一种基于 DQN 的全景视频边缘缓存优化方案

摘要: 为了解决全景视频服务中云服务器和边缘服务器的联合边缘缓存问题，优化边缘缓存机制以降低用户获取视频资源的时延，提出用 DQN 进行深度强化学习以生成视频资源缓存策略的方法。首先，以总节约时间为目标将问题建模成马尔可夫决策过程；其次，使用 DQN 算法进行训练，在迭代中获取最优缓存策略。仿真结果表明，DQN 算法具有最优的性能和较高的收敛速度，且在约束条件改变时能主动变换调整边缘缓存策略使得算法性能稳定上升。

关　键　词: 边缘计算；深度强化学习；缓存策略；网络时延

中图分类号: TP393　　**文献标识码**: A

An Optimization Scheme of Edge Caching based on DQN

Abstract: To solve the edge caching problem of base station and edge server, optimizing the edge caching mechanism to reduce the time delay for obtaining network services, a method of generating cache strategy by using DQN as deep reinforcement learning algorithm is proposed. First, the problem is modeled as Markov decision process with the goal of total time saving. Then, DQN algorithm is used for training to obtain the best cache strategy in the iteration. Simulation shows that DQN algorithm has the best performance and high convergence speed and when the constraints change, it can actively change the edge caching strategy to stably improve the performance of the algorithm.

Key words: edge computing; deep reinforcement learning; cache strategy; network delay

1　引言

随着第五代移动通信技术的发展和移动设备的广泛使用，新一代的移动端网络服务面临着网络流量指数级增长的挑战。尤其是在虚拟现实领域，随着全景视频资源的丰富和网络通信技术的发展，越来越多的用户开始关注虚拟现实技术，并有机会体验到在移动端观看全景视频的新型娱乐方式。一些研究表明，如果视频质量不佳，用户会很快放弃视频会话[1][2]。因此，如何在有限的网络条件下，尽可能的满足用户对视频质量的需求，成为了视频提供商的努力方向。

实际上如何为用户提供高质量的全景视频观看体验是一个非常具有挑战性的问题。首先，与常规视频相比，传输 360°视频所消耗的带宽要高的多。这是因为其全景特性会导致他在相同感知质量下需要传输的数据量比传统视频大得多，一般是普通视频数据的 4 倍到 6 倍。其次，全景视频需要非常低的延迟。目前比较流行的全景视频传输方案是比特率自适应，也就是在传输时，根据网络状况选择合适等级的比特率。常用的比特率自适应的视频流协议就是 MPEG-DASH 流，DASH 播放器中包含多个可供选择的比特率，它可以在网络状态改变时为全景视频更换合适的比特率。在网络条件变好时，会低比特率转换到高比特率，这样会产生一定的延迟，我们需要尽可能把这个延迟降到最低。因为过高的延迟不仅仅会影响用户的观看体验，还可能导致用户产生晕动症而对其健康造成影响[3]，因此全景视频的延迟要求在各方面都会比正常的平面视频更加严格。高带宽和低延迟的矛盾要求全景视频在传输时必须合理利用有限的网络资源，而利用边缘缓存和计算构建云边端协同的传输系统则是一个有效的解决方案。

边缘缓存和计算旨在将部分视频资源的计算

和存储从核心网（中心服务器、云端等）转移到边缘节点（大小基站、小型数据中心、终端设备等）上[4]，通过使用远端云服务器和边缘服务器结合的边缘存储技术，我们可以一定程度上降低用户获取全景视频资源的时间延迟，因为边缘服务器距离用户较近，当边缘服务器有缓存用户所请求的视频资源时，就可以直接通过边缘服务器传输给用户，从而达到低延迟的网络传输。然而该模式下，边缘服务器虽然距离用户近，传输延迟低，但是其储存能力是有限的，相反云服务器的储存能力很强，但是距离用户太远，无法提供低延迟的网络传输。如果边缘服务器没有储存用户所请求的视频资源，则需要向远端服务器请求对应的数据，然后再传输给用户，这个过程同样会造成高延迟的网络传输。在这个基础上，边缘服务器应该缓存什么内容就成为了我们关心的问题，因为如果在对应的时间段内缓存到了对应的视频内容，就可以很大程度上降低总获取时延。

