A Transcoding Task Offloading and Routing Decision-Making Scheme in Live Transmission Architecture Based on Computing Power Network

Zitong Li[1], Shuai Peng[1], Han Xiao[1], Shujie Yang[1] and Changqiao Xu[1]
[1]State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China

In recent years, there has been a significant increase in the demand for high-bit-rate live broadcast services, which has led to the widespread use of edge transcoding technology. Edge transcoding can effectively reduce the throughput of streaming media transmission, making it a popular and extensively researched technology. However, due to the real-time requirements of live broadcasting, the edge server needs to have the sufficient computing power to ensure low-latency calculations, which makes computing power allocation and traffic distribution become quite difficult. Inspired by the real-time and flexible computing power scheduling ability of the Computing Power Network, this paper explores reasonable edge task offloading and efficient traffic routing path planning to ensure overall low latency. This paper proposes a live stream transmission architecture based on the computing power network to solve the problems mentioned above to some degree. The paper first models the computing power network in the scene and then designs a task offloading algorithm based on Deep Reinforcement Learning (DQN) to determine the device for executing the computing task. Furthermore, a hybrid Simulated Annealing Genetic Algorithm (SAGA) is proposed for routing decisions. The effectiveness and superiority of the scheme are validated through simulation experiments.

***Index Terms*—Live streaming architecture, computing power network, DQN-based task offloading, routing path decision, simulated annealing genetic algorithm.**

I. Introduction

THE rapid development of emerging business requirements such as the Industrial Internet, Internet of vehicles, Autonomous Driving, and Live Video requires the network to have high caching and computing capabilities to ensure normal operation and improve user experience quality. And due to the explosive growth in data traffic, particularly video traffic [1], the network has been faced with significant challenges. It is required that the network can provide a higher data transmission rate and lower network delay, to provide users with a better experience quality.

In order to deal with the challenges in the network, it is necessary to solve them in terms of computing services and transmission forms. At present, various forms of computing emerge endlessly, such as cloud computing deployed in the data center, edge computing deployed near the user side etc. [2]. Although the birth of edge computing solves the low-latency requirements of new services and reduces traffic congestion in the network, computing nodes of various computing forms are spread all over the network, lacking unified scheduling management and utilization [3]. A major challenge is how to integrate computing and network resources by connecting and managing a large number of idle resources through the network, enabling collaborative scheduling of multi-level resource nodes and flexible deployment of applications. And improving network performance, reducing transmission delay, and providing users with a consistent service experience have become crucial. At present, various transmission methods are mainly developed for load balancing and reducing network congestion. How to formulate relevant routing and forwarding strategies in combination with distributed computing resources on the network [4] is also an urgent research problem to be solved.

For the field of live video, platforms such as Douyin, Taobao, and Douyu now have very mature application-level client apps to provide users with the convenience of watching live content. The types of live broadcast services are rich and diverse, such as life, shopping, sports, music, etc. Compared with on-demand forms, live broadcasts provide users with the possibility of real-time interaction. Consequently, live broadcast services are growing rapidly in popularity, as they cater to people's increasing spiritual and cultural needs.

The network transmission architecture for live broadcasting is also constantly being proposed and updated. In recent years, the mainstream live video transmission architecture is HIER [5], its main process is to push the live stream to a centralized data center server for the host, and then the server pushes the live stream to the viewers who subscribe to this channel. However, this solution has some obvious disadvantages in actual conditions. For instance, the centralized streaming media server may be overloaded, resulting in significant wastage of back-to-source bandwidth from clients to the server. Furthermore, the routing topology may be inflexible and ill-suited to heterogeneous and dynamic network environments. Recently, Alibaba's research team proposed a new live broadcast transmission architecture for Taobao's live broadcast business, called Livenet [5]. Livenet implements a Content Delivery Network (CDN) server node coverage plane, and the anchor pushes the stream to the edge CDN server. Then Livenet proposed a concept of Streaming Brain, which is a centralized control platform that will obtain the load

Manuscript received December 30, 2022; revised March 22, 2023. Corresponding author: Shujie Yang (email: sjyang@bupt.edu.cn).

