基于深度强化学习的边云协同串行任务卸载算法

张凤荔[1,2*]，赵佳君[1,2]，刘 东[1,2]，王瑞锦[1,2]

(1. 电子科技大学信息与软件工程学院 成都 610054；2. 网络与数据安全四川省重点实验室 成都 610054)

【摘要】在移动边缘计算任务卸载问题中，传统卸载算法仅考虑移动设备和边缘服务器计算资源，在资源利用、系统效能上存在一定的局限性。该文基于 RainbowDQN 算法，考虑了延迟、能耗成本和服务质量保证等因素，提出了一种边缘云协同串行任务卸载算法 (ECWS-RDQN)。该算法通过对串行任务的权重分配，实现了网络边缘和云端协同的串行任务动态分配处理，为不同的用户设备应用提供近似最优的任务分配卸载策略。实验表明，ECWS-RDQN 算法比传统方案有更好的系统效能，提升了应用的服务质量。

关 键 词 深度 Q 网络； 边云协同； 移动边缘计算； 串行任务； 任务卸载算法

中图分类号 TP181 **文献标志码** A doi:10.12178/1001-0548.2021015

Edge Cloud Collaboration Serial Task Offloading Algorithm Based on Deep Reinforcement Learning

ZHANG Feng-li[1,2*], ZHAO Jia-jun[1,2], LIU Dong[1,2], and WANG Rui-jin[1,2]

(1. School of Information and Software Engineering, University of Electronic Science and Technology of China Chengdu 610054;
2. Network and Data Security Key Laboratory of Sichuan Province Chengdu 610054)

Abstract In the offloading problem of mobile edge computing task, the traditional offloading algorithm only considers the computing resources of mobile devices and edge servers, and has some limitations in resource utilization and system efficiency. this paper proposes an edge-cloud weighted serial task offloading algorithm based on rainbowDQN (ECWS-RDQN) based on the RainbowDQN algorithm, considering the factors of delay, energy consumption cost and service quality assurance. This algorithm realizes the serial task dynamic assignment processing of network edge and cloud collaboration through the weight to provide approximately optimal task assignment offloading strategies for different user device applications. Experiments show that the ECWS-RDQN algorithm has better system efficiency than the traditional schemes and improves the service quality of the applications.

Key words deep Q network; edge cloud collaboration; mobile edge computing; serial tasks; task offloading strategy

近年来，移动应用程序在各行各业都有着广泛的应用，但在实现诸如媒体处理、在线游戏、增强现实、虚拟现实和在移动设备中执行各种创新移动应用程序的时间、能量、成本和安全性等相关方面仍存在一些困难和挑战。由于这些移动设备的资源限制，为了满足应用程序运行的低延迟和高数据速率的需求，产生了一种新的计算模式——移动边缘计算 (mobile edge computing, MEC)[1]。

在 MEC 场景中，通过在用户设备周围的无线基站部署 MEC 服务器，将处于远距离云端的计算资源延伸至网络边缘，为用户设备提供物理距离更近的服务，降低由网络延迟造成的任务时延，同时也可以缓解应用数据传输对核心网造成的压力。当进行任务卸载操作时，用户设备将计算任务数据通过无线基站传输至 MEC 服务器上进行计算，MEC 服务器完成计算后将计算结果返回至用户设备，从

收稿日期：2021－01－12；修回日期：2021－03－12

基金项目：国家自然科学基金 (61802033, 61472064, 61602096)；四川省区域创新合作项目 (2020YFQ0018)；四川省科技计划重点研发项目 (2020YFG0475, 2018GZ0087, 2019YJ0543)；博士后基金 (2018M643453)；广东省国家重点实验室项目 (2017B030314131)；网络与数据安全四川省重点实验室开放课题 (NDSMS201606)

作者简介：张凤荔 (1963－)，女，主要从事网络安全和网络工程、云计算以及大数据和机器学习等方面的研究. E-mail：fzhang@uestc.edu.cn

而完成一次任务卸载操作，如此可以使得用户应用获得更低的延迟体验，变相地增加用户设备应用程序拥有的计算能力。

目前大量的研究是基于用户设备与 MEC 服务器之间的任务卸载，并未考虑云端依旧拥有大量的计算资源，且能够针对特定场景优化计算环境的情况。在大量移动设备和物联网设备的高度计算资源需求下，仅依靠 MEC 服务器进行计算是难以完成的，仍然会出现资源瓶颈。在考虑用户设备与 MEC 服务器之间的网络边缘任务卸载的同时，还需要云端的强大计算能力来辅助 MEC 服务器，通过边云协同来共同为移动设备提供相对低时延、高计算能力的服务。由于云端需要考虑网络传输时延和特定任务需求，在进行任务细粒度卸载时，对任务不同的环节需要有优先级考量，这也使得整个任务卸载工作的复杂程度增加。

