云边端场景下基于多智能体深度强化学习的边缘缓存策略

王海艳[1,2]，常博[1]，骆健[1,2]
（1. 南京邮电大学计算机学院，江苏 南京 210023；2. 大数据安全与智能处理省高校重点实验室，江苏 南京 210023）

摘　要：云边端场景下，边缘缓存技术旨在通过促进边缘节点间的协同内容分发，减轻回程链路的流量负载并提升服务质量。考虑内容流行度的动态变化，提出了一种基于时间卷积网络的内容请求状态预测（TCNCRSP）模型。在此基础上，以最大化累积奖励为目标，提出了一种基于多智能体深度强化学习算法的边缘缓存策略，通过在云端利用长短期记忆（LSTM）网络对各边缘节点的状态数据进行降维处理，生成低维全局状态，减少状态共享所需的通信成本。实验结果显示，所提方法显著提升了缓存命中率和服务质量，同时降低了系统开销。

关键词：云边端协同；边缘缓存；内容流行度预测；时间卷积网络；深度强化学习
中图分类号：TP393
文献标志码：A
DOI：10.11959/j.issn.1000–436x.2025108

Edge caching strategy based on multi-agent deep reinforcement learning in cloud-edge-end scenarios

WANG Haiyan[1,2], CHANG Bo[1], LUO Jian[1,2]
1. School of Computer Science, Nanjing University of Post and Telecommunications, Nanjing 210023, China
2. Key Laboratory of Big Data Security & Intelligent Processing, Nanjing 210023, China

Abstract: In cloud-edge-end scenarios, edge caching technology aims to promote collaborative content distribution among edge nodes, thereby alleviating the traffic load on backhaul links and enhancing service quality. Considering the dynamic changes in content popularity, a time convolution network based content request state prediction (TCNCRSP) model for predicting content popularity was proposed. On this basis, aiming to maximize cumulative rewards, a multi-agent deep reinforcement learning algorithm based on edge caching strategy was proposed. This strategy was implemented using long short-term memory (LSTM) network in the cloud to perform dimensionality reduction on the state data of each edge node, thereby generating low-dimensional global states. This approach was used to reduce the communication costs required for state sharing. The experimental results show that the proposed methods significantly improve the cache hit rate and service quality, while also reducing system overhead.

Keywords: cloud-edge-end collaboration, edge caching, content popularity prediction, temporal convolutional network, deep reinforcement learning

0　引言

随着互联网的快速发展，智能终端设备数量呈指数级增长[1]，传统集中式云计算难以满足这些设备对低时延、高效率和低能耗服务的需求[2]。为此，云边端协同网络架构应运而生[3-4]，通过在网络边缘部署服务器节点，降低数据传输时延，同时利用中央云的计算和存储资源，从而提高数据处理速度与效率[5]。在云边端协同框架中，边缘缓存技

收稿日期：2025–04–12；修回日期：2025–06–03
通信作者：王海艳，wanghy@njupt.edu.cn
基金项目：国家自然科学基金资助项目（No.62272243）
Foundation Item：The National Natural Science Foundation of China (No.62272243)

术是重要研究方向，由于边缘服务器缓存容量有限，只能存储一小部分内容[6]。因此，设计有效的边缘缓存策略，以充分利用有限的缓存资源并为用户提供高质量、低时延的服务，成为该领域的研究重点。

现有研究中，内容流行度预测是边缘缓存策略的重要组成部分[7-9]，它能够提高缓存命中率，优化资源利用，从而提升服务质量和响应速度。目前，大多数边缘缓存策略依赖固定规则来预测内容请求状态，如最近最少使用（LRU, least recently used）和最不经常使用（LFU, least frequency used）等[10]。然而，不同边缘服务器的内容请求状态会随时间动态变化，基于固定规则的方法难以适应这种时变性，因此其预测结果往往不够准确，导致边缘缓存策略的命中率较低[11-12]。此外，在缓存决策方面，随着多智能体深度强化学习（MADRL, multi-agent deep reinforcement learning）的快速发展，云边端场景下的协同缓存机制受到越来越多的研究关注[13]。在传统MADRL中，不同边缘服务器之间需要共享传输高维的状态数据用以决策最终的缓存内容[14]，不可避免地导致通信成本随用户和内容数量的增加显著上升的问题。综上，在云边端协同框架下，提升请求内容预测的准确性以及降低数据共享的通信开销，成为边缘缓存策略面临的主要挑战。

为应对上述挑战，本文提出了一种基于多智能体深度强化学习的边缘缓存策略。该策略通过促进缓存资源有限的不同边缘服务器之间的协作，以实现整体缓存性能的最大化。具体而言，首先提出了一种基于时间卷积网络的内容请求状态预测（TCNCRSP, temporal convolutional network based content request state prediction）模型，该模型能够有效适应内容请求状态的动态变化，提升预测准确性，并具有并行处理、高收敛速度、梯度稳定等优点。针对高维数据共享导致的通信开销过大的问题，本文提出了一种成本高效的多智能体演员评论家（CEMAAC, cost efficient multi agent actor critic）算法，该算法在传统多智能体演员评论家（MAAC, multi agent actor critic）算法的基础上引入了基于长短期记忆（LSTM, long short-term memory）网络的数据降维模块。该模块在云数据中心提取各边缘服务器的完整状态，生成低维全局状态，并将其共享至边缘服务器，从而显著降低系统的数据通信开销。本文的主要贡献如下。

1) 本文提出了TCNCRSP模型进行内容流行度预测，通过不断更新神经网络模型的参数以适应用户兴趣的动态变化，实现了请求内容的准确预测。

2)本文提出了CEMAAC算法。在该算法中，各个边缘服务器将本地完整状态上传到云端，并利用云数据中心的LSTM网络隐藏层提取数据特征，生成低维全局状态。边缘服务器基于该低维状态进行共享和决策，从而有效降低了数据通信开销。

3)基于Geolife数据集进行了一系列对比实验，将本文提出的边缘缓存策略与一系列基线方法进行了比较。实验结果表明，本文提出的边缘缓存策略通过促进边缘缓存的协作，有效地提高了缓存奖励和缓存命中率。

1 相关工作

边缘缓存策略最初来源于操作系统和存储系统中的页面替换算法，如先入先出（FIFO,first input first output）、LRU和LFU。然而，这些固定规则的方法很难适应用户的动态需求，导致在动态场景下的性能较低[15]。Zhong等[16]针对单小区场景设计了一种边缘缓存模型，在该模型中每个边缘节点负责一个小区域的服务覆盖。该文引入了一种基于沃尔珀廷格架构（Wolpertinger architecture）的优化算法，旨在提高缓存命中率。然而，缓存策略是固定的，它们难以根据实时的用户需求和网络负载变化灵活调整服务策略，可能导致某些区域出现资源利用率低下和请求响应时延的问题。

为了解决静态边缘节点带来的问题，Zhang等[17]提出利用无人机作为动态缓存载体，根据用户对内容的偏好部署无人机，并提出了基于交换匹配的无人机部署算法，共同优化无人机部署和缓存放置，从而最大化用户体验的质量。尽管无人机的应用提高了边缘缓存业务的灵活性，但并未解决缓存空间有限的问题。

因此，构建多边缘协作网络成为一种趋势，即通过连接多个边缘节点构建一个多边缘协作网络来实现存储资源共享。Moayeri等[18]设计了一种融合环境与用户上下文的智能缓存方法，通过实时感知用户行为特征和网络环境状态，动态优化内容缓存决策，突破传统缓存策略对静态流行度依赖的局

限。此外，多边缘协作网络架构也被应用于包含无人机的缓存模型中[19-20]。Zhang 等[21]研究了支持多无人机系统的计算服务缓存问题，目标是减少所有设备的最大任务完成时延，并提出了一种结合块坐标下降（BCD）和逐次凸逼近（SCA）技术的迭代算法，以求得近似最优解。

随着人工智能技术的进步，越来越多的研究者开始尝试使用神经网络建模复杂的动态合作边缘缓存系统，以便捕捉系统内部特征并学习最佳缓存策略。时间卷积网络和自回归神经网络等技术被用于设计合作缓存替换策略。Tian 等[22]提出的基于深度强化学习的缓存准入算法（DRL-CA）能够从大量请求中提取更广泛的属性集，从而增强缓存效率。仿真结果表明，这些方法可以有效地提升系统的整体效用。一些研究人员还引入了 MAAC 算法来解决边缘缓存问题[23]，该算法将深度神经网络与注意力机制相结合，通过迭代更新行动者和批评者网络，可以快速收敛到一个最优策略，其网络架构的先进性在解决集中式边缘缓存问题方面取得了卓越的成果。深度 Q 网络（DQN, deep Q-network）使用深度神经网络来逼近 Q 值函数，并且可以处理高维状态空间和复杂环境。DQN 算法的衍生算法在很多任务和领域都取得了显著的成果[24-25]。Liu 等[26]设计了一种基于边缘缓存服务的替换算法，有效地降低了任务处理的整体时延。Shen 等[27]通过图注意力卷积核有效聚合各智能体相邻节点的特征信息，从而在车载网络环境中优化缓存决策。Zhou 等[28]将算法引入多边缘系统中，将联合协作缓存和推荐问题公式化为多代理多臂老虎机（MAMAB）问题。他们设计了一种改进的多智能体深度 Q 网络算法（MADQN, multi-agent deep Q-network），以完全分布式的方式解决这个问题。然而，大多数研究工作都没有考虑不同边缘服务器之间数据交换开销的问题。

2　系统模型和问题构建

2.1　系统模型

在云边端架构中，边缘缓存系统通常由云数据中心（CDC, cloud data center）、边缘服务器（ES, edge server）和用户设备（UE, user equipment）组成，系统模型如图 1 所示。边缘服务器通过高速链路相互连接，并共享用户信息及缓存空间，以为其通信范围内的用户提供缓存服务。云数据中心与边缘服务器之间通常通过回程链路连接，距离较远，因此时延较高。用户设备通过无线链路接入本地边缘服务器，并向其发出请求以获取所需内容。根据缓存情况，本地服务器对用户请求的响应方式有以下 3 种。