因此为了优化边缘缓存策略的性能，本文使用了深度强化学习方法，提出了基于 DQN 算法的全景视频缓存优化方案，本文的主要贡献如下：

（1）将边缘缓存决策问题建模成马尔可夫决策过程(Markov decision process)，其中我们使用综合基站缓存状态和服务请求状态两方面来描述当前状态(state)，基站进行的缓存状态的变化即为动作(action)，节余的总时间定义为回报(reward)。

（2）为了解决建模后的问题，我们使用了 DQN 深度强化学习算法，以达到在无未来信息和状态转移概率的情况下进行自动学习的效果。

（3）我们模拟了变化的网络环境进行了实验，并设计了其他两种算法和 DQN 进行对比，分别是将流行度高的服务进行边缘基站本地计算的 Popular 算法和随机选择服务进行边缘基站本地计算的 Random 算法。仿真结果表明，DQN 算法相较于其他算法有着最高的总节约时间和收敛速度，同时在改变计算能力这一限制条件时，尽管算法性能的提升会遇到瓶颈，相较于 Popular 算法，DQN 可以主动调整自己的缓存策略使得算法性能稳定上升。

2 相关工作（还需要总结不足和我们的优势）

在这一节中，我们将对文献中关于边缘缓存策略的最有价值的工作进行概述。之前一些学者为了解决我们上面提到的问题，选择从流行度的角度出发，通过算法预测流行度较高的资源并在可能的范围内更多的缓存这些内容。Ghosh 等[5]提出了一种可扩展的分层缓存机制，可以智能地计算内容流行度并缓存最佳位置的内容。Xia 等[6]则研究了一种在线运行的边缘缓存决策方法，该方法从移动端应用程序供应商的角度出发，可以在未来数据需求未知的情况下计算移动边缘缓存的最优决策。从服务内容角度出发进行缓存策略的研究之前也备受关注。Zhan 等[7]通过预取缓存的方法，研究了流媒体中边缘缓存策略。主要从两个方面进行了创新，一是精确模拟了用户设备中的播放缓冲区，并分析它的空闲期，这些空闲期对应于从基础设施下载的字节；二是成功优化移动缓存的内容分配；并最小化预期的非卸载字节数。Zhang 等[8]提出了一种在移动端进行无线通信的面向内容的缓存的缓存方法，其主要基于流行的内容推荐设计了新颖的边缘缓存方案用于解决传统的 D2D 通信和缓存策略的限制下 QoE 增强问题。同时，许多学者也利用预测信息来辅助生成未来的缓存策略，Masood 等[9]提出了一种基于深回归的视频流行度估计，用于移动边缘计算网络中的主动视频缓存。此外，在不同的场景下，许多研究者希望通过不同维度的数据作为参数来辅助决策边缘计算和缓存的策略，Chakareski 等[10]研究了在移动端播放全景视频的问题场景下，利用观看者的视口变化信息分析不同视频切片的观看概率，并依据其概率帮助基站决定缓存策略，这类方法充分利用了特定场景下的多维度信息，从而为边缘计算的决策提供了多种思路。

3 系统框架

我们构建了全景视频传输系统的架构，如图 3.1 所示。其中有云服务器 C_1，云服务器中缓存有所有视频资源，表示为 R_n。对于所有的视频资源 R_n 来说，它们都被分成了三个比特率等级，从低到高表示为 L_1，L_2 和 L_3。系统中存在多个边缘服务器，定义为 E_n，每个边缘服务器服务于一组用户，用户表示为 U_n。用户请求全景视频资源时，每个用户在同一时间只会请求某个视频资源的指定比特率等级，定义为 $R_n(L_m)$，在该系统中，用户请求的内容需要由边缘服务器传输。边缘服务器同样具有缓存视频内容的能力，但是其缓存能力有限，对于用户所请

求的内容，如果为其服务的边缘服务器在该时刻缓存了对应内容，则直接通过边缘服务器传输给用户。否则，如果用户请求的内容并未缓存在边缘服务器中，则边缘服务器首先需要从云服务器中获取资源，然后传输给用户，这个过程的时延大于直接可以直接从边缘服务器传输的情况。例如图 3.1 中，E_1 中缓存了视频 R_1 的 L_2 和 L_3 两个比特率等级的内容，对于用户 U_1 和 U_3 来说，他们请求的内容已经缓存于边缘服务器中，则直接通过 E_1 进行传输，而用户 U_2 请求的是内容 $R_3(L_2)$，该内容并未缓存于 E_1 中，所以需要 E_1 从云服务器 C_1 中获取该内容，然后传输给指定的用户 U_2。