近几年，移动边缘计算任务卸载问题一直是研究热点问题，因为任务卸载在决策时需要考虑诸多因素，在寻找最优解的过程中充满挑战。其中文献 [2] 针对 MEC 服务器和用户设备间通信和资源分配，提出了 hJTORA 启发式算法，在小规模的任务卸载中，实现了相对基线算法的更优解。文献 [3] 提出一种基于匈牙利和贪心算法的启发式算法，对密集网络下信道分配进行了优化。文献 [4] 针对资源受限的串行任务，提出一种基于化学反应优化算法的 MSTDOS 算法，实现对串行任务细粒度卸载的优化。文献 [5] 提出一种基于组合拍卖模型的算法，解决车联网中的任务卸载问题。文献 [6] 中提出了一种基于博弈论的启发式算法，从云−边−用户 3 层结构中寻找任务卸载策略。以上文献采用的是传统的启发式算法，在面对复杂环境时，计算能力会大幅下降，更好的解决办法是采用深度学习或强化学习的方式来处理高度复杂的场景中任务卸载问题。文献 [7] 提出一种 AHP 和 DQN相结合的任务卸载算法，解决在车辆网环境下车载设备与 MEC 服务器间的任务卸载问题。文献 [8] 基于 LSTM 和 HER 改进深度学习算法，解决单用户多服务器的任务卸载问题，对能耗时延费用等多种指标进行比较。文献 [9] 提出了一种基于 DQN 的 MEC 环境下的多任务卸载和资源分配算法。通过此方法，将混合整数非线性规划转变为一个 RL 问题，找到了更优的解决方案。

但上述研究仍存在以下不足：1) 没有考虑云边协同问题；2) 粗粒度的卸载算法缺乏灵活性。本文算法基于云−边−用户的 3 层结构，针对边云协同场景下的任务卸载资源分配进行研究，基于 Rainbow DQN[10] 算法提出了一种面向边云协同带有权重串行任务卸载算法 (edge-cloud weighted serial task offloading algorithm based on rainbow DQN, ECWS-RDQN)，考虑多用户的串行任务卸载对 MEC 服务器和云端的计算资源竞争，以及不同用户应用的优先级，以任务延迟时间、任务能量消耗和服务质量保证作为评价标准。

1　系统模型及问题定义

本文的任务卸载模型由多个移动用户设备 (UE)、拥有 MEC 服务器的无线基站 (BS) 和云端服务器 (CS) 组成，构成一个云−边−用户的 3 层结构，如图 1 所示。用户设备产生串行任务，MEC 服务器在中间层为用户和云端提供任务的资源分配、任务卸载调度工作，同时可以为用户分担一定程度的计算任务，云端则拥有强大的计算能力，可以更加快速地解决复杂计算问题，但相对边缘服务器会拥有更高的网络延迟。

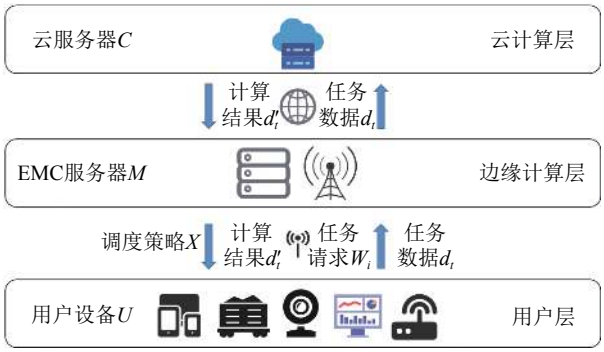

图 1　边云协同系统模型

其中，用户设备用集合$\{1,2,\cdots,U\}$表示，用户设备拥有有限的计算资源，并且仅带有一个串行任务 W_i；每个用户 u 可以用一个 5 元组表示 $u=\{i, W_i, T_i, R_i, Q_i\}$，$i\in\{1,2,\cdots,u\}$为用户编号；$W_i$是该用户设备当前拥有的串行任务；$T_u$是该用户设备当前任务的预计完成时间；$R_u$表示为用户设备与无线基站的通信带宽；$Q_u$表示为该用户设备的服务质量保证，该值越高代表该用户的服务优先级越高；系统中 MEC 服务器用 M 表示，$M=\{C_m, T_m, R_c\}$，其中 C_m为该 MEC 服务器的计算能力，T_m为该服务器当前任务的预计完成时间；云端服务器用C表示，$C=\{C_c, T_c\}$，其中C_c为该云服务器的计算能力，T_c为该服务器当前任务的预计完成时间。