1)如果本地 ES 已缓存用户所需内容，则直接返回请求内容给 UE。这种方式响应速度最快，时延最低。

2)如果本地 ES 没有缓存用户所需内容，则检查邻居 ES 中是否缓存该内容。如果有，则邻居节点将请求内容传送给本地 ES，再由本地 ES 返回给 UE。这种方式虽然增加了传输步骤，但仍能较快响应。

3)如果本地 ES 和邻居 ES 都没有缓存用户所需内容，本地 ES 将从 CDC 获取内容，并将其返回给 UE。这种方式响应时间最长，时延最高。

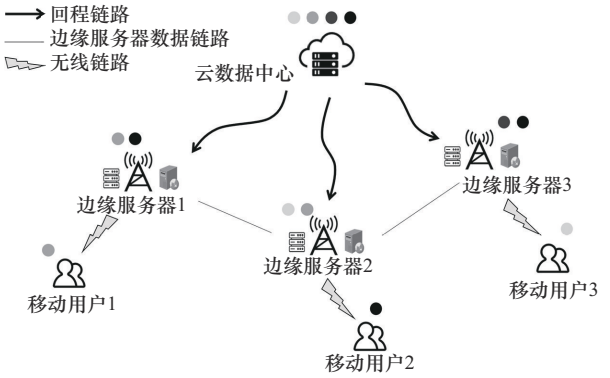


图 1　云边端场景边缘缓存系统模型

2.1.1　边缘服务器模型

边缘服务器集合记为 $\mathcal{B}=\{1,\cdots,b,\cdots,B\}$，所有边缘服务器 $b\in\mathcal{B}$ 具有相同的缓存容量 N。设 $\mathcal{F}=\{1,\cdots,f,\cdots,F\}$ 表示 UE 请求的内容种类集，n_f 表示请求内容 $f\in\mathcal{F}$ 所需的存储空间。事实上，单个 ES 有限的存储容量不足以容纳所有内容，因此有 $N\ll\sum_{f=1}^{F}n_f$。边缘服务器之间需要共享状态信息，才能最终决策缓存内容以及用户请求响应方式。共享的状态信息包括以下 3 个部分。

1)内容缓存状态

设 $c_{b,f}(t)\in\{0,1\}$ 表示边缘服务器 $b\in\mathcal{B}$ 在 t 时隙是否缓存了内容 $f\in\mathcal{F}$，定义边缘服务器 $b\in\mathcal{B}$ 在 t 时隙的缓存状态为

$$\boldsymbol{c}_b(t)=\{c_{b,1}(t),\cdots,c_{b,F}(t)\} \tag{1}$$

在边缘服务器b中缓存的内容必须满足约束

$$\sum_{f\in\mathcal{F}} c_{b,f}(t)\cdot n_f \leqslant N,\forall b\in\mathcal{B} \tag{2}$$

2)内容请求状态

设$\hat{p}_{b,f}(t)$表示在第t个时隙内边缘服务器$b\in\mathcal{B}$接收到内容$f\in\mathcal{F}$的请求数量。请求概率较高的内容通常在用户中更受欢迎，因而更有可能被缓存到边缘服务器。因此本文基于内容f的请求概率来近似表示其流行度。

$$p_{b,f}(t)=\frac{\hat{p}_{b,f}(t)}{\sum_{f\in\mathcal{F}}\hat{p}_{b,f}(t)} \tag{3}$$

边缘服务器$b\in\mathcal{B}$在t时隙的内容请求状态定义为

$$\boldsymbol{p}_b(t)=\{p_{b,1}(t),\cdots,p_{b,F}(t)\} \tag{4}$$

3)用户访问状态

由于用户移动的随机性，用户在不同边缘服务器之间的分布可能表现出地理上的差异。假设每个用户对特定内容的请求机会相同，但由于终端在服务覆盖范围内分布不均，内容请求可能会出现区域偏差。因此，为了描述非均匀用户分布对内容请求的影响，需要对用户访问过程进行建模。

发起内容请求的所有用户集合表示为$\mathcal{E}=\{1,\cdots,e,\cdots,E\}$。在时隙系统中，移动终端可以在一个时隙内跨越多个基站的覆盖区域。设$w_{b,e}(t)$表示用户$e\in\mathcal{E}$在第t个时隙内停留在边缘服务器$b\in\mathcal{B}$覆盖区域内的时间。访问边缘服务器b的终端总数表示为

$$m_b(t)=\sum_{e\in\mathcal{E}} I[w_{b,e}(t)] \tag{5}$$

其中，$I(\cdot)$表示一个指示函数，定义为

$$I(x)=\begin{cases}1, x>0\\ 0, \text{其他}\end{cases} \tag{6}$$

定义所有用户在边缘服务器$b\in\mathcal{B}$的平均停留时间为

$$w_b(t)=\frac{\sum_{e\in\mathcal{E}} w_{b,e}(t)}{m_b(t)} \tag{7}$$

将访问边缘服务器b的终端总数m_b和用户平均停留时间w_b归一化为[0,1]。

$$\bar{m}_b=\frac{m_b}{M},\bar{w}_b=\frac{w_b}{\Delta t} \tag{8}$$

综上，任意边缘服务器$b\in\mathcal{B}$在第t个时隙内的访问状态表示为

$$\boldsymbol{v}_b(t)=\{\bar{m}_b(t),\bar{w}_b(t)\} \tag{9}$$

2.1.2 POMDP模型

下一时刻的内容缓存状态、内容请求状态和用户访问状态仅与当前状态和缓存决策相关，而与历史状态无关，因此可以通过马尔可夫过程对服务器状态的演变进行建模。在时隙t开始时，ES已知其本地缓存状态$\boldsymbol{c}_b(t)$。然而，请求状态$\boldsymbol{p}_b(t)$和访问状态$\boldsymbol{v}_b(t)$作为统计量，分别反映了时间段[t,t+1]内的内容流行度和用户分布情况，并不能反映t时隙就被完全观察到。因此本文将缓存决策过程建模为部分可观察马尔可夫决策过程（POMDP）。其中状态、行动、观测和奖励定义如下。

1) 状态

定义边缘服务器$b\in\mathcal{B}$在时隙t的本地状态$\boldsymbol{s}_b(t)$为

$$\boldsymbol{s}_b(t)=\{\boldsymbol{c}_b(t),\boldsymbol{p}_b(t),\boldsymbol{v}_b(t)\} \tag{10}$$

整个系统的状态定义为

$$\boldsymbol{s}(t)=\{\boldsymbol{s}_1(t),\cdots,\boldsymbol{s}_b(t),\cdots,\boldsymbol{s}_B(t)\} \tag{11}$$

2) 行动

为了适应内容流行度和用户分布的动态变化，每个ES通过删除一些不流行的内容或添加其他流行的内容来主动调整其本地缓存。设$a_{b,f}(t)$表示边缘服务器$b\in\mathcal{B}$在t时隙决定是否缓存内容$f\in\mathcal{F}$，即$c_{b,f}(t+1)=a_{b,f}(t)$。

为了表示执行动作$a_{b,f}$后内容缓存状态的变化，将$a_{b,f}$与$c_{b,f}$进行比较。具体来说，若有$a_{b,f}>c_{b,f}$，则内容f被添加到边缘服务器b中；若$a_{b,f}<c_{b,f}$，则内容f将从边缘服务器b中移除；若两者相等，则边缘服务器b不会进行调整。在时隙t，边缘服务器$b\in\mathcal{B}$所采取的动作定义为

$$\boldsymbol{a}_b(t)=\{a_{b,1}(t),\cdots,a_{b,f}(t),\cdots,a_{b,F}(t)\} \tag{12}$$

整个系统的动作定义为

$$\boldsymbol{a}(t)=\{\boldsymbol{a}_1(t),\cdots,\boldsymbol{a}_b(t),\cdots,\boldsymbol{a}_B(t)\} \tag{13}$$

3) 观测

在时隙t，边缘服务器$b\in\mathcal{B}$无法直接获得当前的内容请求状态$\boldsymbol{p}_b(t)$和用户访问状态$\boldsymbol{v}_b(t)$。然而，通过计算时间间隔[t−1,t]内每个内容的请求数

量、访问UE的数量以及平均逗留时间，可以获得这些状态的先前信息$\boldsymbol{p}_b(t-1)$和$\boldsymbol{v}_b(t-1)$。因此，可以基于历史状态序列$\widetilde{\boldsymbol{p}_b}(t)=\{\boldsymbol{p}_b(t-1),\boldsymbol{p}_b(t-2),\cdots\}$和$\widetilde{\boldsymbol{v}_b}(t)=\{\boldsymbol{v}_b(t-1),\boldsymbol{v}_b(t-2),\cdots\}$预测当前的内容请求和用户访问状态。综上，$t$时隙的观测状态定义为

$$\boldsymbol{o}_b(t)=\{\boldsymbol{c}_b(t),\widetilde{\boldsymbol{p}_b}(t),\widetilde{\boldsymbol{v}_b}(t)\} \tag{14}$$

整个系统的观测状态定义为

$$\boldsymbol{o}(t)=\{\boldsymbol{o}_1(t),\cdots,\boldsymbol{o}_b(t),\cdots,\boldsymbol{o}_B(t)\} \tag{15}$$

4)奖励

在云边协作缓存系统中，用户请求的内容可以从本地ES、附近ES或云数据中心获取，每种方式分别对应以下3类内容交付成本。

①如果请求的内容已缓存在本地ES中，则可以以高带宽、低时延的方式直接将内容传输到终端。设α_b表示从本地服务器$b\in\mathcal{B}$获取内容的单位成本。假设ES b在时隙t内从其本地缓存中获取了存储大小为$r_b^b(t)$的内容，则本地ES b的服务成本为$\alpha_b r_b^b(t)$。

