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(Deep Q-Learning)进行深度强化学习以生成视频资源缓存策略的方法。该方法针对当前全景视频中多比特率版本的情况，不仅仅考虑边缘缓存内容，同时将不同比特率版本间的转码时延考虑在内。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，结合模型中数据离散的特性，使用 DQN 算法对模型进行训练，在迭代中获取最优缓存策略。仿真结果表明，所提算法具有较高的收敛速度和最优的性能，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，为全景视频边缘缓存机制提供了切实可行的方案。

关　键　词：边缘计算；深度强化学习；缓存策略；全景视频

中图分类号：TP393　　**文献标识码**：A

An Optimization Scheme of Edge Caching for Panorama Video based on DQN

Abstract: To solve the edge caching problem of cloud server and edge server in panorama video service, optimizing the edge caching mechanism to reduce the time delay for obtaining video resources, a method of generating cache strategy by using DQN (Deep Q-Learning) as deep reinforcement learning algorithm is proposed. The method considers not only the edge cache content but also the transcoding latency between different bit-rate versions for the current situation of multiple bit-rate versions in panoramic video. 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 high convergence speed and the best performance. And when the constraints change, it can actively change the edge caching strategy to stably improve the performance of the algorithm, which provides a practical solution for the edge caching mechanism of panoramic video.

Key words: edge computing; deep reinforcement learning; cache strategy; panorama video

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。然而，全景视频快速发展的同时，用户对其服务质量的需求也在持续增长，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，传输全景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

议是 MPEG-DASH 流， DASH(Dynamic Adaptive Streaming over HTTP，基于 HTTP 传输的动态自适应流)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，会将低比特率版本的视频更换到高比特率版本，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。该模式下，边缘服务器虽然距离用户近，传输延迟低，不过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。

针对全景视频传输场景的特点，本文特别考虑到了其多比特率版本的资源结构，在缓存策略的制定上不只简单的考虑缓存什么内容，还考虑到不同比特率版本缓存内容间的转码特征。转码是多比特率视频资源特有的一种性质，即通过视频转码或者基于机器学习的视频增强技术，将指定视频的某一比特率版本转化为另一比特率版本。利用这种方式，边缘服务器可以在用户请求某内容未缓存比特率版本的时候，尝试采用转化该内容其他已缓存比特率版本的方式提供服务。值得注意的是，这种转换所需要消耗一定的计算时间，特别是低比特率版本转化为高比特率版本的过程。因此，考虑到全景视频传输场景的时延要求，结合不同比特率版本间的转码时间，决定在对应的时间段内每个边缘服务器缓存什么内容的何种比特率版本是本文主要研究的内容。

为此，本文通过深入分析上述研究内容，通过将其建模为无监督的感知决策问题，利用当前深度强化学习在该种问题的优势，同时考虑所建模型数据的离散性特征，提出了基于 DQN 算法的全景视频边缘缓存协作优化方案，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)，其中我们使用综合边缘服务器缓存状态和服务请求状态两方面来描述当前状态(state)，边缘服务器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，根据全景视频传输场景的特性，我们创新性地将不同比特率版本相互转码带来的计算时延加入到缓存决策考虑中，并基于 DQN 深度强化学习提出了智能化缓存决策算法从而实现了用户服务时延长时间平均值的最优化。

（3）我们模拟了变化的网络环境进行了实验，并使用了当今实际场景中部署最多的 Popular 算法和 Random 算法与我们提出的算法进行对比。同时，我们还对比了 2021 年 CCF A 类会议 ACM MM 中所提的关于该问题的多更新强化学习(MURL, Multi Update Reinforcement Learning)算法[14]。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，但 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

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。在[5]中，Ghosh 等人强调了流行度对于边缘缓存决策的重要意义，并且提出了一种可伸缩的分级缓存机制，可以智能地计算内容流行度并将内容缓存到最佳位置。文献中提到的算法是针对缓存的贪婪算法的变体，它对每个内

