NDN 中边缘计算与缓存的联合优化

张宇[1,2]，程旻[1,2]
（1. 北京理工大学信息与电子学院，北京 100081；2. 上海机电工程研究所，上海 201109）

摘 要：命名数据网络（NDN）基于内容名称进行路由，且节点配备一定的缓存能力，故在架构上更易与边缘计算结合。首先，提出一个在 NDN 中实现网络、计算和缓存动态协调的综合框架。其次，针对不同区域内容流行度的差异性，提出基于矩阵分解的局部内容流行度预测算法；以最大化系统运营收益为目标，利用深度强化学习解决计算和缓存资源分配以及缓存放置策略的联合优化问题。最后，在 ndnSIM 中构建仿真环境，实验证明所提方案在提高缓存命中率、降低平均时延和远程服务器负载等方面具有明显优势。

关键词：命名数据网络；边缘计算；缓存策略；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.2022160

Joint optimization of edge computing and caching in NDN

ZHANG Yu[1,2], CHENG Min[1,2]
1. School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China
2. Shanghai Institute of Mechanical and Electrical Engineering, Shanghai 201109, China

Abstract: Named data networking (NDN) is architecturally easier to integrate with edge computing as its routing is based on content names and its nodes have caching capabilities. Firstly, an integrated framework was proposed for implementing dynamic coordination of networking, computing and caching in NDN. Then, considering the variability of content popularity in different regions, a matrix factorization-based algorithm was proposed to predict local content popularity, and deep reinforcement learning was used to solve the the problem of joint optimization for computing and caching resource allocation and cache placement policy with the goal of maximizing system operating profit. Finally, the simulation environment was built in ndnSIM. The simulation results show that the proposed scheme has significant advantages in improving cache hit rate, reducing the average delay and the load on the remote servers.

Keywords: named data networking, edge computing, cache policy, deep reinforcement learning

0 引言

随着互联网业务的蓬勃发展，网络的业务模式正在由传统的点对点数据传输模式演变为以信息共享为主的转发模式。通信方式中数据物理位置的重要性被逐渐淡化，用户关注的重心转向了数据内容本身。在此趋势下，美国国家科学基金会针对基于 TCP/IP 的传统网络架构提出的解决方案——命名数据网络（NDN, named data networking），其采用基于内容名称的路由和转发实现对数据的检索和获取[1]，成为当前的研究热点。与此同时，随着计算和存储逐渐成为网络的重要功能，计算、存储与网络基础设施的融合也正成为未来网络发展的重要趋势。边缘计算作为 5G 网络的关键技术之一，将计算和存储资源一同下沉到靠近终端用户的边缘节点，以缓解带宽压力、改善用户体验。NDN 的相关机制如基于内容名称进行路由、节点具备一定存储能力等，与边缘计算的设计方向不谋而合，恰

收稿日期：2022−03−31；**修回日期：**2022−07−01
基金项目：国家重点研发计划基金资助项目（No.2019YFB1803200）
Foundation Item: The National key Research and Development Program of china (No.2019YFB1803200)

好能够为构建网络、计算、存储一体化的未来网络提供重要的技术支撑，因此在 NDN 中设计与边缘计算相结合的综合框架并实现对计算和缓存资源的协同管理具有重要的研究意义与价值。

传统的资源分配和缓存策略根据特定的数学模型做出决策[2-7]，忽略了节点之间流量波动的相关性和不同区域用户偏好的差异性，使计算和缓存资源不能得以有效利用。此外，虽然全球每天产生约 80 EB 的数据量，但用户对其中绝大部分内容的评价是没有记录的，而长尾理论表明，在网络时代，冷门内容的运营收益未必会低于当前关注度高的内容。事实上，当前大多数缓存策略的设计都忽略了长尾内容的潜在收益。

现有的优化方法如凸优化和博弈论等虽然已被广泛应用于改进资源分配方案和缓存策略，但仍存在以下问题：1) 一些关键因素如无线信道条件、不同应用的具体要求和内容流行度等被提前设定，而在现实中，这些信息难以直接获得且会随时间变化；2) 除 Lyapunov 优化[8]外，目前大多数算法只对系统快照进行优化而没有考虑到当前决策对资源管理的长期影响，即系统的动态问题没有得到很好的解决。

机器学习作为一种新兴的数据分析及处理手段，可以从传统方法难以建模和分析的数据中得到隐含的趋势和关联，能够更好地在内容流行度未知且动态变化的网络中赋予计算和缓存更多的自主性和智能性，帮助其学习如何根据已有的经验进行协调优化，有望推动实现未来网络的内生智能。深度强化学习作为机器学习领域中最具潜力的研究方向之一，将深度学习的感知能力和强化学习的决策能力相结合，有助于解决实际场景中的复杂问题。深度 Q 网络（DQN, deep Q network）算法作为经典的深度强化学习算法，虽然解决了高维观察空间的问题，但其依赖于找到使价值函数最大的动作[9-10]，在连续域的情况下需要在每个步骤进行迭代优化，因此目前只能处理离散和低维的动作空间。此外，DQN 采取随机的动作策略，即每次进入相同状态的时候，输出的动作会服从一个概率分布，这导致智能体的行为具有较大的异变性，参数更新的方向很可能不是策略梯度的最优方向。

针对上述问题，本文提出在 NDN 边缘节点部署计算模块，使节点兼具缓存和计算能力，在网络边缘创建分布式的轻型数据处理中心[11]；利用深度确定性策略梯度（DDPG, deep deterministic policy gradient）算法对计算和缓存资源的管理进行联合优化，以实现网络、计算和缓存功能动态协调的综合框架。具体创新点如下。

1) 在 NDN 中设计与边缘计算相结合的综合框架。在 NDN 边缘节点部署计算模块和智能体，在将内容和资源向终端用户靠近的同时，实时感知网络状态，在与环境的交互中学习最优的计算和缓存的资源分配以及缓存放置策略。

2) 提出基于矩阵分解[12]的局部内容流行度预测算法。矩阵分解算法引入了隐向量的概念，将高维矩阵映射成 2 个低维矩阵的乘积，使之具有强大的处理稀疏矩阵的能力；通过补全用户对内容的评分矩阵，将与边缘节点直接连接的所有用户对某个内容的相对评分作为该内容的局部内容流行度。

3) 在平均时延的约束下，以系统运营收益最大化为目标，利用 DDPG 算法对计算和缓存资源分配以及缓存放置策略进行联合优化。DDPG 算法将动作策略的探索和更新分离，前者采用随机策略，后者采用确定性策略[13]；可以在高维的连续动作空间中学习策略，适用于边缘计算和缓存联合优化时的连续性控制问题。

4) 在 ndnSIM 中构建仿真环境，通过 DDPG 算法和经典的 DQN 算法在边缘计算和缓存的联合优化问题上的横向对比，证明本文方案在稳定性、收敛速度和性能表现上都有明显优势；与传统缓存放置策略相比，本文方案可以有效地提高缓存命中率、降低系统成本和平均时延，改善用户体验。

1　框架设计

NDN 中基于机器学习实现网络、计算和缓存动态协调的综合框架如图 1 所示，在边缘节点部署计算模块，结合 NDN 的网内缓存机制，将网络功能、内容和资源向终端用户靠近[14-15]；为了优化资源分配，处理具有多样性和时变性的复杂问题，在边缘节点部署智能体，通过状态、行动和奖励与环境互动。智能体首先实时地感知网络状态，例如，与节点直接连接的用户发布的计算任务和请求的内容、节点当前可用的计算资源和缓存容量、局部内容流行度等，继而根据当前状态自主地设计行动，包括计算和缓存的资源分配以及缓存放置策略，最后基于环境反馈的奖励来更新和改进其行动策略，形成感知−动作−学习的循环结构。

