朱绍恩. 缓存辅助的 MEC 卸载和资源分配研究[J]. 智能计算机与应用,2024,14(10):149-157. DOI:10.20169/j.issn.2095-2163.241021

缓存辅助的 MEC 卸载和资源分配研究

朱绍恩
(南京邮电大学 通信与信息工程学院,南京 210003)

摘 要:由于 MEC 在面对大规模数据传输和计算需求时存在性能瓶颈,在实际应用中,通过将任务数据缓存到本地,可以大大减少数据传输时间和网络带宽的消耗,提高 MEC 的性能。本文研究了多边缘服务器多移动用户缓存辅助场景下的系统长期平均开销优化问题,提出了一种计算卸载、资源分配和缓存决策联合优化方案。该方案基于 D3QN 框架,利用了遗传算法和 KKT 分别获得本地和边缘计算资源分配,基于任务请求概率分布更新 MEC 服务器缓存空间,并通过 D3QN 网络的学习和保序量化得到近似最优的卸载决策。仿真结果表明,在不同系统参数下,本文所提出方案相较于其他方案具有更佳的性能。

关键词:移动边缘计算;深度强化学习;计算卸载;资源分配;数据缓存
中图分类号:TP391 **文献标志码**:A **文章编号**:2095-2163(2024)10-0149-09

Research on cache-assisted MEC offloading and resource allocation

ZHU Shao'en
(School of Communications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: Due to the performance bottleneck of MEC in facing large-scale data transmission and computing demands, caching task data locally can significantly reduce data transfer time and network bandwidth consumption, thereby improving the performance of MEC in practical applications. This paper investigates the optimization problem of long-term average system cost in a scenario of multiple edge servers and multiple mobile users with cache assistance, and proposes a joint optimization scheme for computing offloading, resource allocation, and caching decision-making. The proposed scheme is based on the D3QN framework and utilizes genetic algorithm and KKT to respectively obtain local and edge computing resource allocation. And the scheme updates the MEC server cache space based on the probability distribution of task requests and obtains an approximately optimal offloading decision through learning and quantization of the D3QN network. Simulation results show that the proposed scheme has better performance compared with other schemes under different system parameters.

Key words: mobile edge computing; deep reinforcement learning; computing offloading; resource allocation; data caching

0 引 言

随着物联网和 5G 的发展,智能互联应用如 VR、AR、人脸识别、自动驾驶等迅速兴起[1]。但移动设备计算能力有限,部分任务需上传至云服务器协助计算,导致数据处理出现时效性和安全性问题[2]。移动边缘计算(Mobile Edge Computing, MEC)可以解决此问题。MEC 将计算、存储和网络资源推向网络边缘,提供强大的计算能力,接近设备,避免核心网络拥塞。MEC 利用网络边缘的闲置计算能力和存储空间,能处理需要实时响应或高计算复杂度的任务[3-4]。

目前针对 MEC 的研究中,文献[5-6]关注于优化单用户任务的时延问题。文献[5]关注 0-1 方式下的任务卸载决策,目的是最小化任务的执行时延。文献[6]则研究了部分卸载模式下的最优卸载决策,通过对任务进行数据分割,降低了任务的执行时延。文献[7-8]则关注多用户 MEC 系统中的时延优化问题。文献[7]通过部分卸载任务来最大限度地降低任务处理时延,但未考虑用户设备的本地能耗约束。相比之下,文献[8]研究了用户设备的本地能耗对任务卸载决策的影响,提供了一种计算资

基金项目:江苏省重点研发计划(BE2020084-3,BE2021013-2)。
作者简介:朱绍恩(1997-),男,硕士研究生,主要研究方向:边缘计算,人工智能。Email:171350844@qq.com
收稿日期:2023-05-08

源分配的优化方案。文献[9-11]深入研究了 MEC 系统的能源消耗问题。其中,文献[9]考虑了任务计算和文件传输所消耗的能量,设计了一种节能方案,通过计算任务的卸载和无线电资源的优化分配,在满足时延约束的同时最大程度地降低了能耗。文献[10]研究了面向多用户的 MEC 系统、需要将多个智能移动设备的计算任务卸载到 MEC 服务器的情况,提出了卸载决策、计算资源分配和无线电资源分配在内的联合优化方法。文献[11]针对小规模网络,提出了一种计算卸载的能效管理方案,该研究将计算分流决策、功率和频谱进行联合优化,旨在最大程度地减少所有用户设备的能耗。

然而,由于网络传输和计算复杂度的限制,MEC 在处理大规模数据传输和计算任务时仍然存在性能瓶颈。为了提高 MEC 系统的性能,缓存技术被引入到 MEC 中,通过在本地缓存数据来减少数据传输时间和网络带宽的消耗[12]。文献[13]研究了配备共享缓存的 IoV 中的完整任务卸载问题,以提高用户体验的质量。文献[14]通过最小化系统成本来解决超密集 MEC 网络中的移动感知内容缓存和用户关联问题,并提出了一种基于移动的在线缓存算法。

本文提出了一种计算卸载、资源分配和缓存决策联合优化方案,旨在有效地利用 MEC 服务器的缓存空间和优化资源分配,以提高 MEC 系统的性能和降低系统开销。该方案采用了基于深度强化学习的算法来优化计算卸载、资源分配和缓存决策,并通过仿真实验验证了该方案的有效性和性能优势。该研究对于 MEC 系统中的缓存技术应用和性能优化具有一定的参考价值。

1 系统模型

系统模型如图 1 所示。本文考虑一个多 MEC 服务器多移动设备(Mobile Device, MD)组成的 MEC 系统,其中每个 MEC 服务器部署在一个无线接入点附近并为其覆盖范围内的 MD 提供计算和缓存等服务。系统共有 M 个小区,每个小区有一个无线接入点和 MEC 服务器。每个小区内随机分布一些 MD,可以在小区之间移动,并采用 0-1 卸载模式。系统以时隙结构运行。记 $M^{mm}=\{1,2,\cdots,M\}$ 表示所有小区的集合, $N^{mm}=\{1,2,\cdots,N^m\}$ 表示第 m 个小区内的 MD 集合, $N^m=\{1,2,\cdots,N\}$ 表示整个系统内的 MD 集合, $N=\sum_{m=1}^{M}N^m$, $T^m=\{1,2,\cdots,T\}$ 表示所有时隙的集合,每个 MEC 服务器具有 D(比特)的存储空间用于存储缓存的任务数据,计算能力为 F(CPU 周期/s)。