容进行流行度分析，并动态的维护缓存策略。算法为每个缓存对象进行评级，确保高访问次数和低缓存占用的内容获得更高的评级，而低访问次数和高缓存占用的内容则获得低评级。算法会在缓存决策中不断删除低等级的内容，并放到云服务器上，而对于始终保持高等级的内容则缓存在边缘服务器中。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。该算法的核心是将在线的移动边缘数据缓存(MEDC，Mobile Edge Data Caching)算法分隔成一些小的时间间隙执行，通过对边缘缓存内容的清洗动态地删除非流行内容。同时为了避免过高的缓存决策更新频率，在线的移动边缘数据缓存算法只在产生足够的收益时才更新其策略。论文[7]同样设计了一种基于内容的边缘缓存策略优化方法，其核心思路仍然是基于流行度信息在边缘服务器中缓存最热门的内容，并且考虑到计算资源的有限性和内容传输的时延要求，实现了两种简单的方法来完成这个目标，即潜在语义索引(LSI，Latent Semantic Indexing)和协同过滤(CF，Collaborative Filtering)。根据内容流行度设计的边缘缓存策略在思路上利于理解，但是需要对流行度有一个合适的分级，通过算法定义什么是高流行度的内容，并且在动态过程中最终收敛至最佳的边缘缓存决策。

而在流媒体服务的边缘缓存问题上，也有很多研究者针对视频传输服务的特点设计了一些边缘缓存策略优化方法。Zhan 等[8]针对异构无线网络中可拓展视频编码流的缓存策略问题，提出了基于凸规划松弛的启发式解法。文章中特别提到了分层视频传输的特点，即同一视频内容会编码成不同的质量等级，并且提到了在传输过程中，内容下载的延迟应该由用户请求的所有层的最大下载延迟决定。因此，在优化缓存策略的时候，具体应该考虑缓存哪些视频，以及应该缓存视频的哪一个质量等级的版本。文章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。在[9]中，Yu 等人研究了在没有视频内容流行度先验知识的边缘计算系统中如何制定最优的全景视频缓存策略，文中特别考虑了全景视频在边缘服务器中可能需要组合不同的视频切片从而组合成用户观看的全景视频，即视频切片在云服务器传输到边缘服务器的过程中可能是 2D 的而在边缘服务器中需要组合成 3D 视频，这个过程依赖边缘服务器的计算能力。因此该文章中同时考虑到了用户体验质量(QoE，Quality of Experience)和边缘服务器能耗，使用联合优化问题的思路，采取组合多臂老虎机(Combinatorial Multi-Armed Bandit)理论进行解决。

同时，许多学者也利用预测信息来辅助生成未来的视频内容缓存策略，Masood 等[10]提出了一种基于深度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。方案将算法的时隙分为两个阶段，在第一个阶段用户发送内容请求，边缘基站在接受请求后将内容发送给用户，并且会评估请求内容在本地的流行度，并与相邻的边缘设备进行信息共享。在第二个阶段边缘基站则通过深度回归模型，根据受欢迎程度的估计，选择下一个时间端的最佳缓存策略。文中特别提到，将深度学习方法应用于边缘计算网络中的缓存问题可以使得流行度的估计更加准确。此外，在全景视频传输和播放场景下，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[11]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，并依据其概率帮助边缘服务器决定缓存策略，作者特别提到了用户观看的视口区域的“贡献度”概念，即通过用户头盔反馈的视野数据，结合像素数量来计算每个切片对于视野区域的占比，占比高的视频内容被认为是高贡献度的，在算法中就会优先被缓存于边缘服务器中。这类方法充分利用了全景视频播放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。

然而现有的方案对于缓存策略的优化大部分是仅针对内容的，并没有利用到全景视频多比特率版本的特点，即全景视频的传输系统中不仅仅有视频内容的区分，同一视频内容还有不同的比特率等级可选。在传输过程中，边缘服务器的转码功能可以在未缓存指定内容的时候，通过转化同一视频其他比特率版本的资源，利用边缘服务器计算能力辅助提供服务，该特性是该问题场景下缓存策略的重要组成部分。另外，一些方案的优化目标是短时期内的传输时延，并未考虑长时间整体平均时延的优化目标，而本文的方案中不仅考虑到了同一视频的不同比特率版本互相转化的功能，而且专注于长时间平均时延优化目标，这些创新对于全景视频的传输系统中边缘缓存策略制定有着重要的意义。