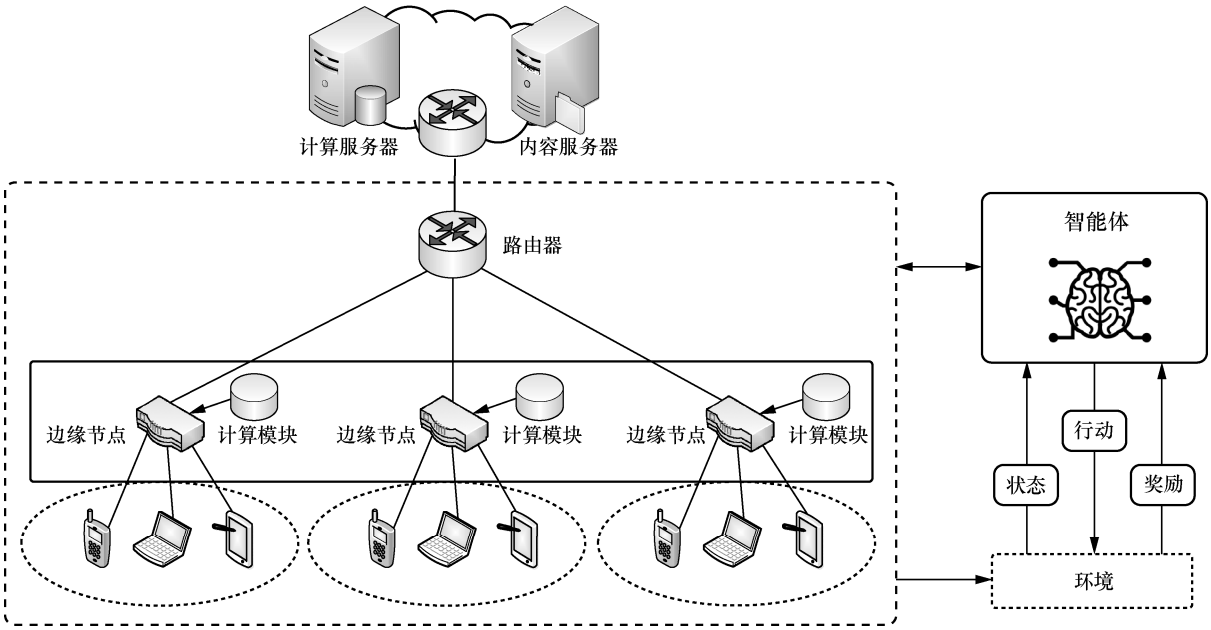


图 1　NDN 中基于机器学习实现网络、计算和缓存动态协调的综合框架

在此框架中，边缘节点为其覆盖区域内的用户提供通信、计算和缓存功能，同时考虑到节点计算和缓存能力的变化、不同区域用户偏好的差异性以及内容流行度的时变性，智能体自适应地调整行动策略，为不同区域的用户提供个性化的解决方案。相比边缘节点，远程服务器具有更丰富的计算和缓存资源。当边缘节点不足以支持用户的计算任务或没有缓存用户的请求内容时，计算任务或请求内容可以被卸载到远程服务器（即计算服务器和内容服务器）。因此，智能体在优化计算资源和缓存容量外，还需要学习在边缘节点缓存更受欢迎的内容和处理时延敏感的计算任务，以降低网络时延、改善用户体验。

2　具体方案

2.1　局部内容流行度预测

互联网时代信息量的成倍增长导致了信息过载问题，即不仅用户在海量数据面前束手无策，网络运营商也很难发现用户的兴趣点为其提供个性化的服务。推荐系统通过研究用户的历史行为、兴趣偏好或者不同区域的人口统计学特征，产生用户可能感兴趣的内容列表，精准高效地满足不同用户的信息需求。将每个节点对所有内容的评分抽象为一个 m 行（m 个与该节点直接连接用户）、n 列（n 个内容）的矩阵，然而由于大多数用户只对网络中极少部分的内容有过评价记录，这个矩阵是很稀疏的。基于矩阵分解的协同过滤算法引入了隐向量的概念，将高维矩阵映射成 2 个低维矩阵的乘积，加强了处理稀疏矩阵的能力，同时能挖掘更深层的用户与用户、用户与内容间的关系，较高的预测精度使之成为当前热门的推荐系统主流算法。矩阵分解对原稀疏矩阵进行填充，达到了通过分析已有数据来预测未知数据的目的，从而有助于边缘节点提前缓存当前冷门但未来很可能会流行的内容，挖掘长尾内容的潜在收益。

如图 2 所示，矩阵分解算法将 $m\times n$ 维的共现矩阵 $\boldsymbol{R}$ 分解为 $m\times k$ 维的用户矩阵 $\boldsymbol{U}$ 和 $k\times n$ 维的内容矩阵 $\boldsymbol{I}$ 相乘的形式，其中，m 为用户的数量，n 为内容的数量，k 为隐向量维度。k 的大小决定了隐

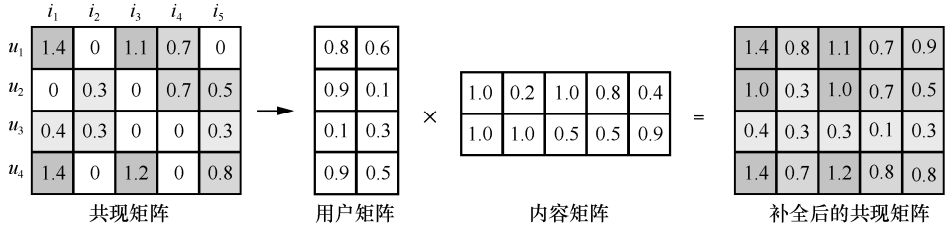


图 2　矩阵分解算法

向量表达能力的强弱，k 的取值越小，隐向量包含的信息就越少，但泛化能力较高；反之，k 取值越大，隐向量的表达能力就越强，但泛化能力相对降低。通常 k 在实验中折中取值，以保证推荐效果和空间开销的平衡。

用户 u 对内容 i 的预估评分 $\hat{r}_{ui}$ 为

$$\hat{r}_{ui} = \boldsymbol{q}_i^{\mathrm{T}} \boldsymbol{p}_u \tag{1}$$

其中，$\boldsymbol{p}_u$ 是用户 u 在用户矩阵 $\boldsymbol{U}$ 中对应的行向量，$\boldsymbol{q}_i$ 是内容 i 在内容矩阵 $\boldsymbol{I}$ 中对应的列向量。

采用梯度下降法优化。定义目标函数为

$$\min_{\boldsymbol{q}^*,\boldsymbol{p}^*} \sum_{(u,i)\in K} \left(r_{ui} - \boldsymbol{q}_i^{\mathrm{T}} \boldsymbol{p}_u\right)^2 + \lambda\left(\|\boldsymbol{q}_i\| + \|\boldsymbol{p}_u\|\right)^2 \tag{2}$$

其中，K 是共现矩阵中已知评分 r_{ui} 的集合，λ 是正则项系数。系统基于已知的评分以最小化均方误差为目标来学习 $\boldsymbol{q}_i$ 和 $\boldsymbol{p}_u$，并通过引入正则化项来避免过拟合。

分别对 $\boldsymbol{q}_i$ 和 $\boldsymbol{p}_u$ 求偏导数，得到梯度下降的方向和幅度分别为

$$2\left(r_{ui} - \boldsymbol{q}_i^{\mathrm{T}} \boldsymbol{p}_u\right)\boldsymbol{p}_u - 2\lambda \boldsymbol{q}_i \tag{3}$$

$$2\left(r_{ui} - \boldsymbol{q}_i^{\mathrm{T}} \boldsymbol{p}_u\right)\boldsymbol{q}_i - 2\lambda \boldsymbol{p}_u \tag{4}$$