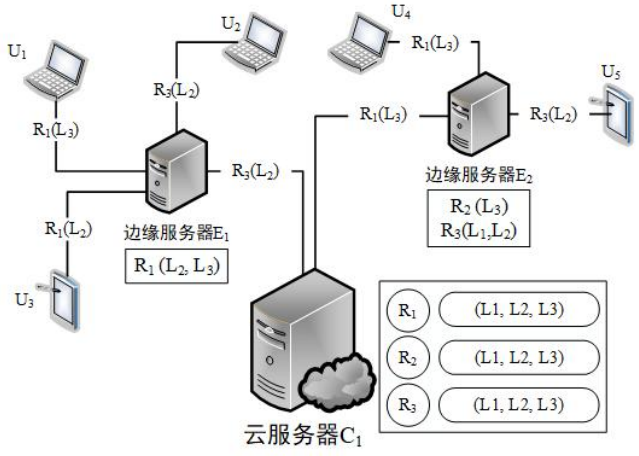

图 3.1 全景视频传输系统架构

此外，边缘服务器还可能缓存了当前时刻未被请求的内容，例如在图 3.1 中，边缘服务器 E_2 在当前时刻缓存了多个视频内容，即 $R_2(L_3)$，$R_3(L_1)$和 $R_3(L_2)$，但是实际上这其中只有内容 $R_3(L_2)$被用户 U_5 所请求，因此剩下的两个内容 $R_2(L_3)$，$R_3(L_1)$可以视为无需缓存的内容，这会造成边缘服务器资源的浪费，在系统中是需要避免的情况。

在该传输系统中，边缘服务器距离用户近，可以快速响应用户请求，降低服务请求时延，但是边缘服务器缓存空间有限，无法缓存所有内容。云服务器虽然可以缓存所有的视频内容，但距离用户较远，服务时延高。综合协同边缘服务器和远端云服务器可以降低用户的视频资源获取时延，其中最重要的问题就是边缘基站应该缓存什么视频内容才能使得总获取时延最小（或节约的总时间最大）。

下面，我们将问题建模成马尔可夫决策过程(Markov decision process)，从状态空间(State Space)，动作空间(Action Space)和回报函数(Reward Function)三方面进行描述。

4 问题建模

4.1 状态空间

状态(State)是描述网络系统环境。在该问题中，我们综合基站缓存状态和服务请求状态两方面来描述当前状态，我们用 S^t 来代表 t 时刻的状态.

我们用向量$x^t=[x_k^t]$代表 t 时刻基站 BS 的缓存状态，其中$x_k^t=1$代表在 t 时刻缓存内容 k，$x_k^t=0$代表 t 时刻不缓存内容 k。由于基站的缓存空间有限，因此应满足(1)中的约束条件：

$$\sum_{k\in K} c_k x_k^t \leq C \tag{1}$$

其中c_k代表缓存内容 k 需要的缓存空间，K 代表内容集合，C 代表基站的缓存空间。

我们用向量$R^t=[r_k^t]$代表 t 时刻基站的用户请求状态，r_k^t代表 t 时刻用户对内容 k 的请求次数。

我们将状态定义为用户请求阶段结束时的系统环境。在 t 时刻用户请求阶段结束时，缓存内容还没有进行调整，因此，此时的缓存状态还是 t-1 时刻的状态，即X^{t-1}，而用户请求已经到达，所以用户请求状态为R^t。因此 t 时刻的网络状态为$S^t=\{(X^{t-1},R^t)|X^{t-1}\in\Gamma\}$，其中$\Gamma$为合法缓存状态的集合，即其中元素满足缓存空间约束条件。

4.2 动作空间

在该内容缓存问题中，基站进行的缓存状态的变化即为动作，我们用向量$A^t=[a_k^t]$表示，其中a_k^t代表 t 时刻基站对内容 k 采取的动作，具体来说，$a_k^t\in\{-1,0,1\}$。当$a_k^t=-1$代表基站移除内容 k，即 t-1 时刻时缓存有内容 k，但 t 时刻不再缓存；$a_k^t=1$代表基站下载缓存内容 k，即 t-1 时刻时没有缓存内容 k，但 t 时刻进行缓存；$a_k^t=0$代表基站不