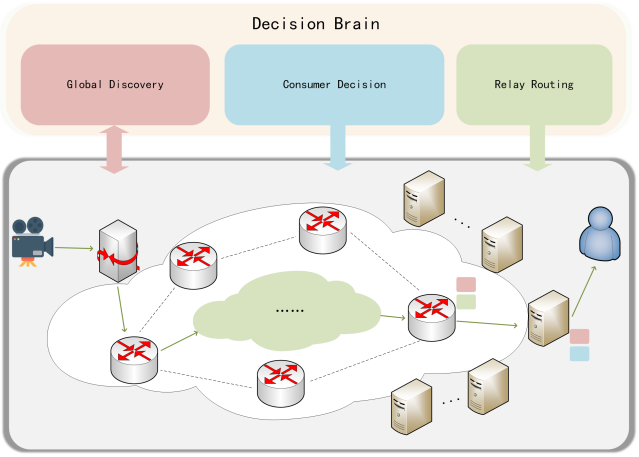

Fig. 1. Live video transmission architecture based on computing power network

status of network links and network nodes, then decide the video transmission routing strategy. In addition, the Twitch live broadcast platform proposes an Intelligent live broadcast platform [6], which deploys data centers in a distributed manner, uses access points to detect network link status information, tracks the resource dynamics of each source station in real-time, furthermore decides on the routing path and the size of the forwarded traffic.

The existing live broadcast frameworks primarily concentrate on traffic and bandwidth allocation, while still utilizing traditional scheduling algorithms. However, it lacks a comprehensive approach to scheduling and allocation of computing resources within the network, making it challenging to satisfy the low-latency computing requirements of live broadcasting. To address this issue and meet the demands for low-latency transcoding in complex network environments during live broadcasting, we first propose a live video transmission architecture based on computing power network in this paper. Here we have two parts: Decision Brain and cloud network equipment network. Among them, Decision Brain is a distributed data center responsible for macro-control and resource allocation. As shown in x. 1, there are three sub-modules in the Decision Brain module that work together. First, the Global Discovery module obtains the bandwidth resources and computing resources of each node in the network scene and then passes this information to the Consumer Decision module. We call the device that is the last mile away from the user a consumer, which can be served by a CDN server with computing resources.

This paper proposes a live broadcast transmission architecture based on computing power network, and proposes a novel scheduling and routing algorithm under this architecture and its main contributions are summarized as follows:

1) First of all, we propose a set of live streaming media transmission framework based on the computing power network, and innovatively propose a centralized decision-making module to macro-control the flow and the selection of computing tasks. Compared with the traditional live stream framework, there is a more flexible way to schedule live streams.
2) Then we proposed the DQN-based Task Offloading Algorithm decision-making calculation task execution object, and the execution of calculation tasks by this object can maximize the benefits of the overall architecture, that is, the calculation and transmission delay of live streams are minimized.
3) After determining the two endpoints of the routing path, we propose to use a hybrid Simulating Annealing Genetic Algorithm to make routing decisions, and determine the routing path with the smallest transmission delay.
4) We designed a series of simulation experiments to verify the superiority of our proposed framework and algorithm. The overall utilization of computing resources has been significantly improved and the user delay has been significantly reduced.

This paper is organized as follows. Section II discusses related works. Section III introduces the system model and section IV formalizes the problem. Section V introduces the computing power distribution phase and corresponding algorithm. Section VI introduces the routing decision phase and proposes the hybrid genetic algorithm. Section VII presents and discusses the simulation results and section VIII draws conclusions.

II. Related Work

Live content delivery has attracted the attention of many related researchers due to its wide range of applications. This section provides an overview of related work and explains how the work presented in this paper differs.

A. Edge Computing and Computing Power Network

At present, both academia and industry are actively promoting research and development in this field for the deep integration of computing and networks. For the industry, major operators and network equipment manufacturers are promoting the formulation of Compute First Networking (CFN) standards and the formation of technical solutions [7][8], and have released several CFN-related Technical white paper. In terms of standardization, the Internet Engineering Task Force (IETF) established the Computing in the Networking Research Group (COINRG), which is dedicated to research on the integrated use of computing and network technologies, so as to improve the performance, flexibility, security and privacy of networks and applications. And a related forum has been set up for discussion and research. In academia, major universities and research institutes have also put forward their own plans and ideas for this problem. In the literature [9], for the streaming service of the image business, the researchers combined Dispersed Computing (DC) to perform calculation processing during transmission in the network. In the literature [10], the joint scheduling and allocation of multi-dimensional resources is mainly aimed at the integrated architecture of network, computing and storage, so as to improve the network transmission and processing capabilities. In literature [11]-[16], on the basis of edge computing, by formulating corresponding

caching strategies to process video stream data, the delay of computing and services is reduced.