其中，γ 是学习率。然后，沿梯度的反方向对 $\boldsymbol{q}_i$ 和 $\boldsymbol{p}_u$ 进行更新，即

$$\boldsymbol{q}_i \leftarrow \boldsymbol{q}_i - \gamma\left(\left(r_{ui} - \boldsymbol{q}_i^{\mathrm{T}} \boldsymbol{p}_u\right)\boldsymbol{p}_u - \lambda \boldsymbol{q}_i\right) \tag{5}$$

$$\boldsymbol{p}_u \leftarrow \boldsymbol{p}_u - \gamma\left(\left(r_{ui} - \boldsymbol{q}_i^{\mathrm{T}} \boldsymbol{p}_u\right)\boldsymbol{q}_i - \lambda \boldsymbol{p}_u\right) \tag{6}$$

重复式(3)～式(6)，直至迭代次数达到设定的上限或者损失函数（目标函数）收敛，由此得到节点对所有内容的评分矩阵。

在补全边缘节点覆盖区域内的用户（与该边缘节点直接连接的所有用户）对所有内容的评分矩阵后，将该区域内所有用户对某个内容的评分之和除以对所有内容的评分之和（即相对评分）作为该区域内该内容的局部流行度。局部流行度即当地用户对该内容的请求概率，记为 $P = \{P_1, P_2, \cdots, P_I\}$，其中，$I$ 为网络内容总数，$P_i \in P$ 为内容 i 的局部流行度，有

$$P_i = \frac{\sum_{u=1}^{U} \hat{r}_{ui}}{\sum_{i=1}^{I}\sum_{u=1}^{U} \hat{r}_{ui}} \tag{7}$$

其中，U 为与节点直接连接的用户总数。

2.2　边缘计算和缓存的联合优化

设缓存放置策略为 $C = \{C_1, C_2, \cdots, C_I\}$，其中，$C_i \in C$ 表示内容 i 是否被选择缓存。$C_i = 0$ 时表示未缓存该内容，$C_i = 1$ 时表示已缓存。缓存命中率 η 表示为

$$\eta = \sum_{i=1}^{I} P_i C_i \tag{8}$$

设 δ_1 为因没有缓存而经完整回程链路传输一个内容的成本。假设各内容大小一致，且所有用户都处于请求内容状态，根据缓存放置策略 C，有 η 的概率可以直接从节点获取，所以因缓存放置而无须经回程链路传输请求内容的收益为 $\eta U \delta_1$。

设 δ_2 为节点内部署单位缓存容量的成本。总缓存部署成本随缓存容量 V 的增大而增大，所以部署缓存容量的支出为 $V\delta_2$。

设 $\triangle\tau$ 为在最后期限 $\tau_{\max}$ 前完成计算任务的平均节省时间。设 δ_3 为运营商执行计算任务期间支付的成本。假设与节点直接连接的用户都处于发布计算任务状态，由于在节点部署了计算模块，部分计算任务可由节点处理而无须卸载到远程计算服务器，因此计算收益为 $\triangle\tau U \delta_3$。如果节点在其最后期限前完成任务，则节省的时间为正，相应的计算收益也为正；否则节约的时间和计算收益皆为负，因为对于时延敏感的任务来说，如果没有在最后期限前完成，可能会导致一定的损失。

设 δ_4 为节点内部署单位计算资源的成本。总计算部署成本随着计算资源 S 的增大而增大。所以部署计算资源的支出为 $S\delta_4$。

上述部署缓存容量和计算资源的支出 $V\delta_2$ 和 $S\delta_4$ 为经济成本，而因缓存放置直接从节点获取内容和因部署计算模块提前完成计算任务的收益 $\eta U \delta_1$ 和 $\triangle\tau U \delta_3$ 并不直接反映在经济盈利上，其不仅包含无须经完整回程链路传输所节约的能量，也包含用户愿意为了更低时延而支付的费用以及运营商因为更低的带宽资源消耗所减少的开销。从网络优化的角度看，支出和收益都是可以通过经济指标来衡量的，因此优化目标为：使利润（运营收益）$\rho = \eta U \delta_1 - V\delta_2 + \triangle\tau U \delta_3 - S\delta_4$ 最大。约束条件如下。

1) 缓存容量：$\sum_{i=1}^{I} C_i \leqslant V \leqslant I$。

2) 请求内容的平均传输时延：

$\sum_{i=1}^{I} P_i[t_1 C_i + t_2(1-C_i)] \leqslant t_{\max}$。其中，$t_1$ 是从节点缓存处获得内容的时延，t_2 是从远程内容服务器处获取内容的时延，$t_{\max}$ 是服务质量（QoS, quality of service）允许的获取内容的最大平均时延，有 $t_1 < t_2$。整理得 $\sum_{i=1}^{I} P_i C_i \geqslant \frac{t_{\max} - \sum_{i=1}^{I} P_i t_2}{t_1 - t_2}$。

3) 计算资源：$S \leqslant U$。S 个用户发布的计算任务由边缘节点处理，根据不同用户不同应用对时延的要求选择这 S 个用户。

4) 完成计算任务的平均时延：$\frac{\tau_1 S + \tau_2(U-S)}{U} \leqslant \tau_{\max}$。其中，$\tau_1$ 是在边缘节点处完成计算任务的时延，τ_2 是在远程计算服务器处完成计算任务的时延，$\tau_{\max}$ 是 QoS 允许的完成计算任务的最大平均时延，有 $\tau_1 < \tau_2$。整理得 $S \geqslant \frac{(\tau_{\max} - \tau_2)U}{\tau_1 - \tau_2}$。

综上，在平均时延的约束下，以系统运营收益最大化为目标，资源分配与缓存放置策略的联合优化问题可表示为

$$\begin{aligned} &\max_{C_i, V, S} \rho = \eta U \delta_1 - V \delta_2 + \triangle \tau U \delta_3 - S \delta_4 \\ &\text{s.t.} \sum_{i=1}^{I} C_i \leqslant V \leqslant I,\ V \in N \\ &\sum_{i=1}^{I} P_i C_i \geqslant \frac{t_{\max} - \sum_{i=1}^{I} P_i t_2}{t_1 - t_2},\ C_i \in \{0,1\} \\ &\frac{(\tau_{\max} - \tau_2)U}{\tau_1 - \tau_2} \leqslant S \leqslant U,\ S \in N \end{aligned} \tag{9}$$

显然，这是一个混合整数规划问题，且是 NP-Complete 的，本文利用深度强化学习对其进行求解。边缘节点的计算及缓存能力、用户发布的计算任务和请求的内容以及局部内容流行度等信息被收集并发送给智能体，在获得上述信息后，智能体会设计一个动作来执行资源分配和缓存放置策略，由此进入新的环境状态，并计算系统运营收益作为对该行动的反馈。上述过程对应深度强化学习中的 3 个关键要素，即状态、行动和奖励。

1) 状态。作为深度强化学习算法的输入，状态包含了智能体做出动作所需要的全部信息。本文的状态由 3 个部分组成：$\text{State} = (u_i, s_i, v_i)$，其中，$u_i$ 是与节点 i 直接连接的所有用户的状态，包括其发布的计算任务、请求的内容、任务的截止期限以及该区域的局部内容流行度；s_i 和 v_i 分别是该节点的可用计算资源和缓存容量。

2) 行动。作为深度强化学习算法的输出，行动是智能体对环境产生影响的方式。本文的行动由 3 个部分组成：$\text{Action} = (S_i, V_i, C_i)$，其中，$S_i$ 和 V_i 分别是智能体分配给节点 i 的计算资源和缓存容量，C_i 是该节点采取的缓存放置策略。