②如果请求的内容是从附近ES中获取的，数据将先转发给本地ES，再由本地ES转发给用户。设$\beta_b^k(k\in\mathcal{B},k\neq b)$表示从附近的ES k向访问ES b的UE传递数据的单位成本。分布式ES之间共享内容会消耗回程资源，因此从附近ES获取数据的成本要高于从本地ES获取数据的成本，即$\beta_b^k>\alpha_b$。假设ES b在时隙t中从ES k获取了大小为$r_b^k(t)$的内容，则附近ES的服务成本为$\sum_{k\in\mathcal{B},k\neq b}\beta_b^k r_b^k(t)$。

③如果请求的内容只能从云数据中心获取。设θ_b表示从云服务器向访问ES b的UE传送数据的单位成本，这里将0作为云数据中心服务器的索引。假设ES b在时隙t中从云数据中心获取了大小为$r_b^0(t)$的内容，则云数据中心的服务成本为$\theta_b r_b^0(t)$。

综上，时隙t的内容交付成本可以表示为

$$C_d=\alpha_b r_b^b(t)+\sum_{k\in\mathcal{B},k\neq b}\beta_b^k r_b^k(t)+\theta_b r_b^0(t) \tag{16}$$

在决定调整本地缓存之后，ES删除一些不太受欢迎的内容，并添加更受欢迎的内容。在时隙t中，在ES b中被替换的内容大小可以用$\sum_{f\in\mathcal{F}}[a_{b,f}(t)-c_{b,f}(t)]^+n_f$表示，其中定义$(x)^+\triangleq\max(x,0)$。设$\delta_b$表示替换ES b中内容的单位成本，则缓存替换的成本为

$$C_r=\sum_{f\in\mathcal{F}}\delta_b\,[a_{b,f}(t)-c_{b,f}(t)]^+n_f \tag{17}$$

系统总成本表示为内容交付成本C_{d}与缓存替换成本C_{r}之和。

$$C_{\mathrm{total}}=C_{\mathrm{d}}+C_{\mathrm{r}} \tag{18}$$

通过将内容缓存至更靠近用户的位置，边缘缓存可以显著降低内容交付成本。节省的成本越高，边缘缓存的效果就越显著。如果本地服务器b覆盖范围内的所有用户请求都从附近ES或云数据中心获取，则相应的成本，即非本地内容交付成本为$\sum_{k\in\mathcal{B}}\theta_b r_b^k(t)+\theta_b r_b^0(t)$。将边缘服务器$b$的即时奖励定义为通过非本地内容交付成本与系统总成本之差，即

$$R_b(t)=(\theta_b-\alpha_b)r_b^b(t)+\sum_{k\in\mathcal{B},k\neq b}(\theta_b-\beta_b^k)r_b^k(t)-\sum_{f\in\mathcal{F}}\delta_b\,(a_{b,f}(t)-c_{b,f}(t))^+n_f \tag{19}$$

其中，参数α_b、β_b、θ_b和δ_b是预定义的常数。最大化该奖励即最大化通过本地边缘缓存节省的内容交付成本。由于r_b^k的影响，ES b的即时奖励不仅取决于其自身的缓存状态，还取决于附近ES的缓存状态。整个系统的即时奖励定义为

$$R(t)=\sum_{b\in B}R_b(t) \tag{20}$$

2.2　问题表述

在分布式边缘缓存系统中，每个边缘服务器可以视为一个智能体，需要根据系统的状态来决定缓存策略。设$\pi=\{\pi_1,\cdots,\pi_b\cdots,\pi_B\}$表示所有智能体的缓存策略集合，其中$\pi_b$为边缘服务器$b$的策略，作用是将状态$s$映射到动作$a$，即$a=\pi(s)$。由于智能体的行为影响即时奖励和长期奖励，所有智能体需要协同工作以确定最优策略π^*，从而最大化长期奖励。假设长期奖励的折扣因子为$\gamma\in(0,1)$。协作缓存问题可以表述为一个多智能体决策问题，其目标是最大化累积折扣奖励。

$$\max_{\pi}V^{\pi}(s)=\mathbb{E}\Big[\sum_{t=0}^{\infty}\gamma^t R(t)\,\Big|\,s(0)=s,\pi\Big] \tag{21}$$

其中，$V^{\pi}(s)$为策略π下状态s的价值函数。给定系统的马尔可夫性质，最优策略π^*遵循贝尔曼（Bellman）方程

$$V^{\pi^*}(s)=R(s,\pi^*(s))+\gamma\sum_{s'\in S}P_{ss'}V^{\pi^*}(s') \tag{22}$$

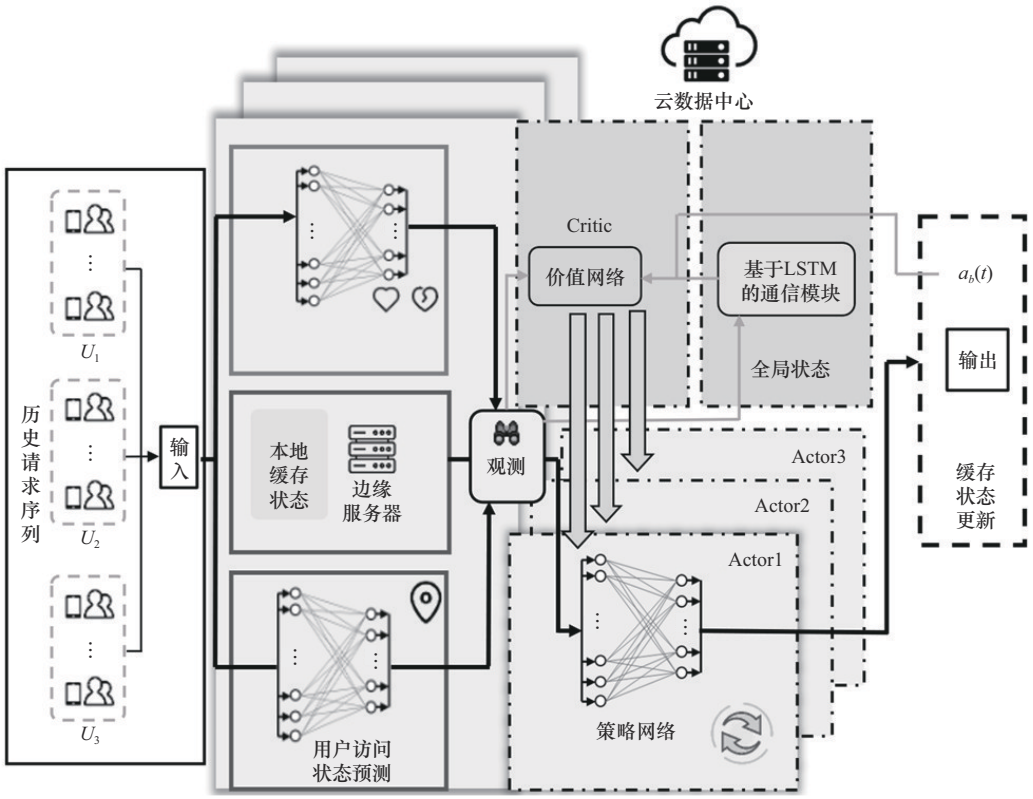


图2 基于CEMAAC算法的协同边缘缓存策略

其中，$R(s,\pi^*(s))$表示在状态s下采取最优策略π^*的即时奖励，$P_{ss'}$表示从状态s转移到状态s'的概率。

3 基于CEMAAC算法的协同边缘缓存策略

为有效解决协作缓存问题，本文提出了基于CEMAAC算法的协同边缘缓存策略，如图2所示。具体来说，该策略首先通过时间卷积网络（TCN）对内容请求状态和用户访问状态的时间序列进行预测，并将预测结果输入策略网络。策略网络利用CEMAAC算法，结合由云数据中心的LSTM网络层生成的本地状态和来自云端的低维全局状态，协调边缘节点资源与云服务资源，决定内容的缓存分配。

3.1 内容请求状态和用户访问状态预测

3.1.1 内容请求状态预测

由于每个边缘服务器的用户偏好差异，内容请求状态可能呈现出显著的时空变化。传统的固定规则方法难以适应这些动态需求，导致预测准确性较差。因此，需要采用深度学习模型进行请求状态预测。本文采用基于卷积神经网络的TCNCRSP模型来预测下一时刻的内容请求状态，预测结果作为策略网络的输入，辅助边缘服务器进行缓存决策。

TCNCRSP模型如图3所示，主要包括内容流行度预测器和请求状态整合器2个部分。

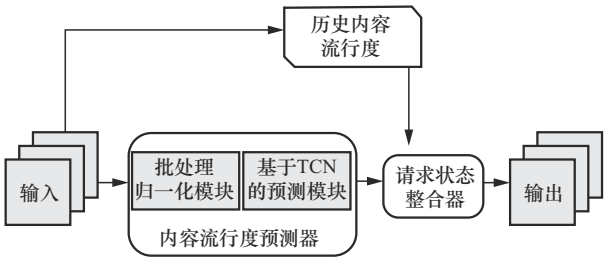


图3 TCNCRSP模型

1)内容流行度预测器

该模块以内容的历史请求状态序列作为输入，主要包括批处理归一化模块和基于TCN的预测模块。批处理归一化模块通过将输入进行归一化处理，以提升神经网络模型的稳定性。基于TCN的预测模块则处理归一化后的时间区间$(t-k,t)$内的内容请求状态特征向量$\{\boldsymbol{p}_b(t-1),\boldsymbol{p}_b(t-2),\cdots,\boldsymbol{p}_b(t-k)\}$，并输出对未来内容流行度的预测值$\{\boldsymbol{p}_b(t),\boldsymbol{p}_b(t+1),\cdots,\boldsymbol{p}_b(t+k)\}$。TCN采用1-D全卷积网络（FCN）结构，每个隐藏层与输入层保