用户设备在产生任务后，会将任务卸载决策请

求发送至 MEC 服务器，MEC 服务器会根据当前系统状态决定任务是在本地执行还是上传至 MEC 服务器或是云端执行。卸载策略由X表示，其中$X\in\{0,1,2,3\}$，0 表示本次策略为空操作，1 表示任务将会在本地执行，2 表示任务将会卸载至 MEC 服务器执行，3 表示任务将会卸载至云端服务器执行。

1.1 串行任务模型

本文中的任务假设为多个微任务串行组成的任务，一个任务应用由多个串行微任务组成，其中微任务中的起始和末尾代表该任务中的输入和输出部分，并且输入必须由本地完成。本文使用链表$W_i=\{t,l\},i\in\{1,2,\cdots,u\}$表示任务，$t$为当前微任务，$l$为下一个任务指针，其中用三元组$<c_t,d_t,\max T>$来表示微任务$t$，$c_t$为该任务计算所需的 CPU 周期数 (cycle)，d_t为该任务卸载时需要传输的数据量 (kb)，$\max T$为该任务的时延上限要求 (ms)。

串行任务的特点在于其任务之间的依赖关系，微任务 i−1 项完成才可以执行微任务 i，直至执行至最后一个微任务，该项串行任务才全部执行完成。在实际应用场景中，串行任务也是最为常见的任务类型之一，如图像识别等应用。

1.2 时延模型

时延在各种任务场景中都是一项至关重要的指标，时延表示一项任务从任务请求开始至任务全部完成的时间，并且时延大小决定了一项任务的完成时间是否可以满足应用需求。本节中的时延为用户设备完成其当前微任务的时延，分别从本地、边缘和云端 3 个部分讨论其任务时延的组成。

1.2.1 本地执行

在本地执行策略中，代表该微任务会在本地进行计算，不会将数据传输至 MEC 服务器，所以当前的任务时延dT_u表示为：

$$\mathrm{dT}_u^{\mathrm{local}}=\frac{c_t}{f_u} \tag{1}$$

式中，c_t为该任务所需的 CPU 周期数 (cycle)；f_u为用户设备的 CPU 频率 (Hz)。

1.2.2 边缘服务器执行

在边缘服务器执行策略中，代表该微任务将会通过无线传输进行任务卸载，把计算该微任务的必要数据传输至 MEC 服务器中。在此策略中，时延由计算时延、传输时延、排队时延组成，所以当前的任务时延dT_u表示为：

$$\mathrm{dT}_u^{\mathrm{up}}=\frac{d_t}{R_u}+\frac{c_t}{f_s}+\frac{d_t^{'}}{R_u}+T_m \tag{2}$$

式中，f_s为边缘服务器的 CPU 周期数；$d_t^{'}$为回传数据的数据量；R_u为用户设备和 MEC 服务器的传输速率。

1.2.3 云端执行

在云端执行策略中，代表该任务将会通过无线传输和主干网将任务所需的数据提交至云端服务器。在此策略中，时延由计算时延、传输时延、传播时延、排队时延组成，所以当前的任务时延dT_u表示为：

$$\mathrm{dT}_u^{\mathrm{upc}}=\frac{d_t+d_t^{'}}{R_u}+\frac{d_t+d_t^{'}}{R_c}+\frac{c_t}{f_c}+2\mathrm{dt}+T_c \tag{3}$$

式中，f_c为云服务器的 CPU 周期数；R_c为 MEC 服务器与云端服务器的传输速率；dt 为 MEC 服务器与云端的传播时延。

1.3 能耗模型

能耗在实际场景中也是十分重要的指标之一，采用电池供电的用户设备会对能耗大小更加敏感。在此场景中，能耗由 CPU 的计算和闲置消耗、无线传输时的能量消耗构成，以下将会从本地、边缘和云端 3 部分分别讨论系统能耗的组成。

1.3.1 本地执行

在本地执行策略中，将会使用本地用户设备的 CPU 执行计算任务，所以该策略的执行能耗为：

$$E_u^{\mathrm{local}}=\kappa(f_u)^2c_t \tag{4}$$

式中，κ为 CPU 的能量效率系数[11]；f_u为用户设备的 CPU 周期数。

1.3.2 边缘服务器执行

在边缘服务器执行策略中，计算任务将会提交至 MEC 服务器进行计算，在提交和计算的过程中，能耗由用户设备的传输能耗和空闲能耗组成：

$$E_u^{\mathrm{up}}=p_{\mathrm{up}}\frac{d_t}{R_u}+p_{\mathrm{idle}}\frac{c_t}{f_s} \tag{5}$$