3) 奖励。作为智能体学习的指导，从 Learning= Representation+Evaluation+Optimization 的角度看，奖励是 Evaluation 的重要组成成分。由于深度强化学习的目标是最大化累积奖励，故即时奖励的设置应与上述联合优化问题的目标（尽可能找到使系统运营收益最高的资源分配和缓存放置策略）相关。本文将当前的系统运营收益 ρ 与现有的最高系统运营收益 $\rho_{\max}$ 的差值作为即时奖励 $\text{Reward} = \rho - \rho_{\max}$。当累积奖励收敛时，表明最佳资源分配和缓存放置策略训练完成。

本文基于 DDPG 算法对计算和缓存的资源分配以及缓存放置策略进行联合优化，算法流程如图 3 所示。DDPG 智能体由 3 个部分构成：主网络、目标网络和经验回放池。

主网络由 2 个深度神经网络组成，即一个行动网络 Actor-M 和一个评价网络 Critic-M。行动网络用于动作的探索，根据随机动作策略尽可能完整地探索动作空间，即通过引入随机噪声，将动作的决策由确定性过程转变为随机过程，再从该随机过程中采样得到要采取的行动。评价网络通过 Q 值评估行动网络选择的动作，并通过计算策略梯度来更新行动网络。

目标网络包括一个目标行动网络 Actor-T 和一个目标评价网络 Critic-T。目标网络的输入是经验元组 (s_i, a_i, r_i, s_{i+1}) 的下一个状态 s_{i+1}，输出是用于更新 Critic-M 的目标 Q 值。

经验回放池用于储存经验元组，经验元组为智能体在与环境交互过程中所得到的状态转移序列 (s_i, a_i, r_i, s_{i+1})，包括当前状态、所选动作、奖励和下一个状态。在主网络和目标网络的更新阶段，会从经验回放池中随机采样，以减小数据相关性的影响。

基于 DDPG 算法的详细工作过程如下。

1) 将当前环境状态 s_t 输入主网络的 Actor-M，智能体根据随机动作策略采取行动 a_t，即计算资源和缓存容量的分配以及缓存放置策略。

$$a_t = \pi(s_t) = \mu(s_t \mid \theta_\mu) + N_t \tag{10}$$

其中，μ 是由 Actor-M 的卷积神经网络（CNN,

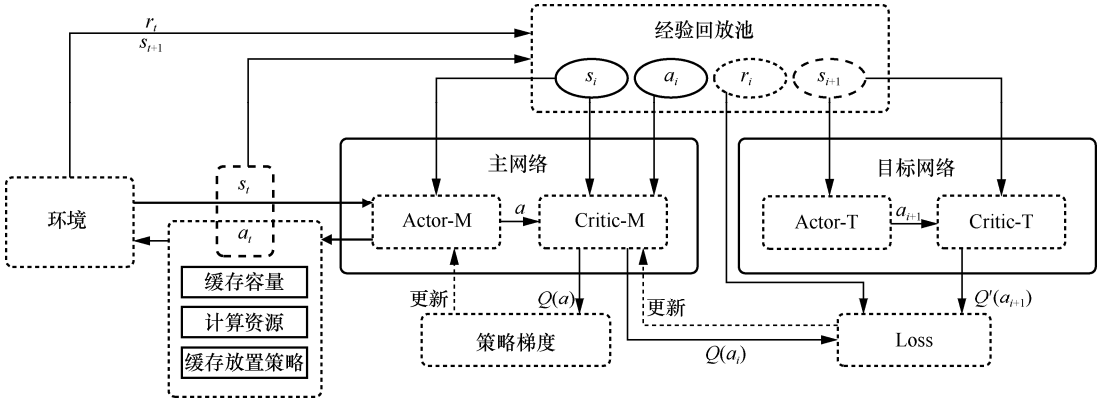


图 3　DDPG 算法流程

convolutional neural network）模拟的确定性动作策略，θ_μ 是 Actor-M 的动作策略参数，N_t 是动作探索噪声。

2) 环境进入下一个状态 s_{t+1}，并向智能体反馈即时奖励 r_t。

3) 将 (s_t, a_t, r_t, s_{t+1}) 存入经验回放池。

4) 从经验回放池中随机采样小批量的含有 N 个经验元组 (s_i, a_i, r_i, s_{i+1})。

5) 将 s_i 和 a_i 输入主网络的 Critic-M，Critic-M 利用 CNN 模拟贝尔曼方程计算在状态 s_i 下选择动作 a_i 的 Q 值为

$$Q(s_i, a_i \mid \theta_Q) = \mathrm{E}[r_i + \gamma Q(s_{i+1}, \mu(s_{i+1} \mid \theta_\mu) \mid \theta_Q)] \quad (11)$$

其中，θ_Q 是 Critic-M 的 Q 值参数。

6) 将 s_{i+1} 输入目标网络的 Actor-T，Actor-T 通过确定性动作策略 μ' 得到动作 $a_{i+1} = \mu'(s_{i+1} \mid \theta_{\mu'})$，其中，$\theta_{\mu'}$ 是 Actor-T 的动作策略参数。再将 s_{i+1} 和 a_{i+1} 输入 Critic-T，得到在状态 s_{i+1} 下、选择动作 a_{i+1} 的目标 Q 值 $Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta_{\mu'}) \mid \theta_{Q'})$，其中，$\theta_{Q'}$ 是 Critic-T 的 Q 值参数。由 r_i 和 $Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta_{\mu'}) \mid \theta_{Q'})$ 得到在状态 s_i 下选择动作 a_i 的目标 Q 值为

$$Q'(s_i, a_i \mid \theta_{Q'}) = r_i + \gamma Q'\left(s_{i+1}, \mu'(s_{i+1} \mid \theta_{\mu'}) \mid \theta_{Q'}\right) \quad (12)$$

7) 将 $Q'(s_i, a_i \mid \theta_{Q'})$ 传入 Critic-M，通过最小化损失函数 Loss 来更新 Critic-M 的 Q 值参数 θ_Q，即

$$\min \mathrm{Loss} = \frac{1}{N}\sum_i^N \left(Q(s_i, a_i \mid \theta_Q) - Q'(s_i, a_i \mid \theta_{Q'})\right)^2 \quad (13)$$

8) 将 s_i 输入主网络的 Actor-M，Actor-M 通过确定性动作策略 μ 选择动作 $a = \mu(s_i \mid \theta_\mu)$。再将 s_i 和 a 输入 Critic-M，通过策略梯度来更新 Actor-M 的动作策略参数 θ_μ，即

$$\nabla_{\theta_\mu} J = \int_s \rho^\mu(s) \nabla_a Q(s, a \mid \theta_Q)\big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta_\mu} \mu(s \mid \theta_\mu)\big|_{s=s_i} \mathrm{d}s = \mathrm{E}_{s\sim\rho^\mu}[\nabla_a Q(s, a \mid \theta_Q)\big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta_\mu} \mu(s \mid \theta_\mu)\big|_{s=s_i}] \quad (14)$$

其中，$\rho^\mu(s)$ 是在确定性动作策略 μ 下状态 s 服从的分布函数；$J = \int_s \rho^\mu(s) Q(s, a \mid \theta_Q)\big|_{a=\mu(s)} \mathrm{d}s = \mathrm{E}_{s\sim\rho^\mu}[Q(s, a \mid \theta_Q)\big|_{a=\mu(s)}]$ 是状态 s 服从 $\rho^\mu(s)$ 分布时，智能体根据策略 μ 选择动作能够产生的 Q 值的期望，用以衡量策略 μ 的表现。

基于蒙特卡罗方法，将小批量 N 的经验元组数据代入式(14)，可以作为对策略梯度的无偏估计。

$$\nabla_{\theta_\mu} J = \frac{1}{N}\sum_i^N \nabla_a Q(s, a \mid \theta_Q)\big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta_\mu} \mu(s \mid \theta_\mu)\big|_{s=s_i} \quad (15)$$