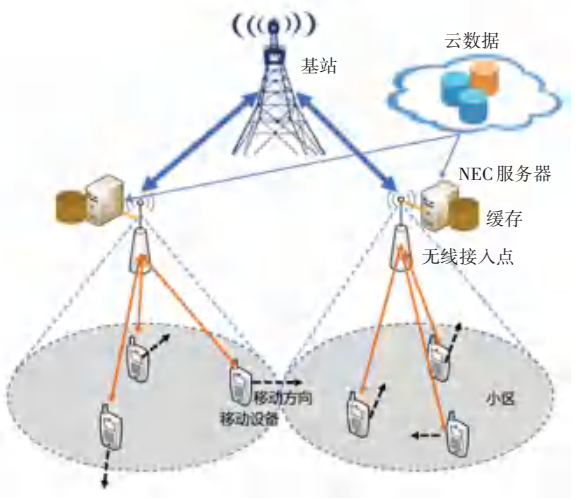

图 1 系统模型

Fig. 1 System model

1.1 任务模型

假设总共有 K 个异构计算任务,由集合 $K=\{1,2,\cdots,K\}$ 表示,每个任务 $k\in K$ 有不同的数据量、计算资源需求,用 $\{b_k,d_k\}$ 表示第 k 个任务的属性,其中 b_k 表示任务的输入数据大小, d_k 表示计算任务所需的计算资源。当任务被请求时,MD 或 MEC 服务器使用输入数据执行计算,获得计算结果。

每个用户在每个时隙开始从任务集合中随机请求一个计算任务,定义任务请求变量 $k_{n,t}^m\in K$。每个用户对不同任务的偏好程度不同,即请求每个任务的概率不同。假设用户从 K 中请求任务的概率服从 Zipf 分布[15-16],请求的任务的动态概率分布表示为 $\phi_t=\{\phi_{1,t},\phi_{2,t},\cdots,\phi_{k,t}\}$,进而推得:

$$\phi_{k,t}=\frac{z_{k,t}^{-\eta}}{\sum_{l=1}^{K}z_{l,t}^{-\eta}} \tag{1}$$

其中, $\phi_{k,t}$ 表示第 k 个任务的请求概率; $z_{k,t}$ 表示每个任务的请求概率排序;Zipf 参数 $\eta\geqslant 0$ 用于控制请求概率偏差。任务间请求概率差异随着 η 增加而变大,请求概率越高表示任务越容易被重复请求。

1.2 缓存模型

任务缓存是指在 MEC 服务器缓存任务输入数据,可以降低 MD 的能耗和任务卸载的传输时延。假设云端拥有所有任务的输入数据,MEC 服务器可以从云端或者 MD 卸载获取任务数据并缓存,所有 MEC 服务器共享同样的缓存内容。对于所有 MEC

服务器，定义 $c_{k,t} \in \{0,1\}$ 表示缓存决策变量，$c_{k,t}=1$ 表示将计算任务 k 的输入数据在 t 时隙缓存在 MEC 服务器中，并且可以在 $t+1$ 时隙中使用，$c_{k,t}=0$ 表示不缓存。

定义 MEC 服务器的缓存任务集合 $K_t^c \triangleq \{k: c_{k,t-1}=1\}$。如果用户请求的任务在 MEC 服务器缓存中，那么任务无需卸载，由 MEC 服务器直接计算。如果用户请求的任务不在 MEC 服务器缓存中，那么用户设备需要在本地执行任务或者将任务卸载到其所在小区的 MEC 服务器 m 中执行。

1.3　移动性模型

用户在小区的平均驻留时间为 β，表示用户的移动强度服从高斯分布，可以通过分析该小区所有用户的历史驻留数据获得。如果用户选择在 MEC 处理任务，假设所需时间为 T_t^n，那么当 MEC 服务器准备发回计算结果时，用户还在原小区的概率表示为：

$$p_1 = \int_0^{T_t^n} \frac{1}{\beta} e^{\frac{-\tau}{\beta}} \mathrm{d}\tau = 1 - e^{\frac{-T_t^n}{\beta}} \tag{2}$$

离开的概率表示为 $p_{\text{move}} = 1 - p_1$。

如果 MEC 准备发回计算结果时，用户已经离开原小区，则计算结果需要先转发给用户当前所在小区的 MEC，此过程将产生迁移开销 C_{move}。假设迁移开销正比于计算结果大小，计算结果大小正比于输入数据大小，因此迁移成本可表示为 $C_{\text{move}} = \gamma b_{k_{n,t}^m}$，其中 γ 表示每比特任务的迁移开销。

1.4　本地计算模型

假设 MD 的计算资源为 $f_{n,t}^{m,l}$，当 MD 选择完全在本地计算任务时，那么本地计算时延为：

$$T_{n,t}^{m,l} = \frac{d_{k_{n,t}^m}}{f_{n,t}^{m,l}} \tag{3}$$

对应的本地计算能耗为：

$$E_{n,t}^{m,l} = K \cdot d_{k_{n,t}^m} \cdot (f_{n,t}^{m,l})^2 \tag{4}$$

其中，K 表示取决于 MD 芯片架构的有效开关电容。

1.5　边缘计算模型

当 MD 选择完全在边缘计算任务时，时延主要分为以下 3 部分：

(1)任务从用户上传到 MEC 服务器的上行链路时延 $T_{n,t}^{m,u}$。

(2)边缘服务器的任务计算时延 $T_{n,t}^{m,e}$。

(3)任务计算后的结果回传给用户的下行传输时延。一般来说，任务输出数据比输入数据小得多，边缘服务器之间传输速率和下行传输速率比上行速率大得多，因此不考虑第 3 部分的时延。

如果 $c_{k_{n,t}^m,t-1}=1$，即对于请求任务在缓存中的用户，采用本地 MEC 服务器计算的方式，任务数据不需要上传给 MEC 服务器，时延只需考虑第 2 部分，表示为：

$$T_{n,t}^{m,e} = \frac{d_{k_{n,t}^m}}{f_{n,t}^m} \tag{5}$$

其中，$f_{n,t}^m$ 表示 MEC 服务器 m 分配给该小区用户 n 的计算资源。

如果 $c_{k_{n,t}^m,t-1}=0$，即对于请求任务不在缓存中用户，需要将任务数据上传给本地 MEC 服务器进行计算，时延需同时考虑第 1、2 部分：