持相同的长度，并由扩张因果卷积和残差块组成。

扩张因果卷积：传统的因果卷积在扩大接受野以构建长期记忆时，通常需要较多的层或较大的卷积核。为解决这一问题，模型中引入了扩张卷积。通过应用扩张因果卷积，模型能够获得更大的感受野。对于每一个卷积层 $\mathcal{F}=\{f_1,f_2,\cdots,f_k\}$，扩张因果卷积表示为

$$y_t=\sum_{k-1}^{K}f_k p_{t-(K-k)d} \tag{23}$$

其中，d为膨胀系数，K是卷积核的长度。

残差块：包含2个扩张因果卷积层。每个卷积层的权重经过归一化处理，并在每个扩张因果卷积层之后，TCN中会加入一个丢弃层（Dropout），以实现正则化。

TCN使用FCN实现密集预测，用于全面感知整个输入序列的信息。此外，TCN中的因果卷积确保了在时间序列预测中，未来的信息不会影响对过去的预测。

2)请求状态整合器

为了提高内容请求状态的预测性能，该模块平衡了短时突发记忆和长时记忆，并考虑了内容在不同时间点的优先级。具体而言，该模块综合了内容特征预测器生成的过去k个时间段的内容流行度和未来短期的流行度预测，并通过加权指数平均法得到最终的流行度预测数据。在下一个时隙$(T+1)$，内容f在服务器b区域内的请求状态$\tilde{p}_b(T+1)$为

$$\tilde{\boldsymbol{p}}_b(T+1)=(1-\lambda)\hat{\boldsymbol{p}}_{b,f,T+1}+\sum_{t=T-n+1}^{T}\lambda^{T-t+1}\boldsymbol{p}_{b,f,t} \tag{24}$$

其中，$\boldsymbol{p}_{b,f,t}$是内容f在T时的内容流行度，$\hat{\boldsymbol{p}}_{b,f,T+1}$是内容流行度预测器预测$T+1$时的内容流行度，$\lambda$为调整历史数据与最新数据比例的常数且$0<\lambda<1$，$n$为要考虑的历史数据长度。该方法结合了历史内容流行度信息，有效应对了用户动态性高带来的流行度变化，从而在短期突发记忆和长期记忆之间实现了平衡。

3.1.2　用户访问状态预测

由于用户在服务覆盖范围内的分布具有随机性，内容请求在不同区域可能呈现不均匀分布。为降低这种不均匀分布对缓存策略性能的影响，本文利用LSTM网络，根据用户的历史访问状态序列$\{\boldsymbol{v}_b(t-1),\boldsymbol{v}_b(t-2),\cdots\}$，模拟用户访问状态的动态变化，预测当前访问状态$\boldsymbol{v}_b(t)$。LSTM网络通过遗忘门、输入门和输出门这三大核心机制进行预测，具体机制如下。

1)遗忘门

遗忘门决定应丢弃哪些先前的输入状态，其激活向量定义为

$$f_t=\sigma\left[W_f\boldsymbol{v}_b(t-1)+U_f\boldsymbol{h}'(t-1)+b_f\right] \tag{25}$$

其中，W_f为输入与遗忘门之间的权值，U_f为前一个隐藏状态$\boldsymbol{h}'(t-1)$与遗忘门之间的权值，b_f为偏置项，$\sigma(\cdot)$为Sigmoid型函数

2)输入门

输入门选择性地记住当前输入状态，并更新记忆单元，计算式为

$$i_t=\sigma\left[W_i\boldsymbol{v}_b(t-1)+U_i\boldsymbol{h}'(t-1)+b_i\right] \tag{26}$$

$$\tilde{\boldsymbol{C}}_t=\tanh\left[W_c\boldsymbol{v}_b(t-1)+U_c\boldsymbol{h}'(t-1)+b_c\right] \tag{27}$$

其中，W_i和W_c为输入状态$\boldsymbol{v}_b(t-1)$与输入门的权值，U_i和U_c为前一个隐藏状态$\boldsymbol{h}'(t-1)$与输入门的权值，$\tanh(\cdot)$是双曲正切函数。状态向量更新为

$$\boldsymbol{C}_t=f_t\odot\boldsymbol{C}(t-1)+i_t\odot\tilde{\boldsymbol{C}}_t \tag{28}$$

3)输出门

输出门预测下一个访问状态。其隐藏状态的更新公式为

$$o_t=\sigma\left[W_o\boldsymbol{v}_b(t)+U_o|\boldsymbol{h}'|(t-1)+b_o\right] \tag{29}$$

$$\boldsymbol{h}'(t)=o_t\odot\tanh(\boldsymbol{C}_t) \tag{30}$$

其中，W_o为当前输入$v_b(t)$与输出门之间的权值，U_o为前一个隐藏状态$\boldsymbol{h}'(t-1)$与输出门之间的权值。当前用户访问状态的预测表示为

$$\tilde{\boldsymbol{v}}_b(t)=\sigma\left[\boldsymbol{W}_t\boldsymbol{h}'(t)\right] \tag{31}$$

其中，$\boldsymbol{W}_t$是输出门的权值向量。

该预测模型使用从边缘服务器收集的历史访问状态序列进行预训练。实际状态与预测状态之间的误差用于更新权重参数W和U，从而提高模型的预测精度和泛化能力。

3.2　基于数据降维的CEMAAC算法

传统非协作式边缘缓存策略基于每个边缘服务器的独立决策，导致流行内容在不同服务器上冗余缓存，从而降低了缓存资源利用效率。为解决这一问题，本文利用多智能体强化学习算法进行缓存决策。为了减少共享数据通信开销，本文提出了CEMAAC算法，该算法在传统MAAC算法的基础上

集成了基于LSTM模型的数据降维模块，各边缘服务器将本地状态上传至云端，通过LSTM网络提取的低维全局状态进行共享，从而避免了高昂的数据通信成本。

3.2.1 基于LSTM模型的数据降维模块

该模块使用LSTM网络提取所有智能体动作和观测值的时变特征到其隐藏状态，经过训练后的LSTM网络可以将原始高维输入序列转换为低维的隐藏状态表示。隐藏状态比原始输入具有更低的维度，并且包含了序列中最具代表性的特征。

基于LSTM的数据降维过程如下。

首先，在输入层接受预处理后的高维时序数据。在LSTM网络层，通过堆叠多个LSTM单元，逐步提取序列中的时序特征。

其次，对于每个时间步，LSTM的计算过程如下。

遗忘门为

$$f_t = \sigma(W_f[\boldsymbol{h}_{t-1}, x_t] + b_f) \tag{32}$$

其中，σ为Sigmoid函数，W_f和b_f为可学习参数，$\boldsymbol{h}_{t-1}$为前一时隙的隐藏状态，x_t为当前输入。

输入门为

$$\tilde{\boldsymbol{C}}_t = \tanh(W_C[\boldsymbol{h}_{t-1}, x_t] + b_C) \tag{33}$$

$$i_t = \sigma(W_i[\boldsymbol{h}_{t-1}, x_t] + b_i) \tag{34}$$

$$\boldsymbol{C}_t = f_t \odot \boldsymbol{C}_{t-1} + i_t \odot \tilde{\boldsymbol{C}}_t \tag{35}$$

其中，$\odot$表示逐元素乘法。

输出门为

$$o_t = \sigma(W_o[\boldsymbol{h}_{t-1}, x_t] + b_o) \tag{36}$$

$$\boldsymbol{h}_t = o_t \odot \tanh\left(\boldsymbol{C}_t\right) \tag{37}$$

再次，每个LSTM单元会输出当前时隙的隐藏状态$\boldsymbol{h}_t$和细胞状态$\boldsymbol{C}_t$。

最后，所有时间步的隐藏状态$\{\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_T\}$为数据的低维表示。

生成的低维全局状态可以通过LSTM网络的隐藏状态向量定义为

$$\boldsymbol{g}(t) = \varphi^{\text{com}}[\boldsymbol{o}(t), \boldsymbol{a}(t), \boldsymbol{g}(t-1); \phi^{\text{com}}] \tag{38}$$

其中，φ^{com}为LSTM网络的预测函数，ϕ^{com}为神经网络参数。LSTM网络通过不断收集最近的观测和动作数据，随着时间的推移更新低维全局状态。

引入数据降维模块使智能体之间共享全局状态$\boldsymbol{g}(t)$而非$\boldsymbol{o}(t)$和$\boldsymbol{a}(t)$本地完整状态，从而显著减少数据交换开销。设G为$\boldsymbol{g}(t)$的维度，假设每个维度的数据可以用一个浮点数表示，则将全局状态$\boldsymbol{g}$传输给所有B个智能体所需的开销是GB个浮点数。相比之下，如果没有数据降维模块，直接与所有B个智能体交换$\boldsymbol{a}(t)$和$\boldsymbol{o}(t)$的完整状态需要$B^2(4F+2)$个浮点数，因为$\boldsymbol{o}(t)$和$\boldsymbol{a}(t)$的维度分别是$B(F+2F+2)$和BF。由于LSTM将序列$\boldsymbol{o}(t)$和$\boldsymbol{a}(t)$的变化提取到$\boldsymbol{g}(t)$中，$\boldsymbol{g}(t)$的维度远小于$\boldsymbol{o}(t)$和$\boldsymbol{a}(t)$，即有$G \ll B(4F+2)$。因此，在智能体之间共享全局状态$\boldsymbol{g}(t)$而非$\boldsymbol{o}(t)$和$\boldsymbol{a}(t)$本地完整状态将显著减少数据交互。

在智能体之间直接传输高维数据会迅速消耗可用带宽，假设一个边缘节点需每秒传输1.08 Mbit/s数据，则单节点占用带宽为1.08 Mbit/s × 8 = 8.64 Mbit/s，则100个边缘节点的总带宽需求为864 Mbit/s，远超典型边缘网络的可用带宽（一般为100 Mbit/s），导致严重拥塞。通过引入基于LSTM的数据降维模块，可以有效缓解上述问题，实现通信开销的指数级降低，同时保留关键信息以支持智能体协作决策。

3.2.2 CEMAAC算法

给定全局状态$\boldsymbol{g}(t)$和局部观测值$\boldsymbol{o}(t)$，问题是找到使整体奖励最大化的最优缓存操作。为了解决这一问题，边缘服务器的缓存决策模块采用全连接神经网络来表示状态与动作的映射关系。

$$\hat{\boldsymbol{a}}_b(t) = \varphi_b^{\text{dec}}[\boldsymbol{c}_b(t), \tilde{\boldsymbol{p}}_b(t), \tilde{\boldsymbol{v}}_b(t), \boldsymbol{g}(t-1); \phi_b^{\text{dec}}] \tag{39}$$

