of Systems Architecture, 2020(9).
[13] XU Z, ZHAO L, LIANG W, et al. Energy-Aware inference offloading for dnn-driven applications in mobile edge clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2021(10).
[14] ZHANG N, GUO S, DONG Y, et al. Joint task offloading and data caching in mobile edge computing networks[J]. Computer Networks, 2020(12).
[15] CHEN M, WANG T, ZHANG S, et al. Deep reinforcement learning for computation offloading in mobile edge computing environment[J]. Computer Communications, 2021(7).
[16] GUO K, YANG M, ZHANG Y, et al. Joint computation offloading and bandwidth assignment in cloud-assisted edge computing[J]. IEEE Transactions on Cloud Computing, 2019(10).

Task scheduling and resource allocation methods under cloud edge collaboration

XU Shuai-shuai, SU Min-jie, REN Xun, WU Yi-ye,YU Run-ze, HU Tao

(China Mobile Group Design Institute Co., Ltd. Zhejiang Branch, Hangzhou 310012, China)

Abstract To optimize the allocation of computing resources between multi-user terminals, multiple edge nodes, and cloud servers, and find the optimal task scheduling and resource allocation scheme. This paper proposes a three-layer computing resource allocation scheme based on the integration of computing and network. Considering the limited bandwidth of the communication network, task calculation priorities and task upload strategies were established, and multi node parallel computing rules were formulated in an environment with multiple edge nodes to maximize the utilization of computing resources. Deep reinforcement learning algorithms were trained to learn the optimal task scheduling and resource allocation strategies. The experimental results show that the method proposed in this article performs well in shortening the completion time of computational tasks, and still exhibits good robustness even with an increase in task data volume.

Keywords task scheduling; resource allocation; Markov decision process; deep reinforcement learning