In live broadcast scenarios, video transcoding tasks must be executed in real-time. However, current computing offloading solutions lack the ability to dynamically adapt to real-time live streaming. Therefore, this paper proposes an edge computing task offloading strategy based on a computing power network to achieve real-time scheduling of dynamic computing power. Specifically, a task offloading algorithm based on Deep Q-Network (DQN) [17] is designed. The algorithm utilizes DQN to determine the optimal equipment for performing computing tasks and then employs a dynamic adaptive mechanism to allocate specific computing resources. Further details of the algorithm will be presented in Section V.

B. Routing Strategy Based on Computing Power Network

At present, various routing strategies on the network are relatively mature, whose main aim is increasing the data transmission rate and reducing the transmission delay overhead. There are research on routing and forwarding strategies based on CDN. And an ant colony routing strategy is proposed for decision-making based on the uniformity of distribution [18]. There are research on routing and forwarding strategies based on Named Data Networking (NDN), and the formulation and research of routing strategies for energy consumption in wireless networks [19]. At present, the research on the computing power network is mainly on the scheduling and allocation of resources. The research on the routing and forwarding strategy of the computing power network is currently in its infancy and needs to be analyzed in combination with the specific resource scheduling situation. Moreover, the design of routing strategies for live-streaming scenarios is still relatively rare.

This paper designs a hybrid simulated annealing genetic algorithm (SAGA) to solve the routing problem of live streaming media in this computing power network scenario. We model the routing path scheme as an individual in the genetic algorithm, and design a fitness function based on streaming media transmission delay to screen individuals, and combine the annealing algorithm to obtain the optimal routing scheme. Section VI details the detailed algorithm design.

III. System Model

A. Network Model

The main network elements in this scenario include broadcasters, cloud servers, edge servers, terminal devices, users (i.e. viewers) and relay networks. The basic network architecture is shown in Fig. 1. The broadcaster pushes the video stream to an access point which is a device connected to the relay network. The devices in this relay network are relays devices such as routers which have a flexible topology. Through the relay network, the video stream can be forwarded to the edge servers which are close to viewers. These servers can process the stream and then deliver it to the users. It is worth noticing that the procedure of video streaming is transcoding which means reducing high-bitrate video to low-bitrate video. Later in this subsection, we give a mathematical description of the elements above. The set of broadcasters(i.e. anchors) is represented as $\mathcal{A} = \{1, 2, ...a, ..., A\}$. The access points which receive the streams are named producers. The set of producers which receive the streams from broadcasters is denoted as $\mathcal{P} = \{P_1, P_2, ...P_a, ..., P_A\}$. These producers forward the video streams to the next hop which is included in the relay network. The devices included in the relay network are relay devices. The set of relays is denoted as $\mathcal{O} = \{1, 2, ...o, ..., O\}$. Through the relay network, the edge servers which are considered to be consumers receive streams and process them. These consumers are denoted as $\mathcal{C} = \{1, 2, ...c, ..., C\}$. The set of Viewers is represented as $\mathcal{V} = \{1, 2, ...v, ..., V\}$. In the real scenario, multiple consumers with similar transmission delays can provide services for a certain user. We define these alternative consumer devices with similar transmission delays as $\mathcal{C}_v = \{C_1^v, C_2^v, ...C_i^v, ...\}$ where $v \in \mathcal{V}$. And we define the number of consumers which can serve view v as $|C_v|$.

B. Resource Module

In this network architecture, the resources mainly includes bandwidth and computing power resources. In this live streaming media transmission scenario, bandwidth resources are mainly considered in the relay devices O, and computing resources are mainly considered in the consumer devices C.

1) Bandwidth Module

The delay devices in the relay network have a flexible topology, which is similar to a software-defined network. Therefore, how to decide the path of media traffic is a problem that needs to be solved. We take the bandwidth condition of the link between different $\mathcal{O}$ into consideration. To this end, in the following part of this section, we give a mathematical description of the bandwidth condition of relays in the relay network. The actual bandwidth is expressed as the transmission rate. We define the bandwidth resources which will be used at the time $t+1$ [1]as b_{mn} between node m and $n(m, n \in \mathcal{O})$. And the predictable link bandwidth which is already occupied between two nodes expressed as B'_{mn}. It is worth noting that b_{mn} is decided by the sender of the media stream(i.e. the broadcaster). Besides the link bandwidth bottleneck is mathematically denoted as B_{mn}. Obviously, $B'_{mn} <= B_{mn}$ hengs established.