(1)任务上传阶段。数据传输速率表示为：

$$r_{n,t}^m = W\log_2\left(1 + \frac{p_{n,t}^m h_{n,t}^m}{\sigma^2 + I_{n,t}}\right) \tag{6}$$

其中，W 表示每个 MD 和无线接入点之间的通信带宽；$h_{n,t}^m$ 表示信道增益；σ^2 表示高斯白噪声；$I_{n,t}$ 表示 MD 的干扰信号，包括码间干扰、信道间干扰等；$p_{n,t}^m$ 表示上传功率。因此任务数据上传时延 $T_{n,t}^{m,u}$ 表示为：

$$T_{n,t}^{m,u} = \frac{b_{k_{n,t}^m}}{r_{n,t}^m} \tag{7}$$

对应的上传能耗 $E_{n,t}^{m,u}$ 表示为：

$$E_{n,t}^{m,u} = p_{n,t}^m T_{n,t}^{m,u} \tag{8}$$

(2)本地 MEC 服务器计算阶段，任务的计算时延表示见式(6)。

除了时延、能耗外，边缘计算中系统开销还包括任务结果迁移开销 C_{move}。

2　问题描述

本文的优化目标是设计最优的卸载策略、本地计算资源分配策略、边缘计算资源分配和缓存策略，在满足计算资源、缓存容量等约束条件的情况下来最小化由时延、能耗、计算结果数据迁移开销组成的系统的长期平均开销。

基于建立的系统模型和相关分析，MDn 请求任务的总处理时延 $T_{n,t}^m$ 可以表示为：

$$T_{n,t}^m = x_{n,t}^m\left((1 - c_{k_{n,t}^m,t-1})T_{n,t}^{m,u} + T_{n,t}^{m,e}\right) + (1 - x_{n,t}^m)T_{n,t}^{m,l} \tag{9}$$

其中，$x_{n,t}^m \in \{0,1\}$ 表示卸载决策变量，$x_{n,t}^m=1$ 表示任务卸载，$x_{n,t}^m=0$ 表示不卸载。

总能耗 $E_{n,t}^m$ 可以表示为：

$$E_{n,t}^m = (1 - x_{n,t}^m)E_{n,t}^{m,l} + x_{n,t}^m(1 - c_{k_{n,t}^m,t-1})E_{n,t}^{m,u} \quad (10)$$

因此，系统的即时开销表示为：

$$C_t = \sum_{m \in M} \sum_{n \in N^m} \mu_1 T_{n,t}^m + \mu_2 E_{n,t}^m + \mu_3 x_{n,t}^m p_{\text{move}} C_{\text{move}} \quad (11)$$

其中，μ_1，μ_2，μ_3 分别表示时延、能耗、任务结果迁移开销的权重系数，范围在[0,1]，用于权衡四者的重要程度。系统的长期平均开销为：

$$C = \frac{1}{|T|} \sum_{t \in T} C_t \quad (12)$$

综上，优化问题的公式和约束条件如下：

$$\begin{aligned} P1: & \min_{x_{n,t}^m, f_{n,t}^{m,l}, f_{n,t}^m, c_{k,t}} C \\ \text{s.t. } & \text{C1}: c_{k,t} \in \{0,1\} \\ & \text{C2}: x_{n,t}^m \in \{0,1\} \\ & \text{C3}: 0 \leqslant p_{n,t}^m \leqslant p_n^{m.\max} \\ & \text{C4}: 0 \leqslant f_{n,t}^{m,l} \leqslant f_l^{\max} \\ & \text{C5}: \sum_{n \in N_t^m} x_{n,t}^m f_{n,t}^m \leqslant F_t^m \\ & \text{C6}: 0 \leqslant f_{n,t}^m \leqslant F_t^m \\ & \text{C7}: \sum_{k \in K_t^c} b_k \leqslant D \end{aligned} \quad (13)$$

其中，F_t^m 表示 MEC m 服务器可用计算资源；C1 表示每个任务的缓存决策约束；C2 表示用户请求任务的卸载决策约束；C3 表示传输功率约束；C4 表示本地计算能力约束；C5 和 C6 表示 MEC 服务器分配给任务的计算资源约束；C7 表示 MEC 服务器缓存容量约束。

问题 $P1$ 是一个典型的混合整数非线性规划问题，很难用传统的数学方法直接解决。然而通过上述分析，当卸载决策 $x_{n,t}^m$ 和缓存空间 K_t^c 已知后，本地计算资源 $f_{n,t}^{m,l}$ 分配决策和 MEC 服务器计算资源 $f_{n,t}^m$ 分配决策优化是独立的，本地开销和边缘开销的优化也是独立的。

当 $x_{n,t}^m = 0$ 时，可以提取 $P1$ 问题中关于 $f_{n,t}^{m,l}$ 的项，将其表述为最优本地开销问题 $P2$：

$$\begin{aligned} P2: & \min_{f_{n,t}^{m,l}} C_{\text{local}} \\ \text{s.t. } & 0 \leqslant f_{n,t}^{m,l} \leqslant f_l^{\max}, \forall n \in N_t^m \end{aligned} \quad (14)$$

其中，$C_{\text{local}} = \mu_1 T_{n,t}^{m,l} + \mu_2 E_{n,t}^m$。

当 $x_{n,t}^m = 1$ 时，在当前系统中，上传任务数据的延迟、迁移开销都可以通过变量直接计算而得出。因此，优化边缘处理的时延可以使边缘开销最小化。提取 $P1$ 问题中关于 $f_{n,t}^m$ 的项，将其表述为最优边缘计算时延问题 $P3$：

$$\begin{aligned} P3: & \min \sum_{f_{n,t}^m} \sum_{n \in N_t^m} T_{n,t}^{m,e} \\ \text{s.t. } & \text{C1}: \sum_{n \in N_t^m} f_{n,t}^m \leqslant F_t^m \\ & \text{C2}: 0 \leqslant f_{n,t}^m \leqslant F_t^m \end{aligned} \quad (15)$$

当 $f_{n,t}^{m,l}$ 和 $f_{n,t}^m$ 已知时，将其带入优化问题 $P1$，则问题 $P1$ 可以转化为卸载决策和缓存决策问题 $P4$：

$$\begin{aligned} P4: & \min_{x_{n,t}^m, c_{k,t}} C \\ \text{s.t. } & \text{C1}: c_{k,t} \in \{0,1\} \\ & \text{C2}: x_{n,t}^m \in \{0,1\} \\ & \text{C3}: 0 \leqslant p_{n,t}^m \leqslant p_n^{m.\max} \\ & \text{C4}: \sum_{k \in K_t^c} b_k \leqslant D \end{aligned} \quad (16)$$