3 系统架构

3.1 网络模型

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K_n。对于所有的视频资源K_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$K_n(L_m)$。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。如果边缘服务器缓存了用户请求视频的另一种比特率视频等级，则边缘服务器可以利用其计算能力进行视频转码，将已缓存的另一比特率等级转码成用户所请求的比特率等级版本，然而这种方式同样需要转码时间。例如图 3-1 中，E_1中缓存了视频K_1的L_3比特率等级的内容，对于用户U_1来说，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$K_3(L_2)$，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。对于用户U_3来说，其请求的具体内容是视频K_1的L_2比特率版本，而视频K_1有其他的比特率版本L_3缓存于边缘服务器中，因此边缘服务器E_1可以对$K_1(L_3)$进行转码操作，转码至用户所请求的内容版本$K_1(L_2)$，然后传输给用户。

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$K_2(L3)$，$K_3(L_1)$和$K_3(L_2)$，但是实际上这其中只有内容$K_3(L_2)$被用户U_5所请求，因此剩下的两个内容$K_2(L3)$和$K_3(L_1)$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。

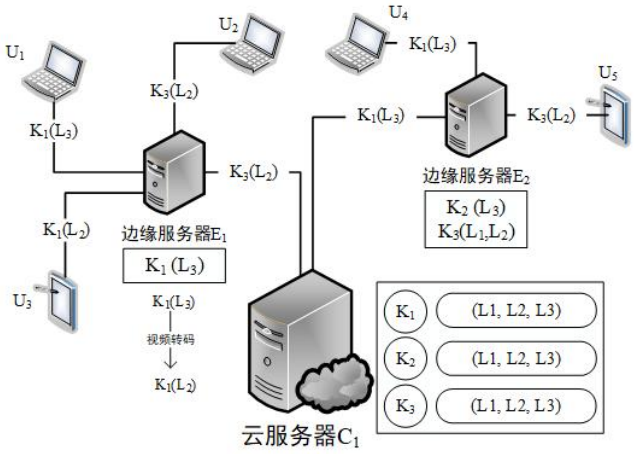

图 3-1 全景视频传输系统架构

3.2 问题形式化

在该网络系统中，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服务器应该缓存什么视频的什么比特率版本内容才能使得总获取时延最小（或节约的总时间最大）。

由于边缘服务器具备计算能力，而同一内容的某个版本可以通过计算转化为其他码率版本。因此，为了充分利用边缘节点的缓存空间，如果边缘服务器已经缓存了视频k的某一码率版本，当用户请求其他码率版本时可以通过边缘服务的计算实现转化。这个转化根据其源版本和目标版本的不同会需要不同的计算时间，由高码率版本转化为低码率版本所需要的时间更少，而从低码率版本到高码率版本会需要更多的计算时间。定义参数$tc_{n,m}^{k}$表示对于视频k而言，从码率版本L_n转化为码率版本L_m需要的计算时间，特别的当n与m相同时，计算时延$tc_{n,m}^{k}=0$。用户从云服务器请求视频k中L_m版本的

服务总时延为T_m^k， 用户从边缘服务器请求视频k的L_m版本的传输时延定义为tt_m^k，则当用户请求视频k中L_m版本时，若边缘服务器已经缓存了视频k中的L_n版本，则总的服务时延为$t_m^k = tc_{n,m}^k + tt_m^k$；若边缘服务器没有缓存视频$k$的任何版本，则需要先从云服务下载所请求的视频$k$的指定版本，下载时延为$td_m^k$，此时总的服务时延为$t_m^k = tt_m^k + td_n^k$。因此，与从云服务器请求内容相比，从边缘服务器提供服务节省的时间为$T_m^k - t_m^k$。从而建立优化模型为：

$$\max \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} r_{k,m}^t (T_m^k - t_m^k) \tag{1}$$

$$s.t. \sum_{k\in K} c_{k,m} x_{k,m}^t \leq C \tag{2}$$

其中$r_{k,m}^t$代表t时刻用户对视频k比特率等级m的内容的请求次数，$c_{k,m}$代表缓存视频k比特率等级m的内容需要的缓存空间，$x_{k,m}^t$为边缘服务器当前的缓存状态，C代表边缘服务器的缓存空间。