2) Computing Power Resource

In this live streaming media transmission scenario, the computing task is video transcoding which converts the high bitrate(e.g. 1080p) to low bitrate(e.g. 720p) to dynamically adapt to real network conditions. It is worth noting that whether to perform the calculation task of transcoding depends on the real-time bit rate adaptive decision of clients. In the following part of this section, a mathematical description of computing power resources of consumer devices $\mathcal{C}$. We define computing resource pool as

$$r_v = \{r_1, r_2, ...r_i, ...r_{|C_v|}\} \tag{1}$$

[1]we define t as the current unit period and $t+1$ as the next unit moment.

Where r_v denotes the set of resource pools of viewers v, $i \in \mathcal{C}_v$. And r_i represents the resource occupied in resource pool i as shown in equation (2).

$$r_i = \{r\_CPU_i, r\_RAM_i, r\_HD_i\} \tag{2}$$

Where r_CPU_i, r_RAM_i and r_HD_i represent the CPU, RAM and hard disk respectively. The computing power of CPU can be denoted in (3).

$$r\_CPU_i = N_{core} \times Fre_{core} \times FP \tag{3}$$

The N_{core} represents the number of cores of CPU, Fre_{core} denotes the frequency of a single core, and FP denotes the floating point calculation value of a single cycle. Here we simply define the amount of data to be processed in the unit period as $data_i$. Sometimes the memory space of the resource pool cannot accommodate all data, which means $r\_RAM_i \leq data_i$. Thus $data_i$ may be stored in the hard disk, and then replaced in memory after the original data has been finished processing. It is worth noting that this computing resource allocation will be dynamically regulated based on the volume of $data_i$.

Similarly, we define the occupied computing resources at this unit time as $R'_v = \{R'_1, R'_2, ...R'_i, ...R'_{|C_v|}\}$. Besides the upper limit of computing resources of devices is mathematically represented as $R_v = \{R_1, R_2, ...R_i, ...R_{|C_v|}\}$. In the real scene, there is a pre-defined target of the utilization u (default 80%) of computing power resources to guarantee the ability to handle surges in computing tasks.

C. Content Model

In the living media stream, broadcasters push their content to viewers who subscribe these channels. The mainstream streaming method is using the Real Time Messaging Protocol(e.i. RTMP) [20] for broadcasters. RTMP is a TCP-based low-latency transmission protocol which is suitable for live broadcasting. In practice, video is transmitted and processed in the form of chunks, which has great benefits. Taking advantage of this feature, we use chunk as the smallest data execution and decision-making unit in this paper. As for the bit rate level of video, broadcasters push streams at certain bit rate based on their own device configuration. The following part of this sectioin, we describe the video content in a mathematical way. Some standard bit rate levels are set here $\mathcal{D} = \{1, 2, ..., d, ..., D\}$, which means the video exist in different versions. The set of chunks of a video stream at d bit rate level is denoted as $\mathcal{G}_d = \{1, 2, ..., g, ..., G\}$(the g-th video chunk is denoted as $\mathcal{G}_{d,g}$).

IV. Problem Formulation

First, in this section, the content delivery process is summarized as follows. When a viewer tries to subscribe to a video stream, firstly the decision brain confirms the information of this stream(e.g. bitrate level and the amount of data in a video chunk). Secondly, according to the resource usage conditions of consumers, the decision brain makes the decision on which consumer serves that viewer from the set of $\mathcal{C}_v$. Thirdly the decision brain makes a routing plan between the producer and consumer which is decided before. As for the specific delivery path, at time t, the g-th video chunk at d bitrate level $\mathcal{G}_{d,g}$ begins to be a push from the broadcaster to the consumer C_i^v and the viewer v. Then this $\mathcal{G}_{d,g}$ goes through the P_b into the relay network. Through the set of $\mathcal{O}^{C_i} = \{1, 2, ..., o, ...\}$ in which the $\mathcal{O}_1^{C_i}$ is the first relay device in the routing path, this $\mathcal{G}_{d,g}$ achieves the consumer C_i^v and execute transcoding task if necessary. Finally C_i^v transfer $\mathcal{G}_{d,g}$ to v. Next, we discuss the specific optimization goals.

A. Optimization goals in Consumer

As mentioned above, the video streams will be transcoded by the consumers if decided to be necessary. Therefore, we propose two optimization objectives for consumer devices.

1) Computing Resource Utilization

The computing power resource utilization is a ratio of the computing resources occupied at a certain moment to the resources owned by the consumer device. In the real-time transmission of streams, computing tasks are always changing dynamically. Therefore, in order to prevent the computing power of a certain node from being overloaded(computing tasks performed exceed the computing resources it has), the decision-making center needs to allocate computing tasks to different computing devices in real-time. In this process, the average computing resource utilization of all devices is a very useful indicator to measure the rationality and fairness of resource allocation. The average computing resource utilization can reflect the performance of the overall system to a certain extent.