9) 通过软更新算法对目标网络中 Critic-T 的 Q 值参数 $\theta_{Q'}$ 和 Actor-T 的动作策略参数 $\theta_{\mu'}$ 进行更新。

$$\theta_{Q'} \leftarrow \tau\theta_Q + (1-\tau)\theta_{Q'} \quad (16)$$

$$\theta_{\mu'} \leftarrow \tau\theta_\mu + (1-\tau)\theta_{\mu'} \quad (17)$$

其中，τ =0.001。

如果当前的资源分配和缓存放置策略满足联合优化问题的所有约束条件，并且当前的系统运营收益大于现有的最高系统运营收益，则每幕（episode）的最高系统运营收益更新为当前的系统运营收益，且节点基于当前的资源分配和缓存放置策略更新其状态。如果当前的资源分配和缓存放置策略满足联合优化问题的所有约束条件，但当前的系统运营收益小于或等于现有的最高系统运营收益，则表明智能体没有产生更好的优化方案，每幕的最高运营收益保持不变，且节点仍根据现有的最佳资源分配和缓存放置策略进行状态更新。如果当前的资源分配和缓存放置策略不能满足联合优化问题的任一约束条件，智能体将受到惩罚。

基于 DDPG 算法联合优化资源分配和缓存放置策略的伪代码如算法 1 所示。

算法 1 基于 DDPG 算法联合优化资源分配和缓存放置策略

1) 随机初始化主网络 Actor-M 的动作策略参数 θ_μ 和 Critic-M 的 Q 值参数 θ_Q

2) 初始化目标网络 Actor-T 的动作策略参数 $\theta_{\mu'} \leftarrow \theta_\mu$ 和 Critic-T 的 Q 值参数 $\theta_{Q'} \leftarrow \theta_Q$

3) 初始化经验回放池

4) for episode $= 1, 2, \cdots, M$ do

5) 智能体接收初始环境状态 s_1

6) 令 $\rho_{\max} = 0$

7) for $t = 1, 2, \cdots, T$ do

8) Actor-M 选择动作 $a_t = \mu(s_t \mid \theta_\mu) + \mathrm{N}_t$

9) 执行动作 a_t，从环境中获得即时奖励 $r_t = \rho - \rho_{\max}$ 并进入下一个环境状态 s_{t+1}

10) if $\rho_{\max} < \rho$ then

11) $\rho_{\max} \leftarrow \rho$

12) end if

13) 将 (s_t, a_t, r_t, s_{t+1}) 存入经验回放池

14) 从经验回放池中随机采样含有 N 个经验元组 (s_i, a_i, r_i, s_{i+1}) 的小批量

15) 计算目标 Q 值 $Q'(s_i, a_i \mid \theta_{Q'}) = r_i + \gamma Q'\left(s_{i+1}, \mu'(s_{i+1} \mid \theta_{\mu'}) \mid \theta_{Q'}\right)$

16) 通过最小化损失函数更新 Critic-M $\min \mathrm{Loss} = \frac{1}{N}\sum_i^N \left(Q(s_i, a_i \mid \theta_Q) - Q'(s_i, a_i \mid \theta_{Q'})\right)^2$

17) 通过策略梯度更新 Actor-M $\nabla_{\theta_\mu} J = \frac{1}{N}\sum_i^N \nabla_a Q(s, a \mid \theta_Q)\big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta_\mu} \mu(s \mid \theta_\mu)\big|_{s=s_i}$

18) 通过软更新算法更新目标网络 $\theta_{Q'} \leftarrow \tau\theta_Q + (1-\tau)\theta_{Q'}$，$\theta_{\mu'} \leftarrow \tau\theta_\mu + (1-\tau)\theta_{\mu'}$

19) end for

20) end for

3 仿真分析

3.1 实验环境和参数设置

本文的仿真使用 ndnSIM2.8 模拟器。仿真场景如图 4 所示，3 个边缘路由器（即边缘节点）覆盖了 3 个不同区域内的用户。在豆瓣电影数据集中收集了 3 个省份共 500 名用户对 1 000 部电影的评分（包含未评分，即一些用户只对其中某些电影打过分）来预测局部内容流行度。对应的远程内容服务器提供 1 000 种不同的内容，每种内容的大小相同。为了体现内容流行度的地域差异性，各边缘路由器覆盖区域内的用户均来自同一省份，每个用户群共 50 人，用户请求兴趣包的频率为 100 个/秒，兴趣包的请求概率分布与局部内容流行度一致。缓存替换策略均采用最近最少使用（LRU, least recently used）策略，仿真时长为 100 s，从用户到远程服务器的路径时延为 60 ms。性能评价指标采用各边缘节点的运营收益、缓存命中率、用户获取数据包的平均时延、平均跳数和远程服务器负载。

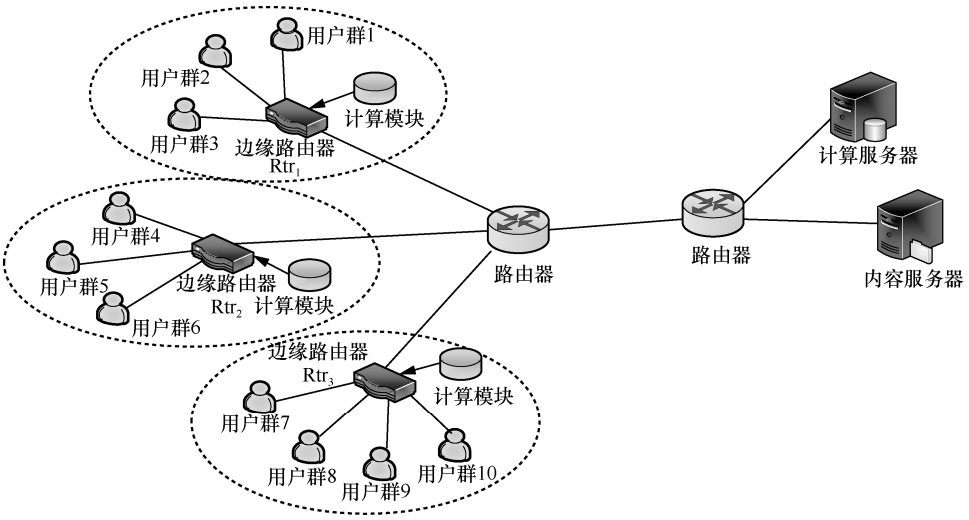


图 4 仿真场景

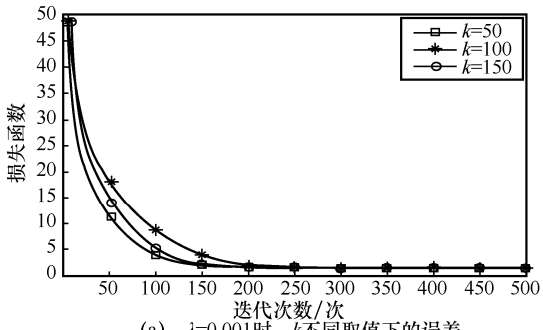
3.2　结果分析

矩阵分解算法中的可调节参数为学习率 γ、正则项系数 λ 和隐向量维度 k。学习率决定每次更新的幅度。对于固定的学习率，如果设置偏大，则会导致结果振荡不收敛；反之，则会导致收敛速度过慢。本文采用指数衰减学习率。