采取任何动作，即内容 k 在 t-1 时刻和 t 时刻的缓存状态相同。动作和缓存状态间存在(2)中所示关系：

$$X_k^{t-1} + A_k^t \to X_k^t \ with\ probability\ 1 \qquad (2)$$

进一步，我们将动作分为两类：合法动作和不合法动作。不合法动作包括无效动作（例如 t-1 时刻基站缓存了内容 k，则$a_k^t = 1$即为无效动作，因为这是重复下载，无意义；或 t-1 时刻基站没有缓存内容 k，则$a_k^t = -1$因为是重复删除，无意义）和不满足缓存空间约束条件的动作。其他为合法动作。如此定义动作空间的优势在于简化了缓存状态转移的表达，可以通过简单的相加完成状态转移过程。

据此，我们可以定义出该问题的状态转移概率，即$P_{i,j} = P_{i,j}(A^t)$，代表在动作A^t下从状态 i 转移到状态 j 的概率。值得一提的是，虽然状态转移概率可以进行定义，动作和缓存状态间的关系是确定的，但是由于用户请求R^t是随机动态的，因此问题的状态转移概率是难以确定的。

4.3 回报函数

回报函数代表的是在状态 s 下采取动作 a 得到的即时回报。在该问题中，我们的优化目的是通过边缘缓存最小化内容获取时间，即通过边缘缓存后节约的总时间最大，因此，我们将节余的总时间定义为回报。我们对在状态S^t下采取动作A^t后的回报函数为每个分量动作a_k^t的总和，进行如下定义：$F^t = \sum_{k \in K} f_k^t$其中$f_k^t$代表分量动作$a_k^t$带来的回报，定义如(3)中所示：

$$f_k^t = \begin{cases} -\infty,\ if\ A^t\ is\ illegal \\ r_k^t(\delta_k - \mu_k),\ if\ a_k^t = 1 \\ 0,\ if\ a_k^t = -1 \\ x_k^{t-1} r_k^t \delta_k,\ if\ a_k^t = 0 \end{cases} \qquad (3)$$

其中，δ^k为与远端服务器获取相比，在本地基站获取内容 k 节约的时间，μ_k代表基站从远端服务器下载内容 k 的时间，r_k^t代表用户请求数量。因此，对于不合法动作，我们应该尽量避免，因此其回报值为$-\infty$；当$a_k^t = 1$时，基站需要先以时延μ_k从远端服务器下载内容 k，再在本地进行服务用户收益为δ_k，因此，在r_k^t个用户情况下，总节余时间为$r_k^t(\delta_k - \mu_k)$；$a_k^t = -1$时，基站不在本地提供服务 k，因此总节余时间为 0；当$a_k^t = 0$时，总节余时间与 t-1 时刻的缓存状态有关，若 t-1 时刻缓存了内容 k，即$x_k^{t-1} = 1$则总节余时间为$r_k^t \delta_k$，否则为 0，综合两种情况即为$x_k^{t-1} r_k^t \delta_k$。

5 基于深度强化学习的解决方法

5.1 DQN 简介

要解决边缘缓存的决策问题，面临两个挑战，一是要求优化目标的平均节约时间最大，这种优化问题往往需要未来信息（如优化 1-10 时刻的平均节约时间，传统方法在第一个时刻决策时往往就需要知道 1-10 的全部状态信息），但是网络是随机动态的，因此获取未来信息是不现实的，即使通过预测方法进行预测，也往往面临着预测精度低等问题。二是问题中的状态转移概率是不确定的，因此传统的动态优化等方法就无法使用。

深度强化学习方法可以在无未来信息和状态转移概率的情况下进行自动学习，实现平均回报最大，在该问题环境下能够应对随机的网络状态和用户请求，因此，我们采用其中最为经典的方法 DQN 对问题进行求解。

DQN 是针对 Q-learning 的改进，属于无模型(Model-Free)的强化学习算法，采用的是时序差分的采样方法，DQN 有两大优势。一是 DQN 用神经网络对 Q-learning 中的 Q-table 进行模拟，即神经网络输入的是状态，输出的是在此状态下对应动作的 Q 值，这样 DQN 就解决了 Q-learning 无法应对状态空间过大或连续的问题。二是经验回放(Experience replay)，即 DQN 会将自己或别人的经历存储到一个内存中，在 DQN 每次更新的时候，会随机从内存中抽取一些之前的经历进行学习，随机抽取这种做法打乱了经历之间的相关性，也使得神经网络更新