3 计算卸载、资源分配和缓存决策联合优化算法

基于所述优化问题，为了求解最优的计算卸载、资源分配和缓存更新方案，本节将优化问题分为 3 个子问题进行求解。首先，利用 D3QN（Double Dueling Deep Q-Network）求解离散计算卸载决策 $x_{n,t}$，采用保序量化（Order-Preserving，OP）算法对生成的卸载决策结果进行量化处理，得到多组可行解；接着，将可行解根据卸载决策分别代入到 $P2$ 和 $P3$ 问题中，采用遗传算法（Genetic Algorithm，GA）求解系统的最优本地计算资源 $f_{n,t}^{m,l}$，利用 KKT 求解最优边缘计算资源分配 $f_{n,t}^m$，进一步求出开销和奖励；然后基于任务请求概率求解 $P4$ 中的缓存决策 $c_{k,t}$，更新 MEC 服务器缓存空间；最后，经过交替迭代运行，积累经验池并从经验中学习，优化 D3QN，得到近似最优的决策和系统长期平均开销。接下来，将对问题的详细求解过程进行介绍。

3.1 卸载决策求解

由于基于深度强化学习（Deep Reinforcement Learning，DRL）的 D3QN 算法在处理高维状态空间、离散动作空间和非线性关系方面具有优势，本小节将采用 D3QN，为所有用户设备制定最优的二进制卸载决策 $x_{n,t}^m$。

首先定义深度强化学习求解问题关键要素：

（1）状态（State）：在 t 时刻，系统的状态包括所有用户设备请求任务的数据量 $\boldsymbol{b}_t$、计算量 $\boldsymbol{d}_t$、信道状态 $\boldsymbol{h}_t$、用户位置 $\boldsymbol{l}_t$、用户移动性 $\boldsymbol{p}_t^{\text{move}}$、MEC 服务

器可用计算资源 $\boldsymbol{F}_t$ 以及任务缓存情况 $\boldsymbol{c}_{k,t-1}$，因此定义状态空间 $\boldsymbol{s}_t$ 为：

$$\boldsymbol{s}_t = \{\boldsymbol{b}_t, \boldsymbol{d}_t, \boldsymbol{h}_t, \boldsymbol{l}_t, \boldsymbol{p}_t^{\text{move}}, \boldsymbol{F}_t, \boldsymbol{c}_{k,t-1}\} \tag{17}$$

其中，$\boldsymbol{b}_t \triangleq [b_{k_{1,t}}, \cdots, b_{k_{N,t}}]^{\mathrm{T}}$；$\boldsymbol{d}_t \triangleq [d_{k_{1,t}}, \cdots, d_{k_{N,t}}]^{\mathrm{T}}$；$\boldsymbol{h}_t \triangleq [h_{1,t}, \cdots, h_{N,t}]^{\mathrm{T}}$；$\boldsymbol{l}_t \triangleq [l_{1,t}, \cdots, l_{N,t}]^{\mathrm{T}}, l_{n,t} \in M$ 表示用户 n 所在的小区索引；$\boldsymbol{p}_t^{\text{move}} \triangleq [p_{1,t}^{\text{move}}, \cdots, p_{N,t}^{\text{move}}]^{\mathrm{T}}$；$\boldsymbol{F}_t \triangleq [F_t^1, \cdots, F_t^M]^{\mathrm{T}}$；$c_{k,t-1} \triangleq [c_{1,t-1}, \cdots, c_{K,t-1}]^{\mathrm{T}}$。

(2)动作(Action)：在 t 时刻，系统的动作包括所有用户设备的请求任务的离散卸载决策 $\boldsymbol{x}_t$，因此定义动作空间 $\boldsymbol{a}_t$ 为：

$$\boldsymbol{a}_t = \boldsymbol{x}_t \tag{18}$$

其中，$\boldsymbol{x}_t \triangleq [x_{1,t}, \cdots, x_{N,t}]^{\mathrm{T}}$。

(3)奖励(Reward)：定义系统在 t 时刻的奖励函数 r_t 为：

$$r_t = \alpha(C_{\text{local}} - C_t)/C_{\text{local}} \tag{19}$$

其中，C_{local} 表示系统内所有用户在本地计算任务的总开销，α 表示奖励调整系数。

接着，采用 D3QN 进行卸载决策求解。D3QN 是一种结合了 Double DQN 和 Dueling DQN 的深度 Q 学习算法，主要用于求解离散动作。通过使用 2 个深度神经网络(Deep Neural Networks，DNN)和拆分 Q 值函数为状态值函数和优势值函数，从而减轻过估计的问题和冗余计算，提高学习效果和稳定性[17-18]。

D3QN 根据 ε − greedy 策略选择一个动作 $\boldsymbol{x}_t$，即以 $1-\varepsilon$ 的概率选择当前最优动作 $\boldsymbol{x}_t = \text{argmax}_{x_t} Q(s, a;\theta)$，以 ε 的概率随机选择动作。D3QN 优化动作主要涉及目标函数计算和网络参数的更新，D3QN 的目标函数如下：

$$y_i = r_i + \gamma(1 - done_i) Q(s_i', \arg\max_a Q(s_i', a;\theta);\theta^-) \tag{20}$$

其中，r_i 和 s_i' 分别表示第 i 个样本的奖励和下一个状态；γ 表示折扣因子；$done_i$ 表示第 i 个样本是否达到终止状态；θ 为评估网络的参数；θ^- 为目标网络的参数。评估网络表示为：

$$Q(s,a;\theta) = V(s;\theta^V) + A(s,a;\theta^A) - \underset{a}{mean}(A(s,a;\theta^A)) \tag{21}$$

其中，$V(s;\theta^V)$ 为状态值网络，$A(s,a;\theta^A)$ 表示优势值网络。

在更新网络参数时，D3QN 算法采用最小化损失函数：

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i;\theta))^2 \tag{22}$$

其中，N 表示批次大小，s_i 和 a_i 分别表示第 i 个样本的状态和动作。

由于动作 $\boldsymbol{x}_t$ 是所有用户设备的卸载决策，每个卸载决策有 0 或 1 两种可能，所以 $\boldsymbol{x}_t$ 有 2^N 种可能的取值，DNN 会根据系统状态 $\boldsymbol{s}_t$ 和动作 $\boldsymbol{x}_t$ 输出 2^N 个 Q 值，然后选择使 Q 值最大的 $\boldsymbol{x}_t^*$。随着 N 的增大，这种复杂度是呈指数增加的，不利于模型的探索和收敛，同时对计算机的内存容量要求极高。根据文献[10]，可以通过 DNN 先输出一个松弛的卸载动作 $\boldsymbol{x}_t$，然后通过 OP 方法将其量化为 V 个二进制卸载动作，这里 V 是 $[1, N+1]$ 内的任意整数，基于奖励值选择一个最佳动作 $\boldsymbol{x}_v^*$。