要解决上述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。

而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，我们采用深度强化学习的思路对问题进行求解。首先，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 马尔科夫决策过程建模

4.1 状态空间模型

状态(State)是描述网络系统环境。在该问题中，我们综合边缘服务器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$X^t = [x_{k,m}^t]$代表t时刻边缘服务器的缓存状态，其中$x_{k,m}^t = 1$代表在t时刻缓存视频k的m比特率等级内容，$x_{k,m}^t = 0$代表t时刻不缓存视频k的m比特率等级内容。由于边缘服务器的缓存空间有限，因此应满足(3)中的约束条件：

$$\sum_{k\in K} c_{k,m} x_{k,m}^t \leq C \tag{3}$$

其中$c_{k,m}$代表缓存视频k的m比特率等级需要的缓存空间，K代表所有的视频资源集合，C代表边缘服务器的缓存空间。

我们用向量$R^t = [r_{k,m}^t]$代表t时刻边缘服务器的用户请求状态，$r_{k,m}^t$代表t时刻用户对视频k的m比特率等级内容的请求次数。

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 动作空间模型

在该内容缓存问题中，边缘服务器进行的缓存状态的变化即为动作，我们用向量$A^t = [a_{k,m}^t]$表示，其中$a_{k,m}^t$代表t时刻边缘服务器对视频k的m比特率等级内容采取的动作，具体来说，

$a_{k,m}^t \in \{-1,0,1\}$。当$a_{k,m}^t = -1$代表边缘服务器移除视频k的m比特率等级内容，即$t-1$时刻时缓存有视频k的m比特率等级内容，但t时刻不再缓存；$a_{k,m}^t = 1$代表边缘服务器下载缓存视频k的m比特率等级内容，即$t-1$时刻时没有缓存视频k的m比特率等级内容，但t时刻进行缓存；$a_{k,m}^t = 0$代表边缘服务器不采取任何动作，即视频k的m比特率等级内容在$t-1$时刻和t时刻的缓存状态相同。动作和缓存状态间存在(4)中所示关系：

$$X_{k,m}^{t-1} + A_{k,m}^t \to X_{k,m}^t \ with\ probability\ 1 \quad (4)$$

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$t-1$时刻边缘服务器缓存了视频k的m比特率等级内容，则$a_{k,m}^t = 1$即为无效动作，因为这是重复下载，无意义；或$t-1$时刻边缘服务器没有缓存视频k的m比特率等级内容，则$a_{k,m}^t = -1$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 回报函数构建

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，我们将节约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$a_{k,m}^t$的总和，进行如下定义：

$F^t = \sum_{k \in K} f_{k,m}^t$其中$f_{k,m}^t$代表分量动作$a_{k,m}^t$带来的回报，定义如(5)中所示：

$$f_{k,m}^t = \begin{cases} -\infty, \ if\ A^t\ \ is\ \ illegal \\ r_{k,m}^t(T_m^k - tt_m^k - td_m^k), \ if\ \ a_{k,m}^t = 1 \\ x_{k,n}^t r_{k,m}^t(T_m^k - tc_{n,m}^k - tt_m^k), if\ \ a_{k,m}^t = -1 \\ x_{k,m}^{t-1} r_{k,m}^t(T_m^k - tt_m^k), \ if\ \ a_{k,m}^t = 0 \end{cases} \quad (5)$$