其中，$\hat{\boldsymbol{a}}_b = \{\hat{\boldsymbol{a}}_{b,1}, \cdots, \hat{\boldsymbol{a}}_{b,F}\}$为输出的动作向量，$\phi_b^{\text{dec}}$为神经网络的参数。通常，在输出层中使用Softmax层将输出值归一化到[0, 1]的范围内，即$\hat{a}_{b,f} \in [0,1]$。

由于缓存决策模块的输出向量$\hat{\boldsymbol{a}}_b$是连续的，因此在执行阶段，每个智能体所执行的实际动作必须进行离散化

$$\boldsymbol{a}_b = \lfloor \hat{\boldsymbol{a}}_b N \rfloor = \{\lfloor \hat{\boldsymbol{a}}_{b,1} N \rfloor, \cdots, \lfloor \hat{\boldsymbol{a}}_{b,F} N \rfloor\} \tag{40}$$

其中，$\lfloor . \rfloor$是向下取整运算。因此，$a_{b,f} = 0$意味着ES b决定不在其本地缓存中保留内容f。

将本地缓存状态、预测模块生成状态与全局状态作为一个整体，Actor采用全连接神经网络来表示策略函数，该策略函数将当前局部观测$\boldsymbol{o}_b(t)$和之前的全局状态$\boldsymbol{g}(t-1)$映射到动作向量$\hat{\boldsymbol{a}}_b$，动作向量$\hat{\boldsymbol{a}}_b$可以表示为

$$\hat{\boldsymbol{a}}_b(t)=\varphi_b[\boldsymbol{o}_b(t),\boldsymbol{g}(t-1);\phi_b] \tag{41}$$

其中，ϕ_b为策略网络参数。

为了评估动作的效果，集中式评价模块采用全连接神经网络来表示动作价值函数，该函数为

$$Q[\boldsymbol{o}(t),\boldsymbol{g}(t-1),\boldsymbol{a}(t)]=\varphi^{\mathrm{cri}}[\boldsymbol{o}(t),\boldsymbol{g}(t-1),\hat{\boldsymbol{a}}(t)] \tag{42}$$

其中，φ^{cri}是评论家网络，在获取观测值$\boldsymbol{o}(t)$、行动$\boldsymbol{a}(t)$与全局状态$\boldsymbol{g}(t-1)$后给出长期奖励。

在实践中，尽管实际的系统参数可能与期望不同，但多智能体演员–评论家模型的参数可以进行实时更新，CEMAAC算法的执行流程如图4所示。

为了加快训练过程，通常在在线部署模型之前使用历史数据集对模型进行训练。在时隙t中，智能体将本地观察值$\boldsymbol{o}(t)$与采取的动作$\boldsymbol{a}(t)$一起发送给评论家模块。

在环境中执行动作后，即时奖励$R(t)$和随后的观察结果$\boldsymbol{o}(t+1)$被反馈给评论家。然后，通过最小化最小二乘时间差来更新评论家网络的参数。

$$L(\varphi^{\mathrm{cri}})=\mathbb{E}\{y(t)-\varphi^{\mathrm{cri}}[\boldsymbol{o}(t),\boldsymbol{g}(t-1),\hat{\boldsymbol{a}}(t)]\}^2 \tag{43}$$

其中，$y(t)=R(t)+\gamma\varphi^{\mathrm{cri}}[\boldsymbol{o}(t+1),\boldsymbol{g}(t),\hat{\boldsymbol{a}}(t+1)]$。缓存决策网络通过最大化的期望长期奖励来进行参数更新。

$$J(\phi_b)=\mathbb{E}\{\varphi^{\mathrm{cri}}[\boldsymbol{o}_b(t),\hat{\boldsymbol{a}}_b(t),\boldsymbol{g}(-1)]|_{\hat{\boldsymbol{a}}_b(t)=\varphi_b(\boldsymbol{o}_b(t),\boldsymbol{g}(t-1);\phi_b)}\} \tag{44}$$

当智能体b采取动作$\hat{\boldsymbol{a}}_b(t)$时。根据链式法则，缓存决策网络参数的梯度为

$$\nabla_{\phi_b}J(\phi_b)=\mathbb{E}\left\{\nabla_{\hat{\boldsymbol{a}}_b}\varphi^{\mathrm{cri}}[\boldsymbol{o}_b(t),\hat{\boldsymbol{a}}_b(t),\boldsymbol{g}(t-1)]\nabla_{\phi_b}\varphi_b[\boldsymbol{o}_b(t),\boldsymbol{g}(t-1);\phi_b]|_{\hat{\boldsymbol{a}}_b(t)=\varphi_b(\boldsymbol{o}_b(t),\boldsymbol{g}(t-1)}\right\} \tag{45}$$

缓存决策网络的参数更新为$\phi_b(t+1)=\phi_b(t)+\xi\nabla_{\phi_b}J(\phi_b)$，其中$\xi$是学习率。通过最小化损耗来更新通信模型的参数。

$$L(\phi^{\mathrm{com}})=\mathbb{E}\{[y(t)-\varphi^{\mathrm{cri}}(\boldsymbol{o}(t),\boldsymbol{g}(t-1))]^2|_{\boldsymbol{g}(t-1)=\varphi^{\mathrm{com}}[\boldsymbol{o}(t-1),\hat{\boldsymbol{a}}(t-1),\boldsymbol{g}(t-2)]}\}-\mathbb{E}\{\varphi^{\mathrm{cri}}[\boldsymbol{o}(t),\boldsymbol{g}(t-1)]|_{\boldsymbol{g}(t-1)=\varphi^{\mathrm{com}}[\boldsymbol{o}(t-1),\hat{\boldsymbol{a}}(t-1),\boldsymbol{g}(t-2)]}\} \tag{46}$$

综上所述，用于协作边缘缓存的CEMAAC算法的训练过程如算法1所示。首先，给定历史用户请求序列，可以训练内容请求的TCNCRSP模型和用户访问的LSTM模型。其次，生成E个样本轨道，每个轨道有T个样本。最后，基于经验池回放更新神经网络的参数ϕ^{decs}、ϕ^{cri}、ϕ^{com}。在线执行阶段，云数据中心负责维护通信模块网络和评论家网络，每个ES维护一个缓存决策网络，最终根据观察结果执行最优动作。

算法1　成本高效的多智能体演员评论家算法

输入　给定历史内容请求轨迹Tr和用户访问历史轨迹Ta

输出　根据$\hat{a}_b(t)$确定缓存文件集合$C(t)$

1)for(每个训练步骤)
2)　for(episode = 1 to E)
3)　　初始化$g(0)$，并令$t=1$
4)　　while ($t<T$)
5)　　根据 Tr 和 Ta 生成观测值；
6)　　对于每个智能体 b，Actor网络输出动作向量$\hat{\boldsymbol{a}}_b(t)=\varphi_b(\boldsymbol{o}_b(t),\boldsymbol{g}(t-1))$
7)　　动作探索：$\hat{\boldsymbol{a}}_b(t)=\hat{\boldsymbol{a}}_b(t)+\varepsilon$
8)　　获得调整后的动作$\hat{\boldsymbol{a}}_b(t)$
9)　　执行动作$\hat{\boldsymbol{a}}_b(t)$，并获得奖励$R_b(t)$和下一时刻的观测$\boldsymbol{o}_b(t+1)$

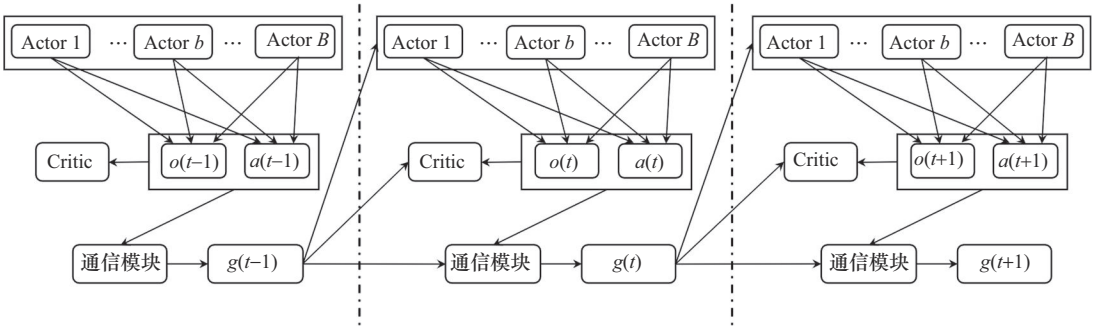


图4　CEMAAC算法的执行流程

10) 更新全局状态 $\boldsymbol{g}(t)$

11) $t=t+1$；

12) end while

13) 将该轮episode$\{\boldsymbol{g}(0),\boldsymbol{o}(1),\boldsymbol{a}(1),R(1)\}$；

14) end for

15) 从 D 中随机采样一个 minibatch 的 episodes M；

16) 对 M 中的每个 episode

17) 从 $t=T$ 倒序至 1

18) 通过最小化损失函数来更新 ϕ^{cri}；

19) 对所有智能体 $b\in B$

20) 通过最大化损失函数来更新 ϕ^{decs}

21) end for

22) 通过最小化损失函数来更新 ϕ^{com}

23) end for

24) 更新目标网络

$$\phi^{\text{tar}}=\tau\phi^{\text{cri}}+(1-\tau)\phi^{\text{tar}}$$

25) end for

26)end for

4 实验及结果分析

为了验证本文模型的可行性和有效性，本节将介绍仿真实验的数据集、实验环境与设置、基线方法与评价指标，并对实验结果进行对比分析。

4.1 仿真参数设置和数据集

为模拟UE在边缘服务器网络覆盖区域的随机分布，本文实验在Geolife数据集[29]上运行算法，该数据集记录了移动用户的真实运动轨迹。在模拟实验中，将地理区域划分为76个边长相同且互不重叠的六边形，每个六边形作为一个ES的覆盖区域，即 $B=76$。当用户的移动轨迹经过该ES的覆盖区域时，用户就与该特定ES相关联。由于用户的随机移动，各边缘服务器覆盖区域内的接入用户数量存在地理差异。