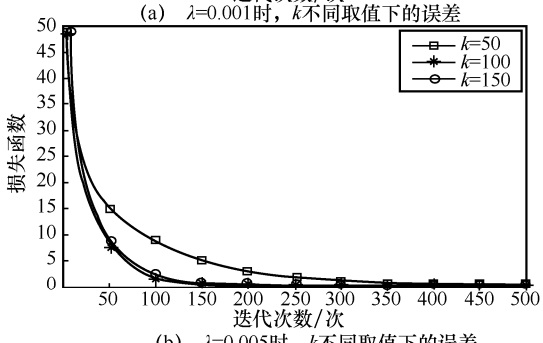
$$\gamma_{\text{exponential\_decay}} = \gamma_{\text{initial}} \text{decay\_rate}^{\left(\frac{\text{global\_step}}{\text{decay\_step}}\right)} \tag{18}$$

其中，$\gamma_{\text{initial}} = 0.002$ 为初始学习率，$\text{decay\_rate} = 0.9$ 为衰减系数，$\text{decay\_step} = 50$ 用来控制衰减速度。在训练前期，采用较大的学习率可以快速获得一个较优的解，随着迭代的继续，学习率逐渐减小，使模型在训练后期更加稳定。

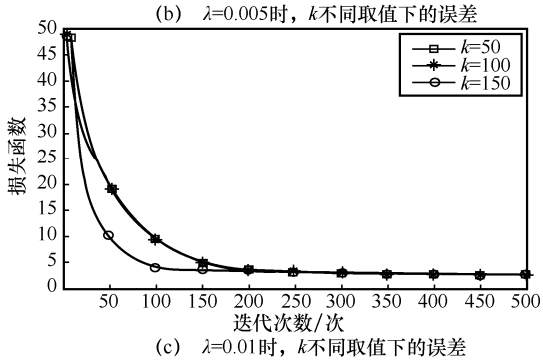
图 5 对比了不同的正则项系数 λ 和隐向量维度 k 组合对矩阵分解损失函数的影响，得出当 $\lambda = 0.005$、$k = 100$ 时损失函数最小，故在后续仿真中均采用此组合。



(a)　λ=0.001时，k不同取值下的误差



(b)　λ=0.005时，k不同取值下的误差



(c)　λ=0.01时，k不同取值下的误差

图 5　不同参数组合的矩阵分解误差对比

在确定了矩阵分解的参数组合之后，按照 2.1 节的算法来预测内容流行度。图 6 以全局流行度排名前 100 位的内容为参照，展示了全局和局部内容流行度的差异性。其中，数据集内全部 500 名用户的评分反映全局内容流行度，而局部内容流行度则由在同一省份随机抽取的 50 名用户计算而得。

如图 6 所示，全局内容流行度大致符合 Zipf 分布，即网络中的少数内容占据了用户的大部分关注度，但在尾部也有持续稳定的小众需求。局部内容流行度则更显著地表现了冷门内容的潜力，出现了很多突出的尾部内容。全局和局部内容流行度的差异性表明本文基于局部内容流行度制定缓存放置策略更加有效，为不同区域用户提供个性化网络服务的同时，还能获得更高的系统运营收益。

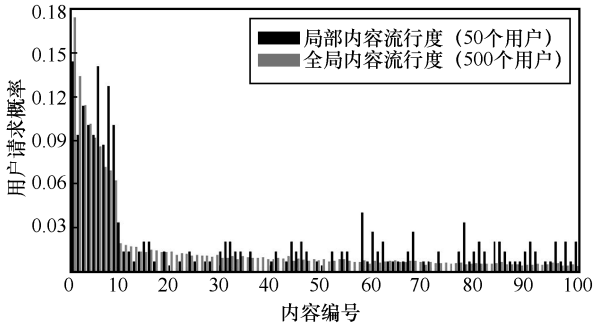
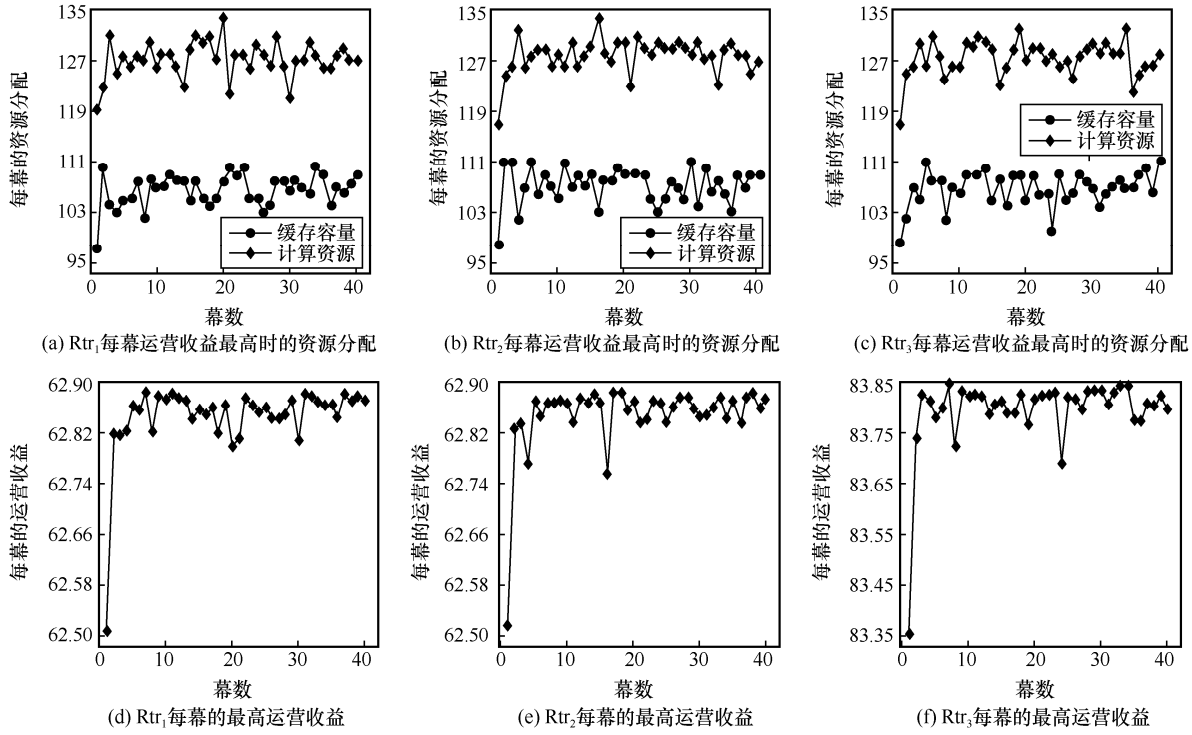


图 6　全局和局部内容流行度对比

在对局部内容流行度完成预测后，边缘节点 Rtr_1、Rtr_2 和 Rtr_3 以系统运营收益最大化为目标，基于 DDPG 算法对计算和缓存的资源分配以及缓存放置策略进行联合优化。图 7 显示了各边缘节点每幕运营收益最高时的资源分配情况及最高运营收益。当节点缓存容量 $V = 108$，计算资源 $S = 127$ 时，运营收益达到峰值。

δ_1（经完整回程链路传输一个内容的成本）、δ_2（节点内部署单位缓存容量的成本）、δ_3（运营商执行计算任务期间支付的成本）和 δ_4（节点内部署单位计算资源的成本）的值皆根据实际边缘计算和缓存设备厂商提供的相关价格资料设置，表 1 对比了 δ_1、δ_2、δ_3 和 δ_4 不同组合下运营收益最高时的资源分配情况。本文方案在不同组合下能够自适应地调节计算和缓存资源的分配，达到相对稳定的平均收益。