式中，P_{up}为用户设备在无线传输时消耗的能量；P_{idle}为用户设备空闲时消耗的能量。

1.3.3 云端执行

在云端执行策略中，计算任务数据将会通过无线和主干网传输至云端服务器，能耗由用户设备的传输能耗和空闲能耗组成：

$$E_u^{\mathrm{upc}}=p_{\mathrm{up}}\frac{d_t}{R_u}+P_{\mathrm{upc}}\frac{d_t}{R_c}+p_{\mathrm{idle}}\left(\frac{d_t}{R_u}+\frac{d_t}{R_c}+2\mathrm{dt}\right) \tag{6}$$

1.4　服务质量保证

服务质量保证在任务调度场景中是决定不同优先级的用户服务质量的关键，使得具有更低时延或能耗要求的用户在资源竞争时会在请求中更有优势。在此场景中，服务质量保证的组成为：

$$\mathrm{Qos}_u = \alpha + \beta q_u \tag{7}$$

式中，$\alpha+\beta=1,\alpha,\beta\geqslant 0$；$\alpha$为基础值；$\beta$为该目标函数中服务质量占比；$q_u$为各用户的优先级。

1.5　问题定义

本文的目标是使用 RainbowDQN 生成决策实现时延和能耗的降低，前文已经给出了时延和能耗模型，采用线性加权的方式来规划目标函数。因此，原问题可以定义为：

$$F=\min\sum_{N}^{u}(\lambda_e E_{u,l}+\lambda_t \mathrm{dT}_{u,l}-\lambda_q \mathrm{Qos}_u) \tag{8}$$

式中，$\lambda_e,\lambda_t,\lambda_q\in[0,1]$，并且$\lambda_e+\lambda_t+\lambda_q=1$，分别表示能耗、时延和服务质量保证在目标函数中的占比。

2　算法模型

本文研究的任务卸载问题根本上是一个多目标优化问题，通常使用启发式算法或机器学习方法。在启发式算法中一般使用遗传算法、粒子群算法、化学反应算法等；在机器学习领域更多采用的是强化学习方法[12]，如 Q 学习、DQN 等[13]。在系统环境相对复杂的条件下，往往采用 DQN 来进行更高效的任务卸载分配工作处理。本文基于改进后的 DQN−rainbow DQN。rainbow DQN 是在原始 DQN 的基础上结合了 Double DQN、优先经验回放 (prioritized experience replay)、决斗网络 (duelling network)、多步学习 (multi-step learning)、分布式网络 (distributional network)、噪声网络 (noisy network)[14-18] 后改进的 DQN 算法，在训练速度、样本效率和性能方面与初始 DQN 算法相比有显著的增强。

DQN 作为强化学习的一种，其核心思想是通过获取环境状态和输出动作互动后的奖励，使用神经网络来近似值函数，使用经验回放来储存之前经历的数据，在更新参数时在其中选取一部分来使用。rainbowDQN 采用 multi-step learning 更新 Loss 函数：

$$(R_t^{(n)}+\gamma_t^{(n)}\max_{a'}q_{\bar{\theta}}(\boldsymbol{S}_{t+n},a')-q_\theta(\boldsymbol{S}_t,\boldsymbol{A}_t))^2 \tag{9}$$

式中，$\boldsymbol{A}$ 和 $\boldsymbol{S}$ 分别为系统状态和动作；γ为折扣因子；R为奖励函数；$\bar{\theta}$为目标网络参数；θ根据损失来更新；$\boldsymbol{S}_t$为迭代时动作$\boldsymbol{A}_t$的观测值；a'为在$\boldsymbol{S}_{t+n}$状态下奖励最优解。之后最小化 Loss 函数来更新网络参数θ。

在更新后存入经验池时，采用的是优先经验值回放，根据损失函数来决定该项采样的权值p_t：

$$p_t\propto\left|r_{t+1}+\gamma\max_{a'}q_{\bar{\theta}}(\boldsymbol{S}_{t+1},A')-q_\theta(\boldsymbol{S}_t,A_t)\right|^w \tag{10}$$