在真实的场景中，内容的受欢迎程度会随着时间的推移而变化。为了模拟特定内容的请求率随时间的变化，本文实验采用文献[30]中提出的散点噪声流量模型。具体来说，假定内容 f 的请求过程是一个时间非同构的泊松过程。瞬时请求率由 $V_f\lambda_f(t-\tau_f)$ 给出，其中 V_f 是内容 f 在生命周期内的平均请求率，λ_f 是幂律分布，τ_f 是内容 f 对用户可用的时间瞬间，τ_f 遵循泊松分布。这样就可以产生时变的内容流行度。

给定上述模型，本文可以通过更改模型参数为不同的内容生成一系列不同的请求序列。将生成的内容请求映射到Geolife数据集中的轨迹可以生成地理上分布式的请求。至此，实验中的内容请求具备了时空动态特性。

在实验中，每一个批次产生40 000个请求。共有100个内容可供用户使用，即 $F=100$。每个内容都由喷泉码（Raptor）编码成不同数量的片段，从100~1 000段不等。假设每个片段都有相同的大小。每个ES的缓存容量设置为100个段，即 $N=100$。从一个ES的本地缓存传送一个段的代价设为1，即 $\alpha_b=1,\forall b\in\mathcal{B}$。从ES k 传送一个文件段到ES b 的成本设为10，即 $\beta_b^k=10,\forall k,b\in\mathcal{B}$。从内容服务器获取一个文件片段的成本设为50，即 $\theta_b=50,\forall b\in\mathcal{B}$。替换一个片段的成本设为50，即 $\delta_b=50,\forall b\in\mathcal{B}$。

实际上，不同位置的内容可能导致不同的下载速度。从本地缓存中获取内容文件具有最高的速度，其下载速度为每秒0.6~1.0段。从附近的ES获取内容文件的速度为每秒0.3~0.5段，从内容服务器获取的速度最慢，为每秒0.1~0.3段。一个时间片的时长设置为10 min，训练集设置为200个时间片。Actor网络、Critic网络和通信网络的参数设置如表1所示。

表1 参数设置

模型	神经网络结构	参数
请求状态预测模型	先验:3个全连接层 生成:5个全连接层 递归:5个LSTM层 推断:6个全连接层	批量大小:8 序列长度:5 学习率:0.001
访问状态预测模型	6个LSTM层	权重衰减系数:0.000 1 经验重放缓冲区大小:50 000
协同缓存决策模型	6个全连接层 1个softmax层	折扣系数:0.9 初始探索系数:0.03 探索衰落系数:0.9
评论家	3个全连接层	目标网络更新率:0.01
通信模块	5个LSTM层	

训练过程和测试过程分别由2 000批和100批组成，使用相同的请求跟踪运行参考算法。

在模拟实验过程中，每次运行花费0.45 s，占用约2 GB的物理内存。训练一次CEMAAC算法模型以达到收敛的时间约为30 h。

4.2　基线方法

为了证明本文提出的基于CEMAAC算法的缓存策略有效性，对比分析了LRU、平均场边缘缓存（MF-ECS）、多智能体元强化学习（MAMRC）、多智能体深度确定性策略梯度（MADDPG）和联邦学习缓存（FedCache）这5种基线方法，并在不同的性能指标上比较了CEMAAC算法与这5种基线方法。

LRU：该算法将替换在最近的时间段内被请求的频率最低的内容[31]。每个ES独立运行一个LRU，用于自己的内容替换。实验中，假设LRU一次可以为单个内容替换10个片段。

MF-ECS：该算法[32]采用平均场强化学习（MFRL）解决边缘服务器的缓存决策问题，根据本地观察独立地做出自己的缓存决策。为了适应分布式边缘缓存场景，MF-ECS模型的状态、动作和奖励被定义为与上文介绍的Actor网络相同。

MAMRC：该算法由内外2个模型组成，内部模型采用MADRL算法来实现缓存，外部模型使用元学习方法来学习元参数并初始化内部模型[33]。与本文模型相比，该模型不包括用于数据降维的通信模块，因此Actor网络的输入中不包括全局状态g。

MADDPG：该算法[34]采用了集中训练分散执行（CTDE, centralized training and decentralized execution）的原则，每个智能体都有自己的策略网络和价值网络，允许智能体在训练过程中访问全局信息，而在实际执行时仅依赖于自身的局部观测。

FedCache：该算法[35]通过利用联邦学习将模型训练过程分发到用户设备上来减轻中央服务器的负担。每个用户使用自己的数据在本地训练模型，然后将训练好的参数上传到边缘服务器。

4.3　性能指标

本文定义了4个指标来评估算法性能。

1)缓存奖励

缓存奖励表示从边缘缓存中获得的总体长期奖励，它被定义为所有ES的即时奖励的总和。缓存奖励可以表示为相较于所有内容均从云数据中心获取的方式所节省的通信成本和内容传输时延，表示为$\tilde{R}=\sum_{t=1}^{T}R(t)$，其中，$R(t)$为即时奖励，$T$表示一个批次的持续时间。

2)缓存命中率

缓存命中率表示边缘缓存的利用率，定义为用户在一个集持续时间内请求的已缓存文件段所占的比例，表示为

$$\tilde{H}=\frac{1}{T}\sum_{t=1}^{T}\sum_{b=1}^{B}\frac{I_b(n,t)}{NB} \tag{47}$$

其中，$I(\cdot)$为量化指标，N为缓存容量，B为边缘服务器总数。如果缓存在ES b中的段n在时隙t期间被用户请求，则$I(\cdot)=1$，否则$I(\cdot)=0$。

3)流量负载

流量负载表示边缘服务器在特定时隙接收的请求数量。所请求的文件段可以从本地ES获取，也可以从附近ES获取，也可以从内容服务器获取，因此可以将其流量负载定义为每个请求段所容纳的比例。

$$\begin{aligned}\tilde{U}_{\text{local}}&=\frac{1}{T}\sum_{t=1}^{T}\frac{\sum_{b\in\mathcal{B}}r_b^b}{S(t)},\\ \tilde{U}_{\text{nearby}}&=\frac{1}{T}\sum_{t=1}^{T}\frac{\sum_{b\in\mathcal{B}}\sum_{k\in\mathcal{B},k\neq b}r_b^k}{S(t)}\\ \tilde{U}_{\text{server}}&=1-\tilde{U}_{\text{local}}-\tilde{U}_{\text{nearby}}\end{aligned} \tag{48}$$

其中，$S(t)$为时隙t期间用户请求的段总数，r_b^k表示从时间段k到时间段b获取到的文件段。

4)平均通信量

平均通信量定义为每轮迭代过程中所有智能体之间传输的数据总量除以智能体数量。

$$\tilde{C}=\frac{\sum_{i=1}^{B}\sum_{j=1,j\neq i}^{B}\text{Data}_{ij}}{B} \tag{49}$$

其中，Data_{ij}表示边缘节点i向边缘节点j发送的数据量，单位为字节。

4.4　实验结果分析

图5为不同算法的缓存奖励对比。由图5可知，CEMAAC获得最高的奖励，其次是FedCache、MADDPG、MAMRC和MF-ECS，最后是LRU。这表明CEMAAC可以很容易地协调ES以充分利用分布式边缘缓存，从而为所有ES获得更高的奖励。与CEMAAC相比，MAMRC也采用了多智能体Actor-Critic框架，但智能体之间没有进行数据通信。具体来说，对于MAMRC，每个ES完全基于本地观察做出缓存决策，而不考虑附近ES的动作和状态。然而，每个ES的缓存奖励，不仅取决于它自己的缓存状态，还取决于其他ES的缓存状态。此外，

MAMRC使用一个简单的全连接神经网络作为Actor，它不能明确地模拟观察状态的可变性。因此，MAMRC的Actor网络无法获得最优策略函数。

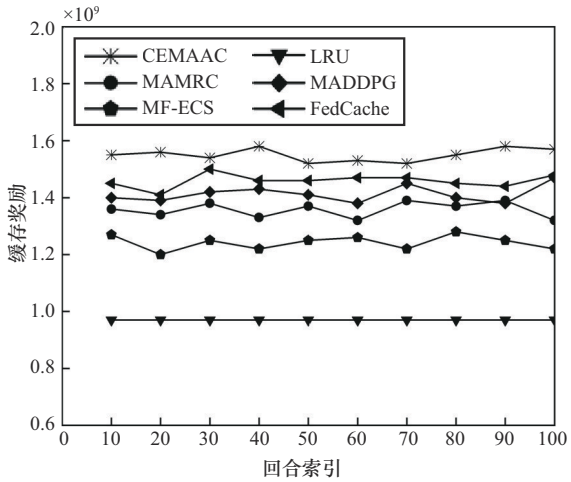


图5　不同算法的缓存奖励对比

图6为不同算法的缓存命中率对比。由图6可知，CEMAAC实现了最高的缓存命中率。尽管单个ES的存储空间有限，但CEMAAC可以通过在ES之间共享缓存的内容来使ES协同工作，这样用户就可以从本地和附近ES而不是从云数据中心获取内容。ES之间的协作避免了相同内容在不同ES中冗余缓存，节省了存储空间。相比之下，MF-ECS和LRU独立做出缓存决策，可能会导致不同边缘服务器缓存重复的内容。因此，CEMAAC具有比MF-ECS和LRU高得多的缓存命中率。