其中$r_{k,m}^t$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$a_{k,m}^t = 1$时，边缘服务器的动作为先从远端服务器下载内容再进行传输，总时延为$tt_m^k + td_m^k$，而云计算的时延为T_m^k，则总节约时间为$r_{k,m}^t(T_m^k - tt_m^k - td_m^k)$；当$a_{k,m}^t = -1$时，边缘服务器移除视频$k$的$m$比特率等级内容，也就意味着边缘服务器中没有缓存内容k的m比特率等级内容，此时若存在视频k的n比特率等级内容，即当$x_{k,n}^t = 1$时，则可以通过计算和转码进行转化，此时由边缘服务器提供服务的时延为$tc_{n,m}^k + tt_m^k$，即转码时间和边缘服务器传输给用户时间的总和，总节约时间为$x_{k,n}^t r_{k,m}^t(T_m^k - tc_{n,m}^k - tt_m^k)$。对应的，当此时边缘服务器中没有缓存视频$k$的任一比特率等级内容，即$x_{k,n}^t = 0$时，总节约时间则为0；当$a_{k,m}^t = 0$时，表明边缘服务器不对视频$k$的$n$比特率等级内容做任何操作，此时总节约时间则与$t-1$时刻的缓存状态有关，若$t-1$时刻缓存了内容$k$的$m$比特率等级内容，即$x_{k,m}^{t-1} = 1$，边缘服务器可以直接向用户传输所请求内容，则总节约时间为$r_{k,m}^t(T_m^k - tt_m^k)$，否则如果若$t-1$时刻没有缓存视频$k$的$m$比特率等级，边缘服务器需要从云服务器请求内容$k$的$m$比特率等级内容，此时总节约时间为0，综合两种情

况即为$x_{k,m}^{t-1}r_{k,m}^{t}(T_m^k-tt_m^k)$。

5 基于深度强化学习的解决方法

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。

深度强化学习算法根据选取动作的策略不同可以分为基于策略的强化学习(Policy-Based RL)和基于价值的强化学习(Value-Based RL)，在基于策略的强化学习中所有可能动作都有一定概率被选中，只是不同动作有不同的概率。而在基于价值的强化学习中则会选用某一时间反馈值最高的动作，因此基于价值的强化学习适合用于离散的动作空间。除了本文选用的 DQN，其他的深度强化学习算法还包括 A2C、A3C[12]、DDPG[13]等，然而这些算法是基于策略的强化学习，本身针对的问题往往有连续的动作空间，本文提出的动作空间是离散的，因为边缘服务器只会进行有限类型的操作，即添加缓存、清除缓存和保持不变三种动作。尽管前面提到的三种算法可以更改成离散动作空间的算法，但是这会对其准确性造成影响。另外，选用更为简单的 DQN 算法更容易部署，时间延迟也更低。综合考虑后本文选用了 DQN 作为解决建模问题的强化学习算法以支持离散的动作空间。

5.1 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(6)中所示的定义：

$$UCB(S,A)=Q(S,A)+\sqrt{\frac{2\ln n}{n_j}} \tag{6}$$

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(7)中所示：

$$loss=\frac{[Q_{target}(S^t,A^t)-Q(S^t-A^t)]^2}{2} \tag{7}$$

其中目标价值为下一个状态的最大Q值，如公式(8)中所示：

$$Q_{target}(S^t,A^t)=F^t+\gamma\max_{A'}Q(S^{t+1},A') \tag{8}$$

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，依据是选取使得$UCB(S,A)$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，每种服务内容有 3 种不同码率，首先限制边缘服务器的计算能力为 8GHz，缓存空间为 8GB，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服务器约束条件改变的情况下的表现，保持缓存空间为 8GB 不变的情况下调整其计算能力为【4GHz, 6GHz, 8GHz, 10GHz, 12GHz】，然后保持计算能力为 8GHz 不变的情况下，调整其缓存空间为【5GB, 6GB, 7GB, 8GB, 9GB, 10GB】