The computing power resource utilization of consumer C_i can be denoted as

$$\mathcal{U}_i = \frac{R'\_CPU_i}{R\_CPU_i} \tag{4}$$

Since the device most directly involved in computing tasks is the CPU, we use the ratio of the occupied CPU to the total CPU to represent the computing resource utilization. In addition, it can be considered that the memory and hard disk mainly play an auxiliary role in computing tasks, so they can be ignored in the consideration of computing resource utilization without negatively affecting the overall performance. It is worth noting that in the same CPU of a device, the frequency Fre_{core} and the floating-point calculation value of a single cycle FP of different cores are equal. Thus $\frac{R'\_CPU_i}{R\_CPU_i}$ in the (4) can be simplified to $\frac{N'_{core,i}}{N_{core,i}}$.

We express the average computing power resource utilization in equation (5).

$$\mathcal{U}_{avg} = \frac{\sum_{i=1}^{|\mathcal{C}|} U_i}{|\mathcal{C}|} \tag{5}$$

2) Computing Latency

Computing latency is defined here. Transcoding latency is a factor that must be considered when consumer devices need to perform transcoding tasks. The time consumed by transcoding directly affects the waiting time for users to watch the live broadcast, which in turn affects the user's viewing experience. Obviously, in the case of limited hardware resources, the task

scheduling scheme with a lower transcoding delay is better. So we set the transcoding latency on the consumer server as one of the optimization goals.

We take a certain video chunk $\mathcal{G}_d$ at bitrate d as an example to describe the transcoding delay. According to the decision of the bit rate adaptive algorithm [21] on the user side, the consumer server converts the video chunk from the bit rate level d to the bit rate level d'. This process requires calculating the $Data_{i,dd'}$ amount of data and requires $N_{FLO,i,dd'}$ floating point operations at the consumer server i. Thus the processing time of the resource pool in i to process this task is represented in equation (6).

$$ct_i = \frac{N_{FLO,i,dd'}}{r\_CPU_i} \tag{6}$$

Sometimes, the memory space of the consumer device cannot accommodate all data at the same time, that is $r\_RAM_i < Data_{i,dd'}$. So $Data_{i,dd'}$ can be stored in the hard disk, and replaced in the memory when operation processing is required. The storage replacement time is denoted by equation (7).

$$st_i = (\frac{Data_{i,dd'}}{r\_RAM_i} - 1) \times t_{m,i} \tag{7}$$

Where $t_{m,i}$ represents a single replacement time in this consumer server i. Therefore, the overall processing time for a chunk $\mathcal{G}_d$ is expressed in equation (8).

$$ProcessTime(\mathcal{G}_d, r_i) = ct_i + a \cdot st_i \tag{8}$$

Where l is an indicator function to denote whether there is st time, that is, whether the data to be processed is larger than the size of the memory space or not.

$$a = \begin{cases} 0 & r\_RAM_i \geq Data_{i,dd'} \\ 1 & r\_RAM_i < Data_{i,dd'} \end{cases} \tag{9}$$

We define the average transcoding delay at some point the in equation (10).

$$ProcessTime_{avg} = \frac{\sum_{i=1}^{|\mathcal{C}|} ProcessTime(\mathcal{G}, r_i)}{|\mathcal{C}|} \tag{10}$$

In general, for consumer equipment, the optimization goal can be expressed in equation (11).

$$\begin{aligned} \min \quad & -\alpha \cdot \mathcal{U}_{avg} + \beta \cdot ProcessTime_{avg} \\ s.t. \quad & \forall i, \quad \mathcal{U}_i < u \end{aligned} \tag{11}$$

Where parameter α and parameter β represent the balance between the average computing resources utilization and the average process time in consumers. Simply, we set α and β to represent the normalized ratio of two dimension information in actual operation.

B. Optimization goals in Relay Network

After determining which consumer device serves the user, the decision brain decides how traffic in the relay network reaches the consumer server. It is equivalent to deciding the routing path in the determined producer node and consumer node. We hope that this routing decision can minimize the transmission delay in the relay network. The content transmission delay in the relay network is also a very important part of the user's viewing delay, which directly determines the viewing experience of the viewer. Therefore, we take this part of the transmission delay as the optimization target of the relay network.