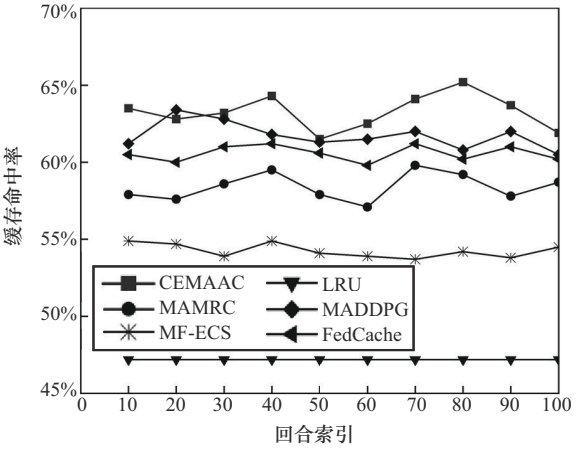


图6　不同算法的缓存命中率对比

由于深度学习模型比启发式模型更能表征内容流行度的变化，MF-ECS比LRU具有更高的缓存命中率。同时，在图6中，MAMRC实现的缓存命中率略低于CEMAAC。对于MAMRC，每个ES的决策不考虑其他ES的缓存状态。因此，使用MAMRC也会导致冗余缓存，浪费有限的存储空间。而MADDPG考虑了智能体之间的交互，一个边缘节点在进行缓存决策时会考虑周围边缘节点的缓存状态，因此其缓存命中率略高于MAMRC。

为表示流量负载情况，图7分别为云数据中心、本地ES、附近ES的流量负载。从图7(a)中可以看出，使用CEMAAC从云数据中心服务器传输的请求段仅占约21%，远低于使用其他算法的56%、43%、34%和25%的比例。这一结果表明，CEMAAC可以最大限度地利用边缘缓存来减少通过核心网络的内容检索，这与图6的结果一致，因为从云数据中心服务器获取内容的成本高于从边缘缓存获取内容的成本。

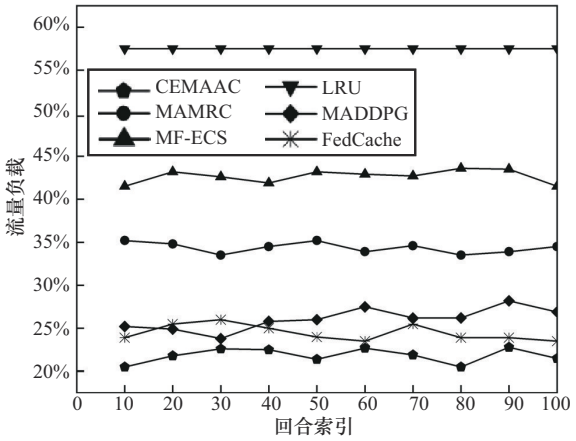
对比图7(b)和图7(c)可以看出，尽管CEMAAC（43%）从本地ES提取的内容片段比例略低于MF-ECS（45%），但使用CEMAAC（36%）从附近ES提取的片段比例远高于使用MF-ECS（12%）。在MF-ECS中，每个ES在决定缓存位置时只考虑容纳来自其本地用户的请求。相比之下，CEMAAC还考虑了ES之间的合作。同时，相较于MADDPG，CEMAAC在本地ES获取内容的比例较高。

图8为不同缓存大小下各算法的性能对比，其中，缓存大小范围为50~300段，步长为50段。从图8(a)可以看出，随着ES缓存容量的增加，缓存奖励也随之上升。更大的缓存容量使得边缘服务器能够存储更多的内容，从而提高了资源利用效率和用户请求满足率。

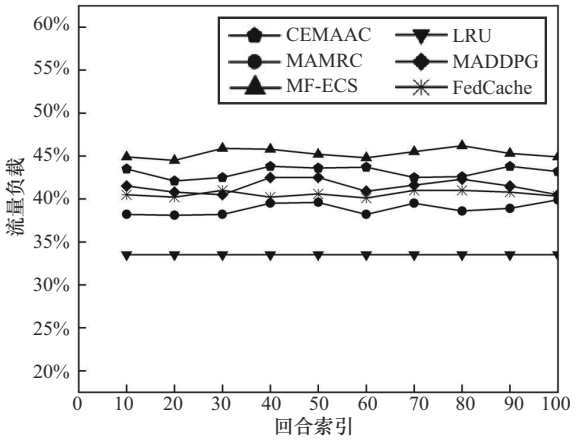
从图8(b)可以看出，随着ES缓存大小的增加，缓存命中率反而呈现下降趋势。较小的缓存容量限制了只能存储最受欢迎的内容，这意味着缓存中的文件段更有可能被用户请求到，从而保持较高的缓存命中率。相反，较大的缓存容量虽然能存储更多内容，但由于包含了一些不太受欢迎的内容，整体缓存命中率有所下降。

此外，随着缓存大小的增加，其他算法与CEMAAC之间的性能差距变得越来越明显。具体来说，CEMAAC在较大缓存容量的情况下依然能够

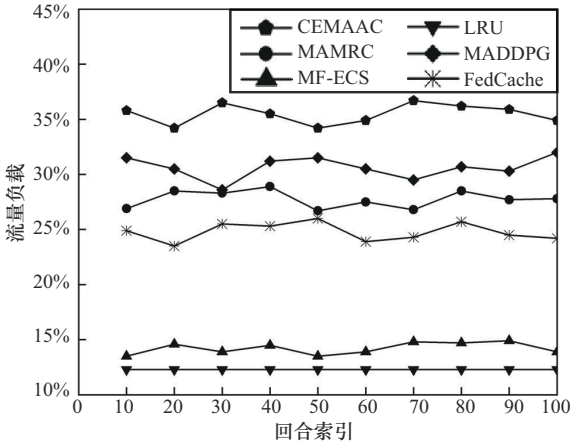
维持较高的缓存命中率和奖励值，而其他算法如MF-ECS和LRU则显示出明显的性能劣势。这表明CEMAAC不仅在小容量缓存场景下表现出色，在大容量缓存情况下同样具备显著优势，能够更有效地利用缓存空间，减少冗余内容，并提高整体系统性能。因此，无论缓存大小如何变化，CEMAAC都能提供更为稳定和高效的缓存策略。



(a) 云数据中心流量负载

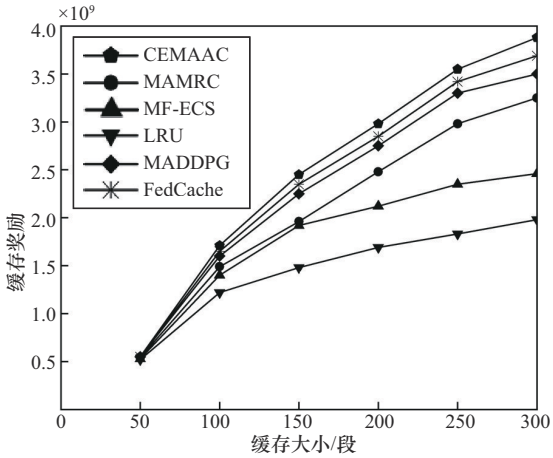


(b) 本地ES流量负载

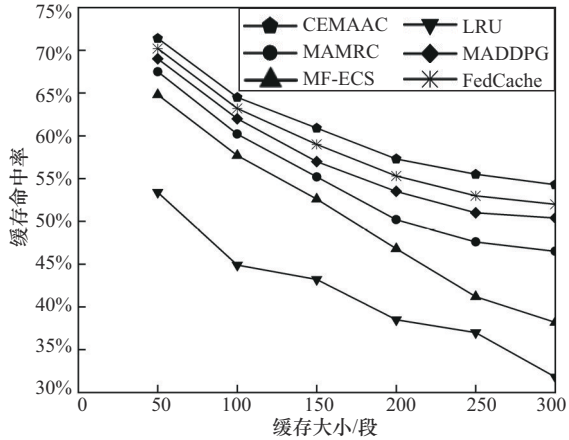


(c) 附近ES流量负载

图7　云数据中心、本地ES、附近ES的流量负载



(a) 缓存大小对缓存奖励的影响



(b) 缓存大小对缓存命中率的影响

图8　不同缓存大小下各算法的性能对比

图9为不同算法平均通信量对比。得益于数据降维模块，CEMAAC在每轮迭代过程中的平均通信量为50~70 MB，远低于MADDPG（160~180 MB）、MAMRC（120~140 MB）和FedCache（70~90 MB）的通信量，展示了CEMAAC的有效性。

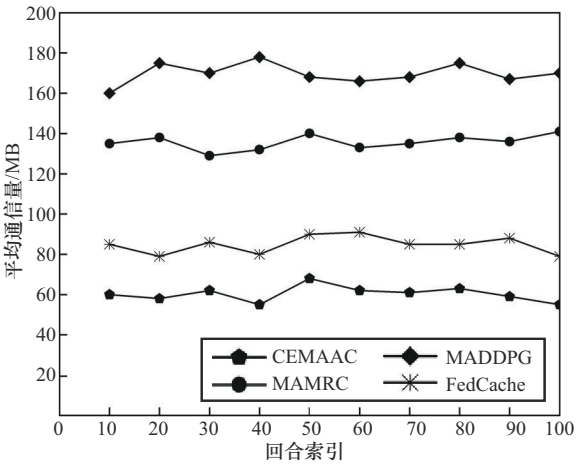


图9　不同算法平均通信量对比

此外，为验证引入LSTM数据降维模块对算法性能的影响，本文直接使用原始高维状态的变体模型、简单全连接层替代LSTM（FC Replacement）的变体模型与CEMAAC在缓存奖励与缓存命中率方面进行了对比，实验结果如图10和图11所示。