式中，p_t将会从损失函数中获取；w为优先经验回放的优先级因子。基于上述改进以及其余扩展，在本文提出的 ECWS-RDQN 中，还需根据系统需求重新定义状态空间、动作空间以及奖励函数，在奖励函数中引入权重来更加贴合实际的运行场景，优化串行任务调度策略。

其中动作决策值$X\in\{0,1,2,3\}$，表示对任务 w 做出的决策，0 表示该次请求暂时跳过；1 表示该任务将在本地执行；2 表示该任务将卸载至 MEC 服务器执行；3 表示该任务将卸载至云端执行。

在奖励函数设计中，考虑到能耗、时延和服务质量等优先级问题，不同的用户设备本身的优先级、任务组中某些微任务在不同计算场景下的效率或是系统的费用成本，都会影响每个微任务的权重。并且由于串行任务的特殊性，会导致多个任务交叉进行，仅对单个微任务的优化可能会导致某些串行任务总时延或总成本升高，所以在对单个微任务计算时延、能耗的同时，也需要对串行任务进行约束。

本文确定各项串行任务权重时，考虑当前任务计算成本标准d_u、参考容忍时延标准t_u、用户设备服务质量保证优先级q_u这 3 项评价因素。权重矩阵$\boldsymbol{A}$表示为：

$$\boldsymbol{A}=\begin{bmatrix} d_1 & t_1 & q_1 \\ d_2 & t_2 & q_2 \\ \vdots & \vdots & \vdots \\ d_u & t_u & q_u \end{bmatrix} \tag{11}$$

计算成本和参考容忍时延使用 z-score 标准化所有用户的实际使用情况作为参考标准，质量保证为用户预设优先级。成本计算和容忍时延计算标准化如下：

$$\begin{cases} \bar{x}=\dfrac{1}{n}\displaystyle\sum_{i=1}^{n}x_i \\ \sigma=\sqrt{\dfrac{1}{n-1}\displaystyle\sum_{i=1}^{n}(x_i-\bar{x})^2} \\ y_i=\dfrac{x_i-\bar{x}}{\sigma} \end{cases} \tag{12}$$

利用式 (12) 求得当前用户正规化值，即得到该用户与该环境中其余用户在时延要求和成本要求的相对值。再根据此相对值进行归一化和区间调整，即可获取各个用户的权重向量，从而优化系统中串行任务的卸载执行效率，使得偏离标准的用户可以在下一次请求中拥有更高的优先级，从而提升系统的稳定性。

结合目标函数，采用 min-max 标准化方法对综合能耗、时延和服务质量 3 项评价指标进行归一化，最后得到奖励函数为：

$$R=\max\sum_{N}^{u}(1-\boldsymbol{A}_i(\lambda_e E_{u,l}+\lambda_t \mathrm{dT}_{u,l}-\lambda_q \mathrm{Qos}_u)) \tag{13}$$

在 ECWS-RDQN 算法中，当 MEC 服务器接收到用户设备发送的任务卸载请求后，获取当前系统状态，通过计算获得权重向量，可以根据当前状态得到最优的卸载策略输出。综合以上模型，ECWS-RDQN 算法如下。

算法 1　ECWS-RDQN 算法

输入：状态向量$\boldsymbol{S}_t$；

输出：动作决策值X_t；

初始化权重矩阵$\boldsymbol{A}$及经验池；

For each iteration

初始化状态$\boldsymbol{S}_1$以及预处理序列；

For each step:

以ε的概率选择随机动作X_t，否则选用最优奖励动作$X_t=\max Q(\phi(S_t),X;\theta)$；

向系统执行动作X_t；

记录该动作下系统和用户状态，根据用户状态更新权值矩阵$\boldsymbol{A}_i$；

根据权值矩阵和系统状态计算奖励值，返回奖励值r_t和状态$\boldsymbol{S}_{t+1}$；

令$\boldsymbol{S}_{t+1}=\boldsymbol{S}_t$，计算$\phi_{t+1}=\phi(\boldsymbol{S}_{t+1})$

将向量$(\phi_t,X_t,r_t,\phi_{t+1})$存放至经验池；

从经验池中按优先经验回放策略抽取向量；

更新网络参数θ；

若$\boldsymbol{S}_{t+1}$为结束状态，则结束本轮迭代；

End For

End For

3　实验设计与结果分析

本节通过 python 语言对本文提出算法进行仿真以评估其性能，仿真场景为云−边−用户 3 层结构，由一个云端服务器、一个带有 MEC 服务器的无线基站和多个用户设备组成。主要通过与本地计算 (local)、全卸载至边缘 (MEC)、全卸载至云端 (cloud)、随机 (平均) 分配 (random) 方案、文献 [8] 中的 DQN 方案和仅使用基于 RainbowDQN 算法对比本文给出的 ECWS-RDQN 算法。在时延、能量、服务质量等约束条件下的策略对比，来验证算法的有效性。其中环境参数设置参考文献 [3, 8]，如表 1 所示。