本文将 DDPG 算法和经典的 DQN 算法进行了横向对比，如图 8 所示，DDPG 算法在平均缓



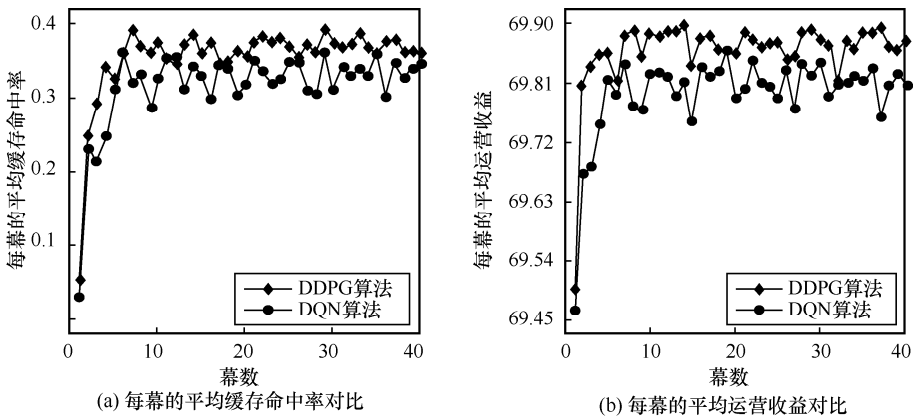
(a) Rtr_1每幕运营收益最高时的资源分配　(b) Rtr_2每幕运营收益最高时的资源分配　(c) Rtr_3每幕运营收益最高时的资源分配

(d) Rtr_1每幕的最高运营收益　(e) Rtr_2每幕的最高运营收益　(f) Rtr_3每幕的最高运营收益

图 7　基于 DDPG 算法优化的各边缘节点的资源分配和运营收益

表 1　　**δ_1、δ_2、δ_3和δ_4不同组合下的资源分配和运营收益**

组合	δ_1	δ_2	δ_3	δ_4	资源分配	平均收益
1	0.25	0.10	0.30	0.10	缓存容量=108，计算资源=127	69.88
2	0.35	0.10	0.15	0.10	缓存容量=136，计算资源=79	67.29
3	0.20	0.10	0.35	0.10	缓存容量=92，计算资源=138	71.13



(a) 每幕的平均缓存命中率对比　(b) 每幕的平均运营收益对比

图 8　基于 DDPG 算法和基于 DQN 算法优化的性能对比

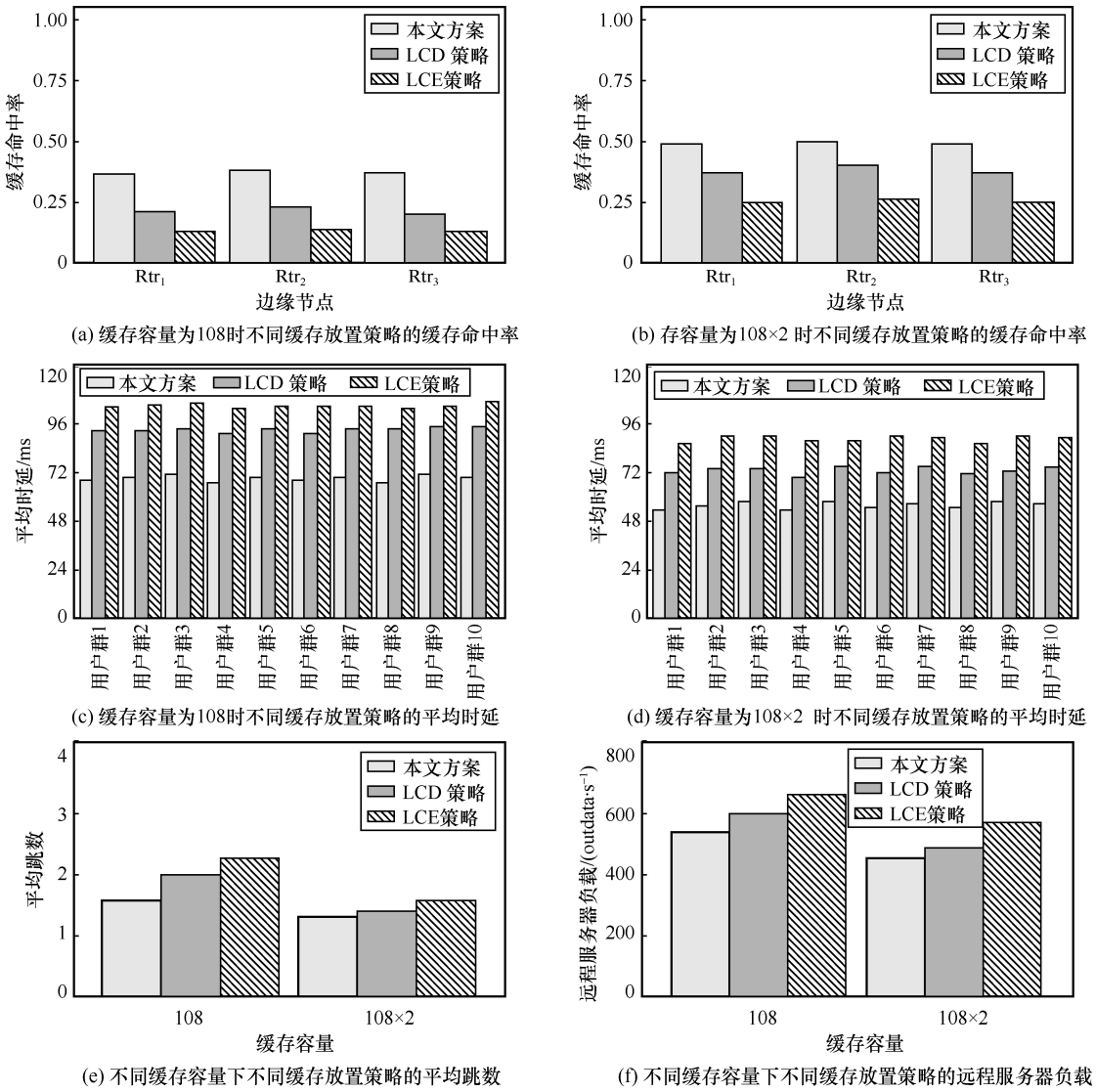
存命中率和平均运营收益两方面的表现都明显优于 DQN 算法，主要原因为 DDPG 采用软更新算法更新目标网络的参数，这种加权移动平均法保证了目标网络训练的稳定性；与 DQN 在每步都计算所有可能动作的 Q 值来进行决策不同，DDPG 采用确定性策略，直接由神经网络 Actor 预测出该状态下需要采取的动作，降低了算法复杂度、加快了收敛速度。

LCE（leave copy everywhere）策略和 LCD（leave copy down）策略[16]是 NDN 缓存放置的基准策略。LCE 策略是 NDN 中默认采用的缓存放置策略，其在数据包返回路径上的每个节点都会缓存该内容的副本，这极易导致网内缓存的冗余度较高且在缓存容量较小时缓存替换频繁。LCD 策略相对 LCE 策略更加保守，只在命中节点的下一跳进行缓存，避免了相同内容在网络中大量复制，在一定程度上降低了缓存冗余度；此外，LCD 策略需要对同一内容进行多次请求才能将其缓存到网络边缘，间接地考虑了内容流行度，使缓存利用率有所提升。

图 9 对比了在节点缓存容量为 108（运营收益最高时的缓存容量）和108×2时，本文 LCD 策略和 LCE 策略的相关性能。在缓存命中率、平均时延、平均跳数（数据包从远程服务器或缓存节点返回所经过的网络跳数）和远程服务器负载方面，本文方案都优于 LCD 策略和 LCE 策略，尤其在缓存容量相对较小时，其优势更加明显：在节点缓存容量为 108 时，相对 LCD 策略，本文方案的缓存命中率提高了 70.97%，平均时延、平均跳数和远程服务器负载分别降低了 25.97%、21.11%和 9.91%。