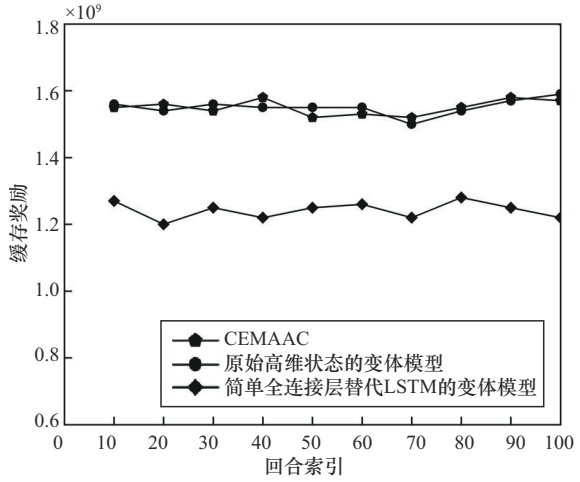


图10　引入LSTM数据降维模块对缓存奖励的影响

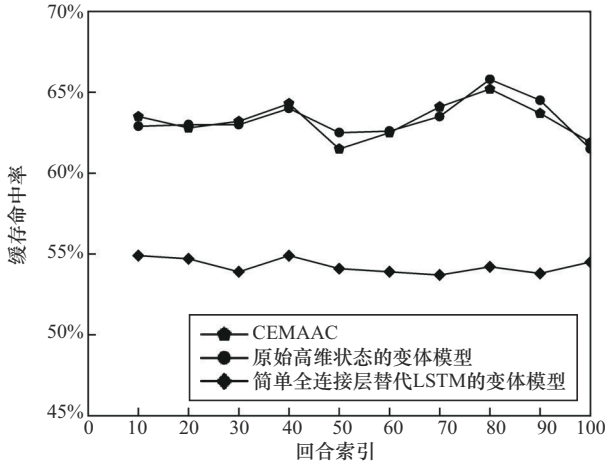


图11　引入LSTM数据降维模块对缓存命中率的影响

由于LSTM能够有效捕捉时序特征，使全局状态更加精确，因此本文算法与直接使用原始高维状态的变体模型在缓存奖励与缓存命中率方面性能相近，但大量节省了通信开销。使用简单全连接层替代LSTM的变体模型在处理非线性时序数据时效果有限，因此该算法性能最低。

5　结束语

本文针对云边端协同场景下边缘缓存面临的动态内容流行度预测与高维状态共享通信开销问题，提出了基于多智能体深度强化学习的边缘缓存策略。首先，通过时间卷积网络建模内容流行度的动态变化，提升了内容请求状态的预测精度；其次，引入基于长短期记忆网络的数据降维模块，在云端生成低维全局状态，显著降低了边缘节点之间的通信开销；最后，结合CEMAAC算法进行多智能体间协同决策，实现内容的缓存分配。实验结果表明，本文算法在缓存命中率、系统开销等方面与现有算法相比具有明显优势。在未来的研究中，可引入用户行为特征、实时网络拓扑等多模态数据，通过融合图神经网络等模型，构建更加精准的动态环境感知框架，进一步提升缓存系统性能。

参考文献：

[1] ABOLHASSANI B, TADROUS J, ERYILMAZ A. Optimal load-splitting and distributed-caching for dynamic content over the wireless edge[J]. IEEE/ACM Transactions on Networking, 2023, 31(5): 2178-2190.

[2] CISCO U. Cisco annual Internet report (2018–2023) [R]. 2018.

[3] SARAH A, NENCIONI G, KHAN M M I. Resource allocation in multi-access edge computing for 5G-and-beyond networks[J]. Computer Networks, 2023, 227: 109720.

[4] QIU L, CAO G H. Popularity-aware caching increases the capacity of wireless networks[C]//Proceedings of the IEEE INFOCOM 2017 - IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2017: 1-9.

[5] REISS-MIRZAEI M, GHOBAEI-ARANI M, ESMAEILI L. A review on the edge caching mechanisms in the mobile edge computing: a social-aware perspective[J]. Internet of Things, 2023, 22: 100690.

[6] CHEN X D, TAN T X, CAO G H, et al. Context-aware and energy-aware video streaming on smartphones[J]. IEEE Transactions on Mobile Computing, 2022, 21(3): 862-877.

[7] TRAN T X, HAJISAMI A, PANDEY P, et al. Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges[J]. IEEE Communications Magazine, 2017, 55(4): 54-61.

[8] GREGORI M, GÓMEZ-VILARDEBÓ J, MATAMOROS J, et al. Wireless content caching for small cell and D2D networks[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(5): 1222-1234.

[9] JIANG W, FENG G, QIN S. Optimal cooperative content caching and delivery policy for heterogeneous cellular networks[J]. IEEE Transactions on Mobile Computing, 2017, 16(5): 1382-1393.

[10] KWAK J, KIM Y, LE L B, et al. Hybrid content caching in 5G wireless networks: cloud versus edge caching[J]. IEEE Transactions on Wireless Communications, 2018, 17(5): 3030-3045.

[11] QIAN Y C, WANG R, WU J, et al. Reinforcement learning-based optimal computing and caching in mobile edge network[J]. IEEE Journal on Selected Areas in Communications, 2020, 38(10): 2343-2355.

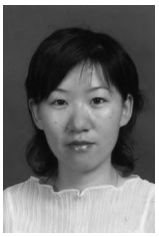
[12] LI Q, SUN Y H, WANG Q W, et al. A green DDPG reinforcement learning-based framework for content caching[C]//Proceedings of the 2020 12th International Conference on Communication Software and

Networks (ICCSN). Piscataway: IEEE Press, 2020: 223-227.
[13] WU P Y, LI J, SHI L, et al. Dynamic content update for wireless edge caching via deep reinforcement learning[J]. IEEE Communications Letters, 2019, 23(10): 1773-1777.
[14] ZHEN Y, CHEN W B, ZHENG L B, et al. Multiagent cooperative caching policy in industrial Internet of Things[J]. IEEE Internet of Things Journal, 2022, 9(18): 16770-16779.
[15] HASSLINGER G, NTOUGIAS K, HASSLINGER F, et al. Scope and accuracy of analytic and approximate results for FIFO, clock-based and LRU caching performance[J]. Future Internet, 2023, 15(3): 91.
[16] ZHONG C, GURSOY M C, VELIPASALAR S. Deep reinforcement learning-based edge caching in wireless networks[J]. IEEE Transactions on Cognitive Communications and Networking, 2020, 6(1): 48-61.
[17] ZHANG T K, WANG Y, LIU Y W, et al. Cache-enabling UAV communications: network deployment and resource allocation[J]. IEEE Transactions on Wireless Communications, 2020, 19(11): 7470-7483.
[18] MOAYERI M M, MOMENI H. An intelligent caching approach in mobile edge computing environment[C]//Proceedings of the 2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP). Piscataway: IEEE Press, 2024: 1-5.
[19] YAO C, JIANG C K, LIU Z, et al. Optimal capacity allocation and caching strategy for multi-UAV collaborative edge caching[C]//Proceedings of the 2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM). Piscataway: IEEE Press, 2021: 905-910.
[20] LI L X, WANG M, XUE K Y, et al. Delay optimization in multi-UAV edge caching networks: a robust mean field game[J]. IEEE Transactions on Vehicular Technology, 2021, 70(1): 808-819.
[21] ZHANG T K, WANG Y, YI W Q, et al. Joint optimization of caching placement and trajectory for UAV-D2D networks[J]. IEEE Transactions on Communications, 2022, 70(8): 5514-5527.
[22] TIAN A, FENG B H, ZHOU H C, et al. Efficient federated DRL-based cooperative caching for mobile edge networks[J]. IEEE Transactions on Network and Service Management, 2023, 20(1): 246-260.
[23] YU D J, WU T, LIU C F, et al. Joint content caching and recommendation in opportunistic mobile networks through deep reinforcement learning and broad learning[J]. IEEE Transactions on Services Computing, 2023, 16(4): 2727-2741.
[24] LI C L, ZHANG Y, GAO X, et al. Energy-latency tradeoffs for edge caching and dynamic service migration based on DQN in mobile edge computing[J]. Journal of Parallel and Distributed Computing, 2022, 166: 15-31.
[25] CHEN S W, YAO Z, JIANG X F, et al. Multi-agent deep reinforcement learning-based cooperative edge caching for ultra-dense next-generation networks[J]. IEEE Transactions on Communications, 2021, 69(4): 2441-2456.
[26] LIU Y N, YANG C, CHEN X, et al. Joint hybrid caching and replacement scheme for UAV-assisted vehicular edge computing networks[J]. IEEE Transactions on Intelligent Vehicles, 2024, 9(1): 866-878.
[27] SHEN J J, LIN Y, ZHANG Y J, et al. Content caching-assisted vehicular edge computing using multi-agent graph attention reinforcement learning[J]. IEEE Transactions on Vehicular Technology, 2025, 74(2): 3509-3514.
[28] ZHOU X B, KE Z H, QIU T. Recommendation-driven multi-cell cooperative caching: a multi-agent reinforcement learning approach[J]. IEEE Transactions on Mobile Computing, 2024, 23(5): 4764-4776.
[29] ZHENG Y, XIE X, MA W Y. GeoLife: a collaborative social networking service among user, location and trajectory[J]. IEEE Data Engineering Bulletin, 2010, 33(2): 32-39.
[30] TRAVERSO S, AHMED M, GARETTO M, et al. Temporal locality in today's content caching[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(5): 5-12.
[31] PODLIPNIG S, BÖSZÖRMENYI L. A survey of Web cache replacement strategies[J]. ACM Computing Surveys, 2003, 35(4): 374-398.
[32] YANG Y J, LOU K H, WANG E, et al. Multi-agent reinforcement learning based file caching strategy in mobile edge computing[J]. IEEE/ACM Transactions on Networking, 2023, 31(6): 3159-3174.
[33] WEI Z C, ZHAO Y, LYU Z W, et al. Cooperative caching algorithm for mobile edge networks based on multi-agent meta reinforcement learning[J]. Computer Networks, 2024, 242: 110247.
[34] WU T, ZHOU P, WANG B H, et al. Joint traffic control and multi-channel reassignment for core backbone network in SDN-IoT: a multi-agent deep reinforcement learning approach[J]. IEEE Transactions on Network Science and Engineering, 2021, 8(1): 231-245.
[35] LEKHARU A, SAMANTA A, SUR A, et al. Content-aware caching at the mobile edge network using federated learning[J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2025, 9(2): 1166-1176.

[作者简介]

王海艳（1974– ），女，江苏东台人，博士，南京邮电大学教授、博士生导师，主要研究方向为服务计算、可信计算、大数据应用与云计算技术、隐私保护技术等。

常博（1999– ），男，河北张家口人，南京邮电大学硕士生，主要研究方向为边缘计算。



骆健（1976– ），女，江西赣州人，南京邮电大学副教授，主要研究方向为服务计算、可信计算、服务推荐等。