The specific plan is as follows: the decision brain obtains the link traffic between relay device nodes in the relay network. On the premise that the link is not overloaded, the routing selection scheme is optimized to minimize the transmission delay in the relay network. The link bandwidth utilization between nodes m and n is expressed in equation (12).

$$U_{B,mn} = \frac{B'_{mn}}{B_{mn}} \tag{12}$$

Assume that the routing path of a certain media stream whose destination is consumer server i is determined as $\mathcal{O}^i = \{O_1^i, O_2^i, ..., O_o^i, ...\}$. Take a chunk $\mathcal{G}_d$ that contains $Data_{i,d}$ amount in this stream as an example, the transmission delay in link l_1^i from node O_1^i to node O_2^i is represented as $\frac{Data_{i,d}}{B_{l_1^i}}$ where $B_{l_1^i}$ is equivalent to $B_{O_1^i,O_2^i}$. Obviously, the l_j^i denotes the link from node O_j^i to node O_{j+1}^i. Further, we describe the transmission delay on this routing scheme $\mathcal{O}^i$ in equation (13).

$$TransmissionTime(\mathcal{G}_d, \mathcal{O}^i) = \sum_{j=1}^{|\mathcal{O}^i|} \frac{Data_{i,d}}{B_{l_j^i}} \tag{13}$$

In general, for the relay network, the optimization goal can be expressed in equation (14).

$$\begin{aligned} \min \quad & TransmissionTime \\ s.t. \quad & \forall j, \quad \mathcal{U}_{B,l_j} < 1 \end{aligned} \tag{14}$$

V. Computing Power Distribution Phase

This section introduces the computing power distribution phase. After the global discovery module obtains the consumer device information, the consumer decision module decides the most suitable consumer device for the user according to the streaming situation of the live broadcaster and the bandwidth of the last mile link. And allocate appropriate computing resources for transcoding tasks. To describe the dynamic process of computing resource allocation, we model it as a Markov Process. Besides, we deploy the DQN-based Computing Resource Allocation Algorithm in the Consumer Decision module.

A. Markov Decision Process Description of Consumer

The MDP for the consumer decision module can be described as four-tuple $\mathcal{M} = \{S, A, P, R\}$.

1) State of MDP

In the consumer, the states S of the MDP $\mathcal{M}$ is set to the computing resource status of each device plus the state information of the flow. The set of S is denoted as $S = \{r_1, r_2, ...r_i, ...r_{|C_v|}, Data_{dd'}\}$. Obviously, according to (1), the computing resource state set of the consumer device that provides services for a user is the first part of the MDP state. The second part is the amount of data to be processed

Algorithm 1: DQN-based Task Offloading Algorithm

Input: The computing power state of consumer devices $\{r_v\}$, the information of the video streamings $\{\mathcal{G}_d\}$

Output: The result of action decision a_t(e.i. the result of consumer decision module)

```
// Initialization
```
Initialize Q-function Q, target Q-function $Q' = Q$

```
// Execute Epoch
```
while *Epoch is not over* **do**
  for *For each time step* t **do**
    1) Given state s_t which means the computing power state of consumers, take action a_t which means the result of chosen consumer, based on π according to the equation (15);
    2) Obtain reward r_t according to the equation (17), and reach new state s_{t+1};
    3) Store (s_t, a_t, r_t, s_{t+1}) into buffer;
    4) Sample (s_i, a_i, r_i, s_{i+1}) from buffer;
    5) Target $y = r_i + \max_{\mathrm{a}} Q'(s_{i+1}, a)$;
    6) Update the parameters of Q to make $Q(s_i, a_i)$ close to y (regression);
    7) Every 5 steps reset $Q' = Q$
  end
end

for this stream. Based on these two parts of information, it can be effectively analyzed, and a reasonable computing task offloading decision can be made.

2) Action of MDP

A stands for action. In this MDP process, the action represents the specific device selection of the consumer decision module, that is, which edge device the actual calculation task of this flow is offloaded to. The set of action here is represented as $A = \{a_1, a_2, ...a_i, ..., a_{|C_v|}\}$, where $i \in \mathcal{C}_v$ stands for the i-th device, and the a_i denotes the action of i-th device. The $a_i \in \{0, 1\}$ is a binary number, 1 indicates that the device i executes the transcoding task, and 0 indicates not. Simply, the action dimension is $|C_v|$.

3) Transition Probability of MDP

P represents the state transition probability. We use $P(s'|s, a)$ to denote the probability that the current state s transitions to the next state s' given the action a is taken. In the consumer task decision-making scenario, this means the impact of the decision result of the target device based on the computing resource status of the network edge device and the size of the media stream data on the overall computing resource status and even traffic conditions.