OP 方法的基本思想是在量化过程中保持有序，即对任意量化动作 $\boldsymbol{x}_v$，如果 $x_{i,t} \geqslant x_{j,t}$，那么应保持 $x_{i,v} \geqslant x_{j,v}$，其中 $i, j \in N$。具体方法为：

(1)首先 DNN 经过最后一层 *Sigmoid* 激活后输出 N 个[0, 1]之间的小数，组成松弛动作 $\boldsymbol{x}_t$，其中 $\boldsymbol{x}_t = \{x_{i,t} \mid x_{i,t} \in [0,1], i \in N\}$。

(2)定义量化函数 $f(\boldsymbol{x}_t)$：$f(\boldsymbol{x}_t) = \{x_v \mid x_v \in \{0,1\}^N, v = 1,2,\cdots,V\}$。

(3)第 1 个二进制卸载决策可以表示为：

$$x_{i,1} = \begin{cases} 1, & x_{i,t} > 0.5 \\ 0, & x_{i,t} \leqslant 0.5 \end{cases} \tag{23}$$

(4)根据 $\boldsymbol{x}_t$ 中的元素到 0.5 的距离，将其进行排序，表示为 $|x_{(1),t} - 0.5| \leqslant |x_{(2),t} - 0.5| \leqslant \cdots \leqslant |x_{(N),t} - 0.5|$，其中 $\boldsymbol{x}_{(i),t}$ 表示该元素在 $\boldsymbol{x}_t$ 的第 i 个排序。

(5)基于 $x_{(v-1),t}$ 求剩余 $V-1$ 个量化动作 $\boldsymbol{x}_v$：

$$x_{i,v} = \begin{cases} 1, & x_{i,t} > x_{(v-1),t} \\ 1, & x_{i,t} = x_{(v-1),t} \wedge x_{(v-1),t} \leqslant 0.5 \\ 0, & x_{i,t} = x_{(v-1),t} \wedge x_{(v-1),t} > 0.5 \\ 0, & x_{i,t} < x_{(v-1),t} \end{cases} \tag{24}$$

其中，$v = 2, \cdots, V$。

通过 OP 获得量化动作后，将动作根据卸载决策分别代入到 $P2$ 和 $P3$ 中求解本地计算资源和边缘计算资源分配，进一步计算当前时刻系统开销和奖励，然后根据当前时刻用户请求任务情况更新缓存空间和状态空间，最后将当前状态、动作、奖励、新状态存入经验池，经过交替迭代运行，积累经验池并从经验中学习，优化 DNN，得到近似最优的决策和系统长期平均开销。

3.2 本地计算资源分配求解

GA 是一种基于生物学遗传进化理论的优化算法。模拟了生物进化的过程,通过对问题解空间中的个体进行交叉、变异和选择等操作,来逐步寻找最优解,全局优化能力强[19]。本小节将采用 GA 求解 $P2$, 通过最小化本地开销 C_{local} 为所有选择本地计算的用户设备制定最优的本地计算资源分配决策 $f_{n,t}^{m,l}$。

将 $f_{n,t}^{m,l}$ 表示为 $f_{n,t}^{m,l}=y_n^m f_l^{\max}$, 其中 $y_n^m \in [0,1]$ 表示本地资源分配比例。将 $y_{n,i}^m$ 设置为 GA 中的一个个体, Y 个个体作为一个种群。为了评估每个个体的适应程度,即本地资源分配比例 $y_{n,i}^m$ 对本地开销 C_{local} 的影响,将适应度函数定义为:

$$Fitness(y_{n,i}^m) = -(C_{\text{local}}(y_{n,i}^m) - \max_j C_{\text{local}}(y_{n,j}^m)) + 10^{-3} \tag{25}$$

其中, $i=1,\cdots,Y$, j 表示使 C_{local} 最大的个体索引。这样每个个体的适应度区间为[min C_{local}, max C_{Local}]加一个很小的正数。GA 的主要求解步骤如下所示。

(1)初始化种群:随机生成符合个体区间的 Y 个个体作为一个种群。

(2)评价个体适应度:通过上述定义的适应度函数评价每个个体的适应度。

(3)选择个体:根据个体适应度值,选择一部分优秀个体作为父代,用于生成下一代个体。

(4)交叉操作:对父代进行随机配对,通过交换基因生成新的子代个体。

(5)变异操作:对新生成的子代个体进行随机变异,引入新的基因信息,增加搜索空间的多样性。

(6)终止条件:根据预设的终止条件,判断是否达到最优解或搜索时间上限,若满足条件,则输出搜索结果,否则返回步骤(3)。

通过 GA 获得最优本地计算资源决策后,计算这种情况下的本地开销,用于后续进一步计算系统总开销和奖励值。

3.3 边缘计算资源分配求解

Karush-Kuhn-Tucker(KKT)条件是解决最优化问题时常用的一种方法,是非线性规划最佳解的必要条件[20]。本小节将利用 KKT 条件求解 $P3$, 通过最小化小区内所有用户设备的边缘计算时延 $\sum_{n\in N_t^m} T_{n,t}^{m,e}$ 为所有选择边缘计算的用户设备统筹制定最优的边缘计算资源分配决策 $f_{n,t}^m$。

将 $f_{n,t}^m$ 表示为 $f_{n,t}^m=y_{n,t}^m F_t^m$, 其中 $y_{n,t}^m \in [0,1]$ 表示边缘资源分配比例,因此 $P3$ 可以重新表述为:

$$\begin{aligned} &\min_{y_{n,t}^m} \sum_{n\in N_t^m} T_{n,t}^{m,e} \\ &\text{s.t. } C1: \sum_{n\in N_t^m} y_{n,t}^m \leqslant 1 \\ &C2: y_{n,t}^m \geqslant 0 \end{aligned} \tag{26}$$