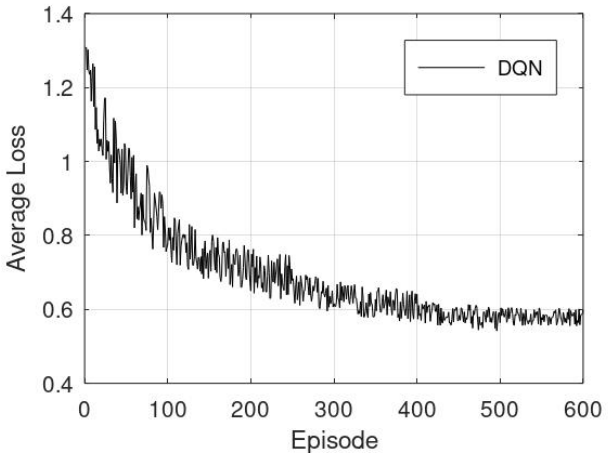

图 6-1 DQN 算法训练过程中的 Average Loss

图 6-1 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。

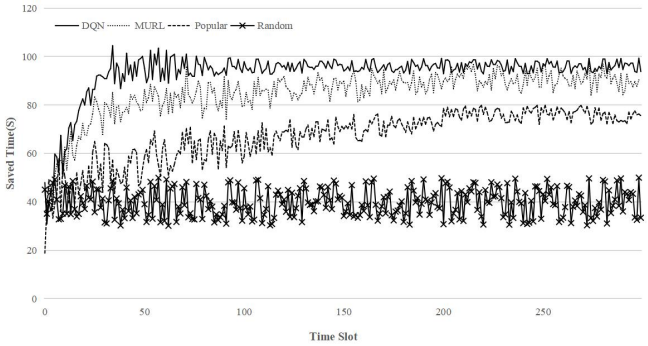

图 6-2 不同算法下节约的时间

图 6-2 显示了算法在性能（即通过边缘计算节约的总时间）方面的表现，并与其他三种算法进行了对比。其中 Popular 是指在满足边缘服务器计算能力和缓存空间约束的前提下，尽可能将流行度高的内容缓存在边缘服务器。Random 是指在满足上述两个约束下随机选择内容进行缓存。另外，我们实现了[14]中提出的多更新强化学习算法(MURL)，该算法同样应用于边缘缓存卸载问题，算法考虑了边缘节点的计算能力和缓存能力限制，但是并未考虑针对全景视频资源结构的转码过程。在图 6-2 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，和 MURL 算法收敛速度相差不大，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。在模型收敛后，DQN 模型的性能略高于 MURL 算法。