表 1　环境参数设置

类型	参数设置
云端服务器计算能力/GHz	80
MEC服务器计算能力/GHz	10
用户设备计算能力/GHz	2
用户设备与基站通信带宽/Mb·s^{-1}	100
基站与云端通信带宽/Gb·s^{-1}	1
基站与云端传播时延/ms	10
用户设备发送功率/W	3
用户设备待机功率/W	0.5
微任务计算能力(低/高)	[1×10^6～3×10^6]、[9×10^6～1.1×10^7]
微任务数据大小(低/高)/kb	40～60，200～300
CPU能量效率k	5×10^{-27}
$\lambda_e,\lambda_t,\lambda_q$权值	0.4、0.4、0.2

ECWS-RDQN 算法首先针对 Rainbow DQN 进行优化，将串行任务拆解成多个微任务后需对奖励值获取频率以及数值标准化进行处理。其中考虑环境复杂度和学习速度，学习速率设置为$r=0.001$，multi-step 值为 10；考虑其经验-探索策略，折扣因子$\gamma=0.99$，优先经验回放参数采用$\alpha=0.5$、$\beta=0.4$[16]，噪声网络标准差为$\sigma=0.01$。最终本文算法的 loss 值收敛如图 2 所示，在 150 000 步时基本收敛，因评估中仍会使用贪婪策略跳出，因此在步数较高等情况下还会出现波动。

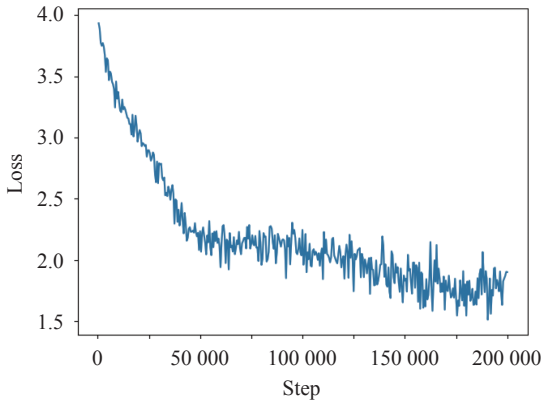

图 2　Loss 值随学习步数的变化

在实验中使用表 1 给出的环境配置，使用 10 个用户设备来生成串行任务，总计需完成 100

个串行任务，每个串行任务由 5 个微任务组成，通过调整串行任务的生成速率和计算复杂度进行对比。

在图 3 中，y 轴目标函数值由目标函数归一加权后得到，x 轴为串行任务卸载请求的间隔时间，以 10 ms 为单位时间。对比发现在任务生成间隔较大、总的任务计算压力较低时，卸载至边缘服务器的效果较好。随着间隔变短，计算传输量都增加的情况下，仅靠边缘服务器已经无法承载。而使用本文提供的 ECWS-RDQN 算法后，各种情况下的表现都远优于传统策略，相较于 DQN 算法也有一定优势。

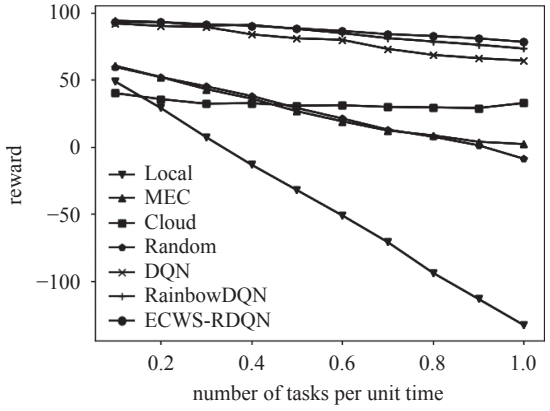

图 3　目标函数值随串行任务生成速度的变化

同样，在固定任务间隔的情况下，改变串行任务中高计算复杂度的占比，如图 4 给出的目标函数数值曲线，在计算复杂度快速升高的情况下，计算能力较低的本地设备难以完成任务；边缘服务器由于计算量超过承载能力出现任务堆积，拥有高计算能力和高网络延迟的云端几乎不受影响。图 5 和图 6 分别从系统时延和能耗开销两方面进行评价，针对串行任务的权值分配策略也在任务复杂度较高的情况下，有效降低了平均时延和成本。相较于其他策略和算法，本文给出的 ECWS-RDQN 算法总体稳定，能够提供更低时延、更低能耗的服务，有效提升了服务质量。