更有效率。通过这两种手段，DQN 可以在一些问题中得到很好的训练效果。

5.2 基于 DQN 的边缘缓存策略

在本节中，我们提出一种基于 DQN 的求解算法，用$Q(S,A)$代表在状态 S 下采取动作 A 的 Q 值。由于传统的ϵ-贪婪策略即使在模型收敛后始终会以一个ϵ的概率去选择不是最优的动作，从而造成了白白的浪费。对此，我们引入置信区间，进行如公式(4)中所示的定义：

$$UCB(S,A)=Q(S,A)+\sqrt{\frac{2\ln n}{n_j}} \tag{4}$$

其中 n 为总的训练次数，n_j 为在状态 S 下选择动作 A 的次数。每次都选择 *UCB(S,A)*最大的动作执行，这样随着模型的训练，被选中次数少的动作被执行的概率也会越来越大，从而保证了模型探索的有效性。采用梯度下降的方式对模型进行训练，loss 函数是目标价值与网络输出价值之间的均方差，loss 函数的定义如(5)中所示：

$$loss=[Q_{target}(S^t,A^t)-Q(S^t-A^t)] \tag{5}$$

其中目标价值为下一个状态的最大 Q 值，如公式(6)中所示：

$$Q_{target}(S^t,A^t)=F^t+\gamma\max_{A'}Q(S^{t+1},A') \tag{6}$$

其中 γ 为折扣因子。基于此，我们提出了相关算法，如下：

算法 1 基于 DQN 的边缘缓存决策算法

初始化回放储存空间 D，设置空间容量为 N

初始化动作函数 Q 并设置随机权重

for 片段=1, M **do**

  for t=1, T **do**

    计算在状态 S 下各个动作的 UCB(S,A)值

    根据 UCB(S,A)的值缓存动作A_t

    在模拟器中执行缓存动作A_t并获得回报值F_t

    将缓存决策形成的数据组(S_t,A_t,F_t,S_{t+1})储存在 D 中

    从 D 中随机取样小批量的数据组(S_j,A_j,F_j,S_{j+1})

    if S_{j+1}是最后一组数据 **then**

      设置缓存动作的 Q 值$Q_{target}=F_j$

    else

      设置缓存动作的 Q 值$Q_{target}=F_j+\gamma\max_{A'}(S_{j+1},A')$

    对$(Q_{target}-Q(S_j,A_j))^2$执行梯度递减的迭代步骤，更新各个缓存动作对应的 Q 值

  end for

end for

算法首先初始化空间容量为N的回放储存空间D和动作函数 Q，之后算法将运行多个循环，在每个循环中选取每个时刻的缓存行为，选取方式为以ϵ的概率选取随机的行为，其他情况选取在当前状态下使得 Q 值最大的行为。之后执行选取的行为并获取回报函数的结果，并将当前状态S_t，当前行为A_t，当前回报F_t以及下一时刻的状态S_{t+1}作为一组数据记录到历史数据存储空间 D 中，之后在 D 中随机选取小批量的样本，对这些样本求得其目标价值Q_{target}和网络输出价值$Q(S_j,A_j)$，之后便得出损失函数的结果，即这两个值的均方差。对损失函数的结果执行梯度下降法的迭代步骤，并进行下一个循环。

6 实验仿真结果

实验环境：实验共设有 10 个服务内容，首先限制边缘基站的计算能力为 8GHz，缓存空间为 8GHz，每个服务内容需要的计算量为【1GHz, 4GHz】之间，需要的缓存空间为【2GB, 4GB】之间。服务的流行度符合 Zipf 分布，用户请求的到达符合泊松分布。之后为了测试不同算法在边缘基站约束条件改变的情况下的表现，保持缓存空间不变

的情况下调整其计算能力为【4GHz, 6GHz, 8GHz, 10GHz, 12GHz】，然后保持计算能力不变的情况下，调整其缓存空间为【5GB, 6GB, 7GB, 8GB, 9GB, 10GB】