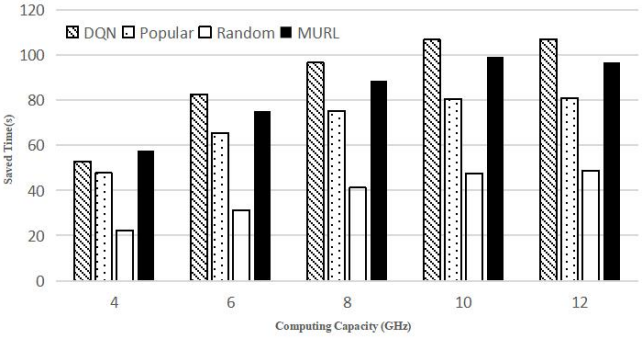

图 6-3 约束计算能力时不同算法的表现

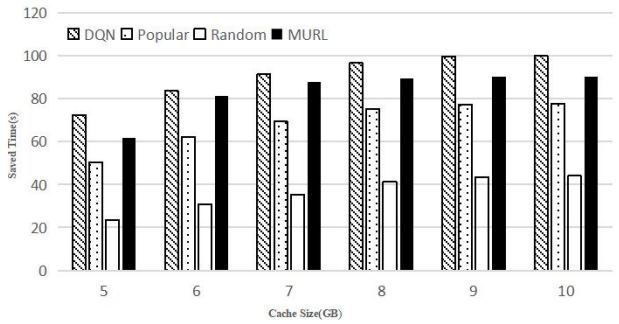

图 6-4 约束储存空间时不同算法的表现

图 6-3 和图 6-4 表现了三种算法在不同约束条件下的性能变化趋势。图 6-3 是不同计算能力下四种算法节约时间的变化，可以看到随着边缘服务器计算能力的增强，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，这是因为此时边缘服务器的计算能力不再是算法性能的瓶颈，而是受到边缘服务器缓存空间的限制，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，这一点与 MURL 算法相同。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。与 MURL 相比，DQN 算法的性能略高于 MURL，这是由于本文提出的 DQN 算法额外考虑到了不同比特率版本之间的转码过程，在这种特殊场景下，我们提出的 DQN 算法更贴合于实验设计的全景视频传输模型，而 MURL 算法由于没有考虑边缘服务器的转码功能，在这种细化场景下的决策不占优势。图 6-4 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。

为了证明同时增加计算能力和缓存能力可以实现算法性能的持续增加，我们分别控制计算能力和缓存能力为【4GHz，4GB】,【6GHz，6GB】,【8GHz，8GB】,【10GHz，10GB】和【12GHz，12GB】共五组情况，同时分别测试四种算法在每种情况下的总节约时间，结果如图 6-5 中所示，当同时增加两种能力时，四种算法节约时间的增加基本随两种能力线性增加，并且对比图 6-3 和图 6-4 可以发现其性能优于同样情况下限制其中一种能力的场景，例如当限制缓存能力为 8GB 而将边缘服务器的计算能力提升到 12GHz 的时候，DQN 算法节约的时间是 106.9 秒，而当同时增加计算能力和缓存能力至【12GHz，12GB】时，DQN 算法节约的时间是 118.5 秒。其中我们还发现 DQN 算法的性能在高性能的情况中表现略优于 MURL 算法，这进一步证明了在特殊的问题场景下，即考虑不同比特率等级视频的互相转码时，我们提出的算法的性能优于 MURL 算法。

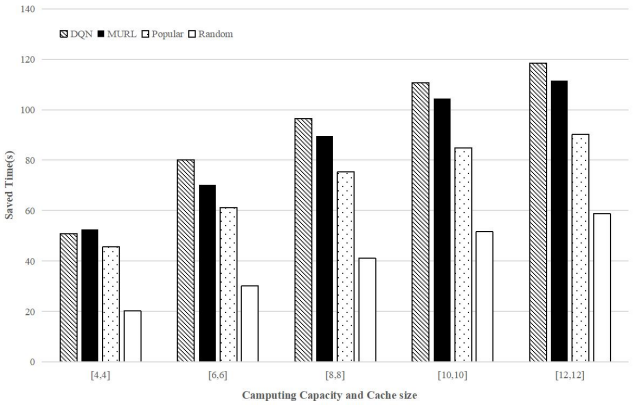

图 6-5 同时增加计算能力和缓存能力时不同算法的表现

7 结束语

在全景视频的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。我们特别考虑到了全景视频多比特率等级的特性，将边缘服务器中进行的转码过程加入到问题建模中。我们使用了三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。同时由于考虑了转码能力，在本文提出的全景视频传输场景中我们提出的算法性能略优于不考虑转码的 MURL 算法。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

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

参考文献：

[1] Dobrian F , Sekar V , Awan A , et al. Understanding the Impact of Video Quality on User Engagement[C]// Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Toronto, ON, Canada, August 15-19, 2011. ACM, 2011.

[2] Krishnan S S , Sitaraman R K . Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs[J]. IEEE/ACM Transactions on Networking, 2013, 21(6):2001-2014.

[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] 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.

[8] 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.

[9] Z. Yu, J. Liu, S. Liu and Q. Yang, "Co-Optimizing Latency and Energy with Learning Based 360° Video Edge Caching Policy," 2022 IEEE Wireless Communications and Networking Conference (WCNC), 2022, pp. 2262-2267.

[10] 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.

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

[12] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[C]//International conference on machine learning. PMLR, 2016: 1928-1937.

[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] Hao H, Xu C, Zhong L, et al. A multi-update deep reinforcement learning algorithm for edge computing service offloading[C]//Proceedings of the 28th ACM International Conference on Multimedia. 2020: 3256-3264.