4) Reward of MDP

R represents the reward, which means the benefits resulting from the above decisions. This reward is mainly to indicate the direction of MDP iterative development, which means that if the decision is incorrect or the result is not ideal, the reward value can be negative to gradually correct the wrong decision. According to (11), This reward can be set as $\max \alpha \cdot \mathcal{U}_{avg} - \beta \cdot ProcessTime_{avg}$.

The above four-tuple is also the basis of reinforcement learning DQN.

B. DQN-based Computing Power Distribution Algorithm

We deploy the DQN reinforcement learning algorithm in the consumer decision module. The centralized training feature of the DQN algorithm fits perfectly with the decision brain architecture of the computing power network. And as shown in the action of the MDP section above, the consumer action selection is discrete, which also provides convenience for the deployment of the DQN algorithm. Below we introduce the task offloading algorithm based on DQN.

1) DQN-based Task Offloading Algorithm

A deep Q-network(DQN) Algorithm is a kind of deep reinforcement learning algorithm that is a way for the agent to iterate with the environment to learn better policies. In the face of the dynamic nature of the network environment and node status, traditional mechanism-based methods lack corresponding robustness. The online learning strategy of reinforcement learning can perform adaptive optimization on dynamic conditions in real-time, so as to achieve better results.

In the reinforcement learning algorithm, an agent interacts with the environment, the agent explores the information in the environment, and uses this information to update the parameters inside the learning agent to make the agent perform better in this environment. DQN is a value-based method [17], and the object of the update iteration is a value function Q, which is used to evaluate the value of the action selected by the agent. A higher value for an action means a better effect achieved in the environment.

Here we use the DQN-based task offloading algorithm to solve the consumer device selection problem. The algorithm is deployed in the Consumer Decision module. First, the Consumer Decision module decides which consumer device the current transcoding task is to execute, obtains the state information S and action information A, and then uses the information to update the value function Q with the Temporal-Difference learning algorithm [22] and finally use the new value function to determine a better decision policy.

Specifically, the state is denoted as $S = \{s_1, s_2, ..., s_t, ...\} = \{r_{v,1}, r_{v,2}, ..., r_{v,t}, ...\}$, where $s_t = r_{v,t}$ represents the resource occupancy status of the resource pool of viewer v in the t-th time slot. The action is represented as $A = \{a_1, a_2, ..., a_t, ...\}$, in which a_t means the decision result at the t-th time slot. Obviously, a_t is a vector in one-hot form and $|a_t| = |C_v|$. And the element with a value of 1 is the selected consumer device.

The strategy function is denoted by π, and the strategy function decides the action selected by the agent at the next moment. $\pi(s_t)$ denotes the next action which means the action decision made by the current policy with the function π at state s at time t. And We denote the value function by $Q^{\pi}(s_t, a_t)$, which means under strategy π, the value of taking action a_t in state s_t at time t. Firstly, we use a greedy algorithm to design policy function π

$$\pi'(\mathrm{s}) = \arg\max_{\mathrm{a}} \mathrm{Q}^{\pi}(\mathrm{s}, \mathrm{a}) \tag{15}$$

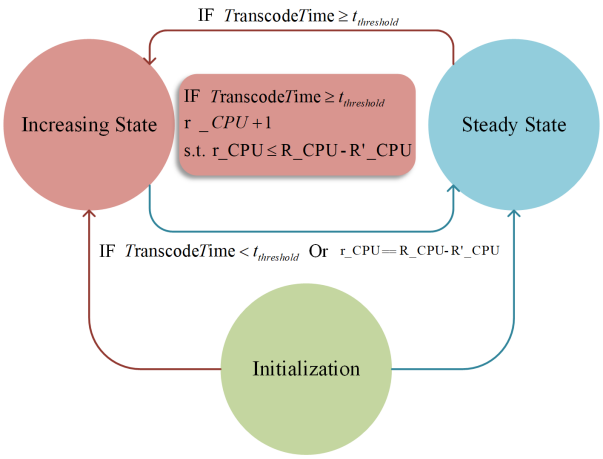

Fig. 2. Adaptive Computing Power Allocation Algorithm

which means we choose the action that yields the most value every time. And we use the Temporal-Difference learning algorithm to update the value function Q

$$\mathrm{Q}^{\pi}\left(\mathrm{s_t}, \mathrm{a_t}\right) = \mathrm{r_t} + \mathrm{Q}^{\pi}\left(\mathrm{s_{t+1}}, \pi\left(\mathrm{s_{t+1}}\right)\right) \tag{16}$$

Among them, reward r_t is represented by the optimization objective defined in the fourth section. Since we usually use the term maximizing the reward, the optimization objective is reversed here.

$$r_t = \alpha \cdot \mathcal{U}_{avg} - \beta \cdot ProcessTime_{avg} \tag{17}$$

In this way, under the iteration of the DQN algorithm, the selection of consumer devices will become more and more superior.