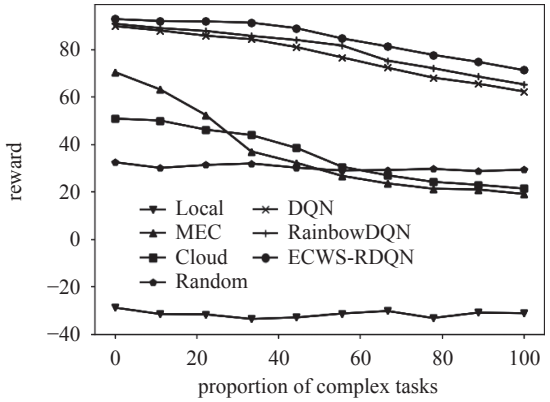

图 4　目标函数随任务计算复杂度的变化

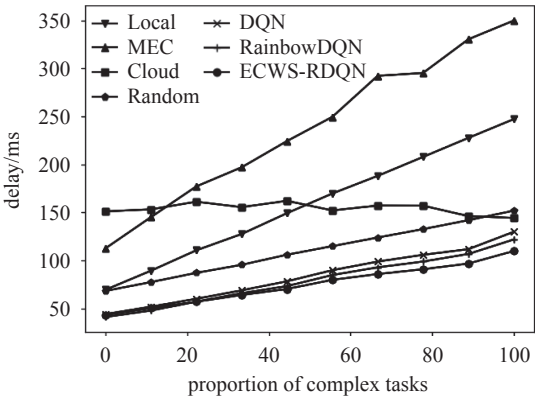

图 5　串行任务完成时延随任务计算复杂度的变化

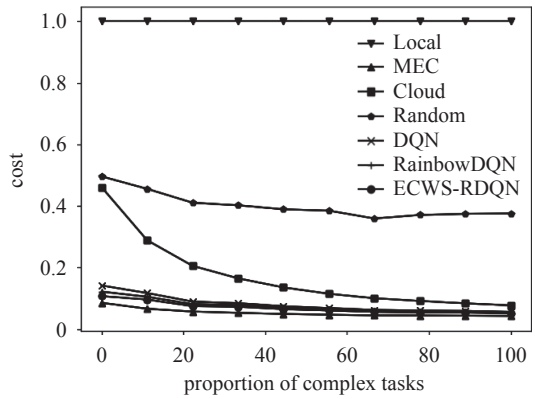

图 6　串行任务能耗成本随任务计算复杂度的变化

而在图 7 中，改变不同的计算复杂度观察 ECWS-RDQN 算法在不同情况下对不同优先级的用户组的调度优化，其中奖励值为时延、能耗的奖励值归一化得到。可以看出随着优先级的提高，更高优先级的用户可以在时延和能耗上获得更好的服务。

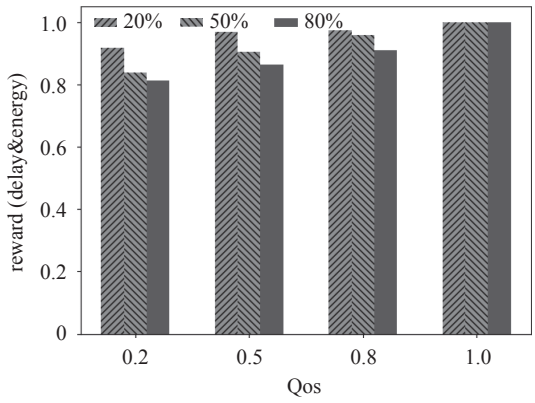

图 7　不同优先级用户组随计算复杂度的目标函数值变化

4　结 束 语

为面对边云协同串行任务卸载调度场景，本文提出了 ECWS-RDQN 算法，并且针对串行任务特点进行优化，在时延、能量和服务质量的约束条件

下能够实现智能任务卸载，并且通过实验验证了其有效性，本算法在各项参数下均有较明显优势。今后将会考虑算法的细节优化，同时在通信资源分配和多服务器调度等方面进行研究。

参 考 文 献

[1] 谢人超, 廉晓飞, 贾庆民, 等. 移动边缘计算卸载技术综述[J]. 通信学报, 2018, 39(11): 138-155.
XIE Ren-chao, LIAN Xiao-fei, JIA Qing-min, et al. Survey on computation offloading in mobile edge computing[J]. Journal on Communications, 2018, 39(11): 138-155.