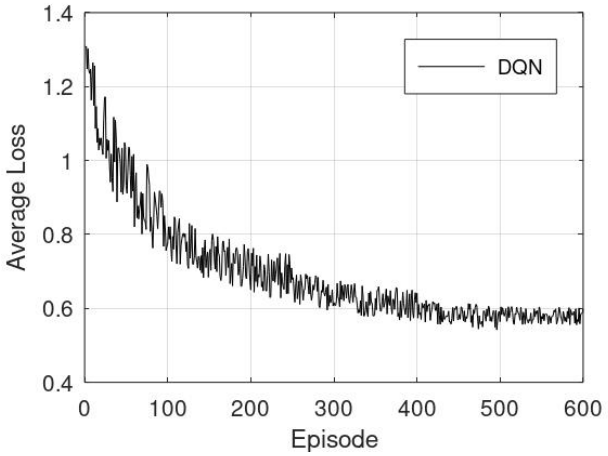

图 1 DQN 算法训练过程中的 Average Loss

图 1 显示了算法在训练过程时 Average loss 的变化过程，体现了算法能够有效收敛。在前 450 次左右的训练中 Average loss 逐渐下降且下降幅度逐渐减小，在 450 次后算法已趋于平稳。另外，随着 Episode 的增加，算法上下震动的幅度也可以看到明显减小。

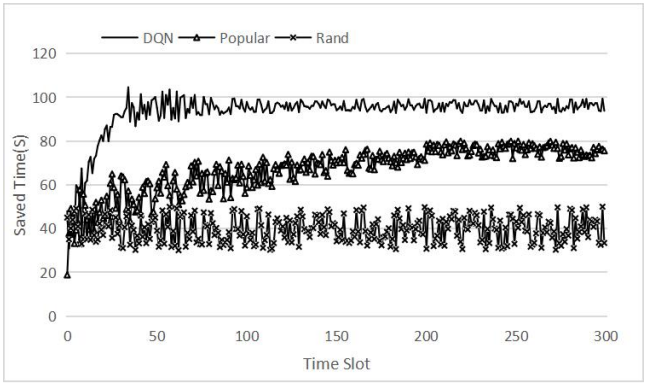

图 2 不同算法下节约的时间

图 2 显示了算法在性能（即通过边缘计算节约的时间）方面的表现，并与其他两种算法进行了对比。其中 Popular 是指在满足边缘基站计算能力和缓存空间约束的前提下，尽可能将流行度高的服务进行边缘基站本地计算。Random 是指在满足上述两个约束下随机选择服务进行边缘基站本地计算。其中可以看到，本文所提出 DQN 算法具有最优的性能，且其收敛速度比 Popular 算法更快，这是因为 DQN 模型可以先通过离线的方式进行训练，在在线运行时就可以快速收敛。但 Popular 算法由于需要通过根据用户请求等信息逐步计算流行度，而当前期信息较少时计算出的流行度会变化很大，因此收敛速度较慢。随机算法由于每次随机选择服务，因此波动幅度始终较大。