其中, $T_{n,t}^{m,e}=\dfrac{x_{n,t}^m d_{k_{n,t}^m}}{y_{n,t}^m F_t^m}$。

因为在任意时刻,各个小区边缘计算资源分配是独立的,为了表达的简洁, 本部分剩余内容省略部分变量的上标 m 和下标 t。 将上式的优化目标函数等价表示为:

$$f(y_1,y_2,\cdots,y_{N^m}) = \frac{x_1 d_{k_1}}{y_1 F} + \frac{x_2 d_{k_2}}{y_2 F} + \ldots + \frac{x_{N^m} d_{k_{N^m}}}{y_{N^m} F} \tag{27}$$

容易证明 $\dfrac{\partial^2 f}{\partial y_i^2} \geqslant 0, i=1,2,\cdots,N^m$,所以式(27)是一个凸函数,$P3$ 是一个凸优化问题, 因此 KKT 条件是该优化问题的充要条件。其拉格朗日函数为:

$$L(Y,\lambda,\zeta) = \sum_{n\in N^m} T_n^e + \lambda\left(\sum_{n\in N^m} y_n - 1\right) - \sum_{n\in N^m} \zeta_n y_n \tag{28}$$

其中, λ 和 ζ 都是拉格朗日乘子。其 KKT 条件为:

(1)稳定性条件: $\dfrac{\partial L(Y,\lambda,\zeta)}{\partial y_n} = -\dfrac{x_n d_{k_n}}{y_n^2 F} + \lambda - \zeta_n = 0$

(2)原始可行性条件: $\sum_{n\in N^m} y_n - 1 \leqslant 0, y_n \geqslant 0$

(3)对偶可行性条件: $\lambda \geqslant 0, \zeta_n \geqslant 0$

(4)互补松弛条件: $\lambda\left(\sum_{n\in N^m} y_n - 1\right) = 0, \zeta_n y^n = 0$

通过求解,可得最优边缘资源分配比例 y_n^*:

$$y_n^* = \frac{\sqrt{x_n d_{k_n}}}{\sum_{i\in N^m} \sqrt{x_i d_{k_i}}} \tag{29}$$

将其代入到式(26)中即可得到小区内所有用户的最小边缘计算时延,进一步计算系统总开销和奖励值。

3.4 动态缓存决策求解

为了提高数据访问速度、减少传输延迟、降低能耗和网络带宽消耗,本节提出了基于任务请求概率的缓存决策策略,具体步骤包括:

(1)统计任务请求的概率:记录当前时刻每个任务被请求的次数,结合历史数据重新计算每个任

务的请求概率。

(2)选择缓存数据：根据任务请求概率，选择请求概率较高的数据存储在缓存中，这些数据通常是频繁访问的数据，例如热点数据、常用数据等。

(3)动态更新缓存空间：当缓存空间满时，根据任务请求概率，淘汰请求概率较低的数据，释放缓存空间，为请求概率较高的数据腾出空间，保持缓存空间的有效利用。

得到缓存决策 $c_{k,t}$ 后，将其作为 D3QN 的下一状态空间 $\boldsymbol{s}_{t+1}$ 的一项，并存入经验池中，用于 D3QN 网络参数的更新，影响卸载决策和资源分配决策的制定。

综上所述，本文所设计的一种计算卸载、资源分配和缓存决策联合优化算法(D3QN with OP, GA and KKT, QOGKNet)的完整流程描述如下。

算法 1　基于 QOGKNet 的计算卸载、资源分配和缓存更新算法

1. 初始化评估网络以及目标网络
2. 初始化经验回放缓冲区
3. 设置超参数，包括批次大小、学习率、折扣因子、ε - greedy 策略等
4. for *episode* = 1 to *max_episode* do
5. 初始化状态 s，*score* = 0，系统长期平均开销 $C = 0$
6. for $t = 1$ to T do
7. 根据 ε - greedy 策略选择一个松弛卸载动作 $\boldsymbol{x}_t$
8. 根据式(23)和式(24)将 $\boldsymbol{x}_t$ 量化为 V 个二进制卸载动作 $\boldsymbol{x}_v$
9. 将所有卸载动作代入到环境，通过遗传算法求解 $f_{n,t}^{m,l}$，基于式(29)求解 $f_{n,t}^{m}$
10. 通过式(11)和式(19)计算每个动作对应的系统开销 C_v 和奖励 r_v
11. 取奖励最大的 r_v 作为 r_t，其对应的 $\boldsymbol{x}_v$ 作为 $\boldsymbol{x}_t$，C_v 作为系统即时开销 C_t
12. 系统基于任务请求概率制定缓存决策 $c_{k,t}$，更新缓存空间，更新 $\boldsymbol{s}_{t+1}$
13. 更新 $score = score + r_t$，$C = C + C_t$
14. 将 $(\boldsymbol{s}_t, \boldsymbol{x}_t, r_t, \boldsymbol{s}_{t+1})$ 存储到经验回放缓冲区中
15. 从经验回放缓冲区中随机采样一个批次的数据 $(\boldsymbol{s}_i, \boldsymbol{x}_i, r_i, \boldsymbol{s}_{i+1})$
16. 计算目标 Q 值 $y_i = r_i + \gamma(1 - done_i) Q(s_i', \arg\max_a Q(s_i', a; \theta); \theta^-)$
17. 使用批次数据，通过最小化式(22)更新评估网络的参数 θ
18. 更新目标网络的参数 $\theta^- \leftarrow \theta$
19. 将状态 $\boldsymbol{s}_t$ 更新为下一个状态 $\boldsymbol{s}_{t+1}$
20. end for
21. $C = C / T$
22. end for

4　仿真与分析

为了验证所提出的算法的有效性和性能，本节在多个场景下进行了仿真实验。本节将展示实验的设置和结果分析。

4.1　实验设置

本实验基于 Python 3.8 编程语言和 Tensorflow 2.4 + Keras 深度强化学习框架实现，使用 GeForce RTX 3060 GPU，使用 Pycharm 作为 IDE。