4　未来方向

前文讨论了基于机器学习在 NDN 网络边缘联合优化计算和缓存的潜力，通过仿真证明了其对运营收益和系统性能的提升。在此基础上，本文提出



(a) 缓存容量为108时不同缓存放置策略的缓存命中率　(b) 存容量为108×2 时不同缓存放置策略的缓存命中率

(c) 缓存容量为108时不同缓存放置策略的平均时延　(d) 缓存容量为108×2 时不同缓存放置策略的平均时延

(e) 不同缓存容量下不同缓存放置策略的平均跳数　(f) 不同缓存容量下不同缓存放置策略的远程服务器负载

图 9　不同缓存容量下不同缓存放置策略的性能对比

以下未来的研究方向。

1) 通过各边缘节点间的相互合作进行实时的情境感知[17-21]，包括信道条件、用户的移动性、计算任务和内容请求的规模与优先级等，以优化移动性管理和主动资源分配。例如，实现对计算任务的自适应分流，缓解单一边缘节点计算压力的同时，充分利用当前相邻区域的闲置资源，提供更低时延的计算服务；挖掘不同区域内容流行度之间的相关关系，预判和缓存呈现由局部到全局流行态势的内容，缩短用户获取内容的时延。

2) 利用联邦学习这一面向隐私保护的分布式机器学习框架[22]。联邦学习在不共享原始数据的基础上，聚合各边缘节点的本地训练的中间结果，构建具有较高泛化能力的共享模型，以推进全局范围内的性能优化，既保证了各参与方之间的数据隔离，又实现了数据价值的安全流通。

3) 利用机器学习优化种类庞杂的计算和缓存服务时，往往需要一定的时间才能使模型收敛，如何提高边缘节点机器学习的效率亦是一个亟待解决的问题。可以基于软件定义网络（SDN, software defined network）和网络功能虚拟化（NFV, network function virtualization）实现网络切片[23]，结合机器学习为不同类型的服务提供差异化的支持，实现高效灵活的边缘计算和缓存，并采用灵活以太网（FlexE, flexible Ethernet）技术使业务流以最短、最快的路径抵达用户[24]。

5 结束语

本文提出了一个NDN与边缘计算相结合的综合框架并基于深度强化学习对资源分配与缓存放置策略进行联合优化，以实现网络、计算和缓存的动态协调。首先，在NDN边缘节点部署计算模块，结合NDN的网内缓存机制，将网络功能、内容和资源向终端用户靠近；然后，利用矩阵分解算法强大的处理稀疏矩阵的能力，补全各区域用户对内容的评分矩阵，将区域内所有用户对某个内容的相对评分作为该区域内该内容的局部流行度；最后，在平均时延的约束下，以系统运营收益最大化为目标，利用DDPG算法对计算和缓存资源分配以及缓存放置策略进行联合优化。仿真结果显示，本文方案在稳定性、收敛速度和性能表现上都优于经典的DQN算法；与传统的缓存放置策略相比，本文方案可以更有效地提高缓存命中率、降低用户获取内容的平均时延，综合提升运营收益和用户体验质量。

参考文献：

[1] ZHANG L X, AFANASYEV A, BURKE J, et al. Named data networking[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 66-73.

[2] JAYAKUMAR S, SHEELVANTHMATH P, AKKI C B. Technical analysis of content placement algorithms for content delivery network in cloud[J]. International Journal of Electrical and Computer Engineering (IJECE), 2022, 12(1): 489.

[3] ZULFA M I, HARTANTO R, PERMANASARI A E. Caching strategy for Web application—a systematic literature review[J]. International Journal of Web Information Systems, 2020, 16(5): 545-569.

[4] CHEN M, HAO Y X, HU L, et al. Edge-CoCaCo: toward joint optimization of computation, caching, and communication on edge cloud[J]. IEEE Wireless Communications, 2018, 25(3): 21-27.

[5] JIANG C F, CHENG X L, GAO H H, et al. Toward computation offloading in edge computing: a survey[J]. IEEE Access, 2019, 7: 131543-131558.

[6] XU J, CHEN L X, ZHOU P. Joint service caching and task offloading for mobile edge computing in dense networks[C]//Proceedings of IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2018: 207-215.

[7] JAYAKUMAR S, SHEELVANTHMATH P, AKKI C B. Technical analysis of content placement algorithms for content delivery network in cloud[J]. International Journal of Electrical and Computer Engineering (IJECE), 2022, 12(1): 489.

[8] POLYAK B, SHCHERBAKOV P. Lyapunov functions: an optimization theory perspective[J]. IFAC-PapersOnLine, 2017, 50(1): 7456-7461.

[9] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing Atari with deep reinforcement learning[J]. arXiv Preprint, arXiv: 1312.5602, 2013.

[10] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

[11] CHEN X T, GE H B, LIU L H, et al. Computing offloading decision based on DDPG algorithm in mobile edge computing[C]//Proceedings of 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics. Piscataway: IEEE Press, 2021: 391-399.

[12] KOREN Y, BELL R, VOLINSKY C. Matrix factorization techniques for recommender systems[J]. Computer, 2009, 42(8): 30-37.

[13] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning[J]. arXiv Preprint, arXiv:1509.02971, 2015.

[14] ZHU H, CAO Y, WANG W, et al. Deep reinforcement learning for mobile edge caching: review, new features, and open issues[J]. IEEE Network, 2018, 32(6): 50-57.

[15] ZHU G X, LIU D Z, DU Y Q, et al. Toward an intelligent edge: wireless communication meets machine learning[J]. IEEE Communications Magazine, 2020, 58(1): 19-25.

[16] LAOUTARIS N, CHE H, STAVRAKAKIS I. The LCD interconnection of LRU caches and its analysis[J]. Performance Evaluation, 2006, 63(7): 609-634.

[17] CHEN M, LI W, FORTINO G, et al. A dynamic service migration mechanism in edge cognitive computing[J]. ACM Transactions on Internet Technology, 2019, 19(2): 1-15.

[18] ZHANG C, ZHENG Z X. Task migration for mobile edge computing using deep reinforcement learning[J]. Future Generation Computer Systems, 2019, 96: 111-118.

[19] HE Y, YU F R, ZHAO N, et al. Software-defined networks with mobile edge computing and caching for smart cities: a big data deep reinforcement learning approach[J]. IEEE Communications Magazine, 2017, 55(12): 31-37.

[20] KADER M A, BASTUG E, BENNIS M, et al. Leveraging big data analytics for cache-enabled wireless networks[C]//Proceedings of 2015 IEEE Globecom Workshops. Piscataway: IEEE Press, 2015: 1-6.

[21] YU Z X, HU J, MIN G Y, et al. Proactive content caching for Internet-of-vehicles based on peer-to-peer federated learning[C]//Proceed ings of 2020 IEEE 26th International Conference on Parallel and Distributed Systems. Piscataway: IEEE Press, 2020: 601-608.

[22] KAIROUZ E B P, MCMAHAN H B. Advances and open problems in federated learning[J]. arXiv Preprint, arXiv: 1912.04977, 2019.

[23] BANNOUR F, SOUIHI S, MELLOUK A. Distributed SDN control: survey, taxonomy, and challenges[J]. IEEE Communications Surveys & Tutorials, 2018, 20(1): 333-354.

[24] DING Z Y, LI W, CHENG Y F, et al. Slice network framework and use cases based on FlexE technology for power services[C]//Proceedings of 2021 International Wireless Communications and Mobile Computing (IWCMC). Piscataway: IEEE Press, 2021: 57-62.

[作者简介]

张宇（1972− ），男，山西原平人，博士，北京理工大学讲师，主要研究方向为网络的路由和缓存等资源分配优化、网络体系架构、协议实现和 NS-3 仿真等。

程旻（1997− ），女，安徽黄山人，北京理工大学硕士生，主要研究方向为命名数据网络、边缘计算和网络资源优化等。