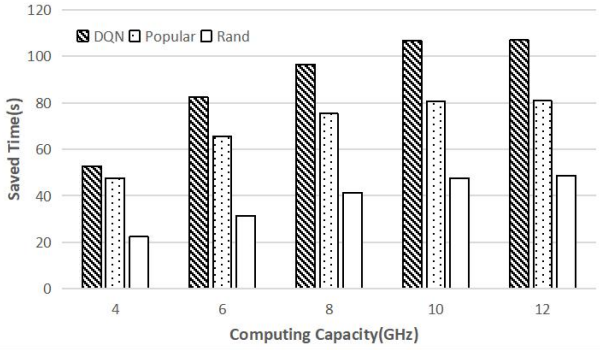

图 3 约束计算能力时不同算法的表现

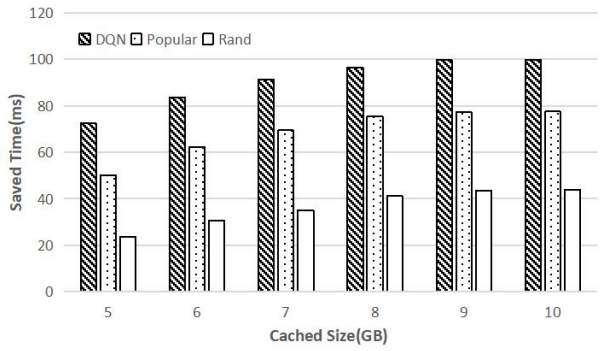

图 4 约束储存空间时不同算法的表现

图 3 和图 4 表现了三种算法在不同约束条件下的性能变化趋势。图 3 是不同计算能力下三种算法节约时间的变化，可以看到随着边缘基站计算能力的增强，算法节约时间也随之增加，但这种增加趋势不会一直进行下去，如图中当计算能力在 10GHz 和 12GHz 时算法节约时间基本一致，这是因为此时边缘基站的计算能力不再是算法性能的瓶颈，而是受到边缘基站缓存空间的限制，无法缓存更多的服务，从而导致算法性能无法进一步提升。另外，从图中可以看出，在计算能力较小时 Popular 算法性能提升较快，而随着计算能力增加，提升幅度迅速减小，如在 4GHz 到 6GHz 过程中性能幅度提升较大，而在 8GHz 到 10GHz 过程中幅度减小。这是因为 Popular 算法随计算能力增加后加入了一些流行度相对较低的服务，所以幅度相对减小，而所提出的 DQN 算法由于可以根据约束条件主动变换调整

边缘计算策略，如随着计算能力增加，缓存空间逐渐成为约束时，将原某些计算需求小缓存大的服务更换为计算需求大缓存小的服务，从而使得算法性能稳定上升。另外，由于随机算法的随机性，当缓存空间成为瓶颈前，其性能基本随计算能力的增加而线性增加。图 4 情况类似。综合两张图可以看出，单个约束条件对算法性能的影响会逐渐减小，只有同时增加两方面能力才能实现算法性能持续增加。

7 结束语

在边缘缓存决策问题中使用深度强化学习的目的是利用其在无未来信息和状态转移概率的情况下进行自动学习的能力，在未来数据未知的情况下训练随时间动态调整的模型。我们将问题建模成马尔可夫决策过程，并通过 DQN 算法进行解决。我们使用了两种对比算法去验证所提出方法的性能，仿真实验表明该方法在应对用户随机的请求状态时相较于使用单一参数的决策方法能降低更多的时间延迟。此外，在边缘服务器的约束条件发生变换时，基于深度强化学习的方法能够通过主动调整决策来获得稳定的性能提升。

致谢

该文章获得国家自然基金青年基金项目：62001057；以及北京邮电大学基本科研业务费项目：2021RC26 的支持，在此表示感谢

参考文献:

[1] F. Dobrian et al. 2011. Understanding the Impact of Video Quality on User Engagement. In *SIGCOMM*. ACM.

[2] S. S. Krishnan and R. K. Sitaraman. 2012. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-experimental Designs. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference (IMC)*. ACM.

[3] Stanney K M , Mourant R R , Kennedy R S . Human Factors Issues in Virtual Environments: A Review of the Literature[J]. Presence, 2006.

[4] Lin L, Liao X, Jin H, et al. Computation offloading toward edge computing[J]. Proceedings of the IEEE, 2019, 107(8): 1584-1607.

[5] Ghosh S, Agrawal D P. A high performance hierarchical caching framework for mobile edge computing environments[C]//2021 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2021: 1-6.

[6] Xia X, Chen F, He Q, et al. OL-MEDC: An Online Approach for Cost-effective Data Caching in Mobile Edge Computing Systems[J]. IEEE Transactions on Mobile Computing, 2021.

[7] C. Zhan and Z. Wen, “Content cache placement for scalable video in heterogeneous wireless network,” IEEE Communication Letter, vol. 21, no. 12, pp. 2714-2717, Dec. 2017.

[8] Y. Zhang, M. S. Hossain, A. Ghoneim and M. Guizani, “COCME: Content-Oriented Caching on the Mobile Edge for Wireless Communications,” IEEE Wireless Communications, vol. 26, no. 3, pp. 26-31, June 2019.

[9] Masood A, Nguyen T V, Cho S. Deep Regression Model for Videos Popularity Prediction in Mobile Edge Caching Networks[C]//2021 International Conference on Information Networking (ICOIN). IEEE, 2021: 291-294.

[10] Chakareski J. Viewport-adaptive scalable multi-user virtual reality mobile-edge streaming[J]. IEEE Transactions on Image Processing, 2020, 29: 6330-6342.