设置实验仿真关键参数：用户数 $N = 12$，MEC 服务器数 $M = 2$，时隙数 $T = 10$，任务数 $K = 50$，任务数据量 b_k 的范围是[0.3, 0.5] Mbit，计算任务所需的 CPU 循环数 d_k 的范围是[900, 1 100] MCycles，MEC 服务器缓存空间大小 $D = 3$ Mbit，信道带宽 $B = 1$ Mhz，信道增益 h_n^m 的范围是[0.1, 0.9]，噪声功率 σ^2 为 10^{-9} W，用户设备计算能力 $f_l^{\max}$ 的范围是[800, 900] MHz，发射功率 p_n^m 的范围是[8, 12] W，平均驻留时间 β 是服从均值为 0.4，方差为 1.2 的高斯分布；MEC 服务器可用计算资源 F^m 的范围是[2 800, 3 200] MHz，每单位数据量任务的迁移开销 γ 为 0.6。时延系数 $\mu_1 = 0.8$，能耗系数 $\mu_2 = 0.2$，迁移开销系数 $\mu_3 = 0.1$，可以根据任务的不同需求动态调整系数的大小。

设置智能体的网络结构和参数：评估网络和目标网络采用相同的网络结构，中间包含 2 个 dense 层，通过 *Relu* 激活函数激活，通过状态值和优势值计算 Q 值后，使用 *Sigmoid* 激活输出。2 个 dense 层的神经元个数设置为 256 和 128。设置回合数为 500，网络的学习率为 0.005，折扣因子为 0.9，贪心策略为 1.0，衰减率 0.001、最终 0.01，目标网络更新速度为 200，经验池大小为 2 000，抽取样本量为 64，奖励调整系数 α 为 5，量化动作数 $V = N$。

4.2　结果分析

首先，为了验证所提出的 QOGKNet 的训练效果，本节选取了 D3QN 和 DQN 两种基线算法进行对比，并根据其在一定数量的训练周期内的平均累积奖励对 3 种算法进行了评估。训练过程如图 2 和图

3 所示,其中图 2 是每个回合的奖励曲线,图 3 是最新 100 个回合的平均奖励曲线。

仿真结果表明,QOGKNet 相比 D3QN 和 DQN,能够获得更高的平均奖励值,并且收敛速度更快,奖励曲线波动更小,说明 QOGKNet 在解决边缘计算卸载和资源分配问题上具有更优越的性能和效率。这是因为 QOGKNet 综合运用了多种优化方法,包括 D3QN 优化离散卸载决策和 GA、KKT 优化连续的资源分配决策,适合优化混合决策变量。而 D3QN 和 DQN 更适合优化离散决策变量,难以优化连续决策变量,导致最后输出的奖励波动较大。

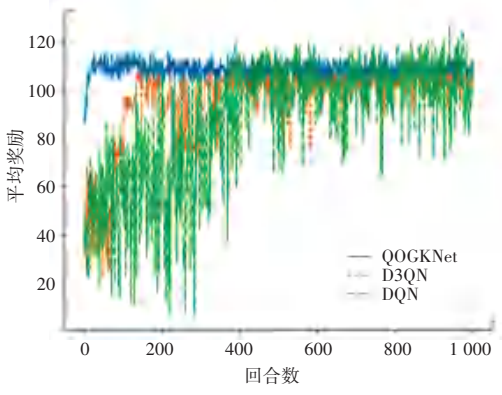

图 2 QOGKNet、D3QN 和 DQN 的回合奖励曲线

Fig. 2 Episode reward curves for QOGKNet, D3QN and DQN

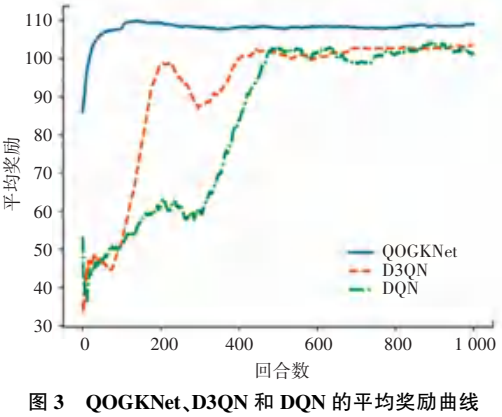

图 3 QOGKNet、D3QN 和 DQN 的平均奖励曲线

Fig. 3 Average reward curves for QOGKNet, D3QN and DQN

接着,在保持其他参数为 4.1 节中的默认参数的情况下,分别研究了不同用户数、不同边缘计算资源对于系统长期平均开销的影响。除了 D3QN 和 DQN 两种基线算法外,本节进一步增加 2 种情况作为比较:

(1)全本地(All-Local):所有移动用户设备以最大的计算能力自行计算请求的计算数据,而无需将任何任务数据卸载到 MEC 服务器。

(2)全边缘(All-MEC):所有移动用户设备请求的任务数据完全由 MEC 服务器进行任务的协助计算。

系统开销和用户数的关系如图 4 所示,随着系统内用户数的增加,所有情况下的系统长期平均开销都在增加。在该系统环境下,全边缘的决策只有在用户数较少时是近似最优的,但深度强化学习算法可以更优化地卸载任务,降低开销。QOGKNet 算法在不同用户数量下都具有明显优势,且优势随着用户数量的增加而扩大,而 D3QN 和 DQN 效果相近。

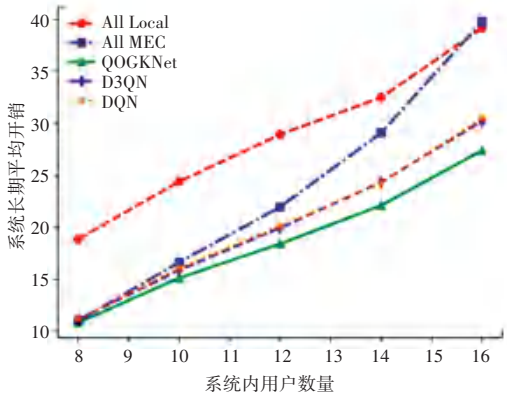

图 4 系统开销和用户数的关系

Fig. 4 Relationship between system cost and number of users

系统开销和 MEC 计算资源的关系如图 5 所示。由图 5 可知,随着 MEC 服务器计算资源的增加,除了全本地情况外,其余 4 种情况的系统长期平均开销不断减少。在 MEC 服务器计算资源小于 3 500 MHz 时,3 种深度强化学习算法都能学习到比全边缘更优的卸载决策,其中 QOGKNet 效果最好。随着 MEC 服务器计算资源的增加,D3QN 和 DQN 的效果受限,而 QOGKNet 通过多种优化方法表现最优。