[2] TRAN T X, POMPILI D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks[J]. IEEE Transactions on Vehicular Technology, 2018, 68(1): 856-868.

[3] 张海波, 李虎, 陈善学, 等. 超密集网络中基于移动边缘计算的任务卸载和资源优化[J]. 电子与信息学报, 2019, 41(5): 1194-1201.
ZHANG Hai-bo, LI Hu, CHEN Shan-xue, et al. Computing offloading and resource optimization in ultra-dense networks with mobile edge computation[J]. Journal of Electronics & Information Technology, 2019, 41(5): 1194-1201.

[4] 刘伟, 黄宇成, 杜薇, 等. 移动边缘计算中资源受限的串行任务卸载策略[J]. 软件学报, 2020, 31(6): 1889-1908.
LIU Wei, HUANG Yu-cheng, DU Wei, et al. Resource-constrained serial task offload strategy in mobile edge computing[J]. Journal of Software, 2020, 31(6): 1889-1908.

[5] 张海波, 栾秋季, 朱江, 等. 基于移动边缘计算的 V2X 任务卸载方案[J]. 电子与信息学报, 2018, 40(11): 2736-2743.
ZHANG Hai-bo, LUAN Qiu-ji, ZHU Jiang, et al. V2X task offloading scheme based on mobile edge computing[J]. Journal of Electronics & Information Technology, 2018, 40(11): 2736-2743.

[6] 王艺洁, 凡佳飞, 王陈宇. 云边环境下基于博弈论的两阶段任务迁移策略[J/OL]. 计算机应用. [2020-12-28]. http://kns.cnki.net/kcms/detail/51.1307.TP.20201017.1757.002.html.
WANG Yi-jie, FAN Jia-fei, WANG Chen-yu. Two-stage task offloading strategy based on game theory in cloud-edge environment[J/OL]. Journal of Computer Applications. [2020-12-28]. http://kns.cnki.net/kcms/detail/51.1307.TP.20201017.1757.002.html.

[7] 赵海涛, 张唐伟, 陈跃, 等. 基于 DQN 的车载边缘网络任务分发卸载算法[J]. 通信学报, 2020, 41(10): 172-178.
ZHAO Hai-tao, ZHANG Tang-wei, CHEN Yue, et al. Task distribution offloading algorithm of vehicle edge network based on DQN[J]. Journal on Communications, 2020, 41(10): 172-178.

[8] 卢海峰, 顾春华, 罗飞, 等. 基于深度强化学习的移动边缘计算任务卸载研究[J]. 计算机研究与发展, 2020, 57(7): 1539-1554.
LU Hai-feng, GU Chun-hua, LUO fei, et al. Research on task offloading based on deep reinforcement learning in mobile edge computer[J]. Journal of Computer Research and Development, 2020, 57(7): 1539-1554.

[9] HUANG L, FENG X, FENG A, et al. Distributed deep learning-based offloading for mobile edge computing networks[J]. Mobile Networks and Applications, 2018, DOI: 10.1007/s11036-018-1177-x.

[10] HESSEL M, MODAYIL J, VAN HASSELT H, et al. Rainbow: Combining improvements in deep reinforcement learning[EB/OL]. [2020-06-14]. https://arxiv.org/pdf/1710.02298.pdf.

[11] CHEN X. Decentralized computation offloading game for mobile cloud computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 26(4): 974-983.

[12] SHAKARAMIA,GHOBAEI-ARANIM,SHAHIDINEJAD A. A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective[J]. Computer Networks, 2020, DOI: 10.1016/j.comnet.2020.107496.

[13] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[EB/OL]. [2020-06-20]. http://www.arxiv.org/pdf/1312.5602.pdf.

[14] VAN H H, GUEZ A, SILVER D. Deep reinforcement learning with double Q-learning[EB/OL]. [2020-07-10]. https://arxiv.org/pdf/1509.06461.pdf.

[15] SCHAUL T, QUAN J, ANTONOGLOU I, et al. Prioritized experience replay[EB/OL]. [2020-07-10]. https://arxiv.org/pdf/1511.05952.pdf.

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

[17] BELLEMARE M G, DABNEY W, MUNOS R. A distributional perspective on reinforcement learning [EB/OL]. [2020-07-15]. https://arxiv.org/pdf/1707.06887.pdf.

[18] FORTUNATO M, AZAR M G, PIOT B, et al. Noisy networks for exploration[EB/OL]. [2020-07-18]. https://arxiv.org/pdf/1706.10295.pdf.

编 辑 叶 芳