Additionally, we employ the target network technique. During training, the parameters in the Q network on the right side of equation (16) are fixed, because the Q network on the right is responsible for generating the target, so it is called the target network. Because the target network is fixed, the value of target $r_t + Q_{\pi}\left(s_{t+1}, \pi\left(s_{t+1}\right)\right)$ obtained now is also fixed. We only tune the parameters of the left Q-network and it becomes a regression problem. We want the output of the model to be as close to the target as possible, so as to minimize its mean square error. When implementing, we will update the Q network on the left multiple times, and then replace the target network with the updated Q network.

The pseudocode is introduced in Algorithm 1. This solves the computing task offloading problem in the Computing Power Distribution process.

2) Computing Allocation Algorithm

In this section, we will introduce the computing power distribution algorithm that we have designed. Our approach is based on an iterative adaptive optimization algorithm that takes into account various factors. The objective of this algorithm is to optimize the allocation of computing resources so that existing resources are used as effectively as possible. Specifically, the resource allocation scheme needs to satisfy two main criteria: it must minimize task execution time while also maximizing the utilization of computing resources.

The flow chart of the algorithm is shown in Fig. 2. We designed a mechanism-based iterative adaptive optimization algorithm. First, after the task offloading algorithm in the previous section decides which consumer the computing task will be executed in, the consumer decides how many computing resources to use to perform the transcoding task.

When the chunk $\mathcal{G}_{d,g}$ of the live stream with the video bit rate level d arrives at this consumer, the amount of computing data needed to calculate the transcoding task is $Data_{i,dd'}$. The consumer obtains this information, first performs the initialization process, and sets the computing resources used to 1, that is $r\_CPU_i = 1$. Then we calculate the processing time $ProcessTime$ that will be spent according to Equation (6)(7)(8). The concept of time-threshold is introduced here, and the time threshold is represented as $t_{threshold}$, which is usually set to the video length of the chunk. In actual live broadcasting, the video processing time is shorter than the length of the video itself, which is a prerequisite for the video to be played without lag. Of course, if the processing time of a small number of video chunks is longer than the length of the chunk itself, it does not mean that there must be a freeze, but it will definitely consume the length of the video buffer and adversely affect video playback. So we set $t_{threshold}$ here to be the length of the chunk. If at this time, $TranscodeTime >= t_{threshold}$, the processing state is transformed from the initial state to the Increasing State. If $TranscodeTime < t_{threshold}$, the processing state is transformed to Steady State which means keep the current allocation. Besides, in the Increasing State, the number of CPUs originally allocated is increased by 1, and then it is checked whether the TranscodeTime is greater than the threshold time. Continue this loop until the number of CPUs makes the processing time meet the threshold condition or the number of CPUs reaches the maximum limit for this device. And then the state is transformed to the Steady State.

VI. Routing Decision Phase

In this section we introduce the routing process. After the decision-making process of the computing power distribution process, the decision brain has obtained the target consumer device, and the anchor who pushes the streaming media content is also determined. Therefore, we need to make a decision to determine the streaming media transmission routing scheme between the two points. Since the increase in the number of routing hops will lead to an increase in transmission delay, we set the number of routing hops to 3 hops [5]. We design a hybrid Simulated Annealing Genetic Algorithm(SAGA) to perform the routing decision task. This algorithm is deployed in the Relay Routing module. This module will first obtain the bandwidth resource occupancy of each routing node from the Global Discovery module, and then make routing decisions based on the size of the transmitted video block.

A. Introduction to Genetic Algorithm

Genetic Algorithm is a randomized search method that draws on natural selection and natural genetic mechanisms in the biological world [23]. Since the algorithm uses random selection, it has no special requirements on the search space,

has the advantages of simple operation and fast convergence speed, and is suitable for dealing with complex and nonlinear problems that are difficult to solve by traditional search methods. The main feature of the genetic algorithm is the group search strategy and the information exchange between individuals in the group. It actually simulates the overall learning process of the group composed of individuals, and each individual corresponds to a solution to the research problem. The genetic algorithm starts from an initial