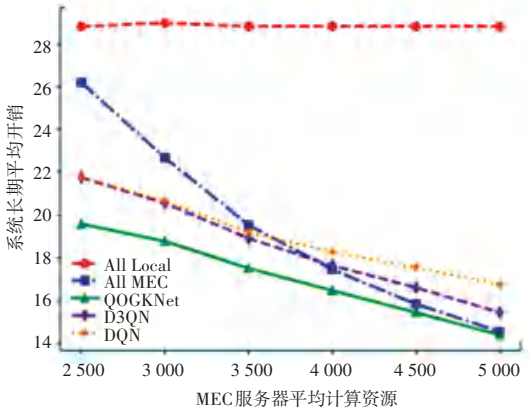

图 5 系统开销和 MEC 计算资源的关系

Fig. 5 Relationship between system cost and MEC computing resources

图 6 展示了全边缘、D3QN、DQN、QOGKNet 四种情况下缓存大小和系统长期平均开销的关系。结果表明,4 种情况下的系统长期平均开销随着缓存

空间的增加而降低，当缓存空间从 1 Mbit 增加到 2 Mbit 时，下降幅度明显。QOGKNet 方案的性能最好，全边缘方案最差，D3QN 和 DQN 居中，但 D3QN 略优。这是因为小缓存空间中任务的平均请求概率高，用户更容易命中具有较高请求概率的任务，从而减少上传任务数据的时延和能耗开销。

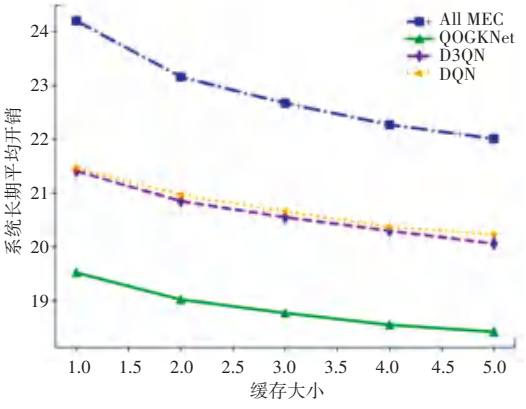

图 6　系统开销和缓存大小的关系

Fig. 6　Relationship between system cost and cache size for full offloading

5　结束语

本文研究了多 MEC 服务器多移动用户缓存辅助场景下的系统长期平均开销优化问题，并提出了一种计算卸载、资源分配和缓存决策联合优化算法。该算法通过 OP 输出量化 D3QN 输出动作，利用 GA 和 KKT 分别优化本地和边缘计算资源分配，基于任务请求概率分布更新 MEC 服务器缓存空间，并通过 D3QN 网络的学习优化得到近似最优的决策和系统长期平均开销。仿真结果表明，该算法具有更好的稳定性、更快的收敛速度，并且在不同用户数、MEC 服务器计算资源量情况下，在降低系统长期平均开销方面具有更好的效果。此外，基于任务请求概率的缓存机制的引入可以进一步降低系统开销，提高系统性能。下一步的研究将考虑在实际网络中，链路带宽和延迟可能受到网络拥塞、信号干扰、设备故障等多种因素的影响，并采取措施进一步完善。

参考文献

[1] 黄韬，刘江，汪硕，等. 未来网络技术与发展趋势综述[J]. 通信学报，2021，42(1)：130-150.

[2] 周悦芝，张迪. 近端云计算：后云计算时代的机遇与挑战[J]. 计算机学报，2019，42(4)：677-700.

[3] 李子姝，谢人超，孙礼，等. 移动边缘计算综述[J]. 电信科学，2018，34(1)：87-101.

[4] 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.

[5] MAO Yuyi，ZHANG Jun，LETAIEF K B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices[J]. IEEE Journal on Selected Areas in Communications，2016，34(12)：3590-3605.

[6] LIU Juan，MAO Yuyi，ZHANG Jun，et al. Delay-optimal computation task scheduling for mobile-edge computing systems[C]//2016 IEEE International Symposium on Information Theory (ISIT). Barcelona，Spain：IEEE，2016：1451-1455.

[7] LE H Q，AL-SHATRI H，KLEIN A. Efficient resource allocation in mobile-edge computation offloading：Completion time minimization[C]//2017 IEEE International Symposium on Information Theory (ISIT). Aachen，Germany：IEEE，2017：2513-2517.

[8] SALEEM U，LIU Yu，JANGSHER S，et al. Performance guaranteed partial offloading for mobile edge computing[C]//2018 IEEE Global Communications Conference (GLOBECOM). Abu Dhabi，United Arab Emirates：IEEE，2018：1-6.

[9] ZHANG Ke，MAO Yuming，LENG Supeng，et al. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks[J]. IEEE Access，2016，4：5896-5907.

[10] ZHAO Pengtao，TIAN Hui，QIN Cheng，et al. Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing[J]. IEEE Access，2017，5：11255-11268.

[11] GUO Fengxian，ZHANG Heli，JI Hong，et al. An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing[J]. IEEE/ACM Transactions on Networking，2018，26(6)：2651-2664.

[12] 黄永明，郑冲，张征明，等. 大规模无线通信网络移动边缘计算和缓存研究[J]. 通信学报，2021，42(4)：44-61.

[13] HE Xiaoming，LU Haodong，DU Miao，et al. QoE-based task offloading with deep reinforcement learning in edge-enabled Internet of Vehicles[J]. IEEE Transactions on Intelligent Transportation Systems，2020，22(4)：2252-2261.

[14] LI Hui，SUN Chuan，LI Xinhua，et al. Mobility-aware content caching and user association for ultra-dense mobile edge computing networks[C]//GLOBECOM 2020-2020 IEEE Global Communications Conference. Taipei，China：IEEE，2020：1-6.

[15] NATH S，WU Jingxian. Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems[J]. Intelligent and Converged Networks，2020，1(2)：181-198.

[16] YANG S，LIU J，ZHANG F，et al. Caching-enabled computation offloading in multi-region MEC network via deep reinforcement learning[J]. IEEE Internet of Things Journal，2022，9(21)：21086-21098.

[17] HASSELT H V，GUEZ A，SILVER D. Deep reinforcement learning with double q-learning[J]. arXiv preprint arXiv：1509.06461v3，2015.

[18] WANG Ziyu，SCHAUL T，HESSEL M，et al. Dueling network architectures for deep reinforcement learning[C]//International Conference on Machine Learning. New York：ACM，2016：1995-2003.

[19] 葛继科，邱玉辉，吴春明，等. 遗传算法研究综述[J]. 计算机应用研究，2008，25(10)：2911-2916.

[20] 解可新，韩立兴，林友联. 最优化方法[M]. 天津：天津大学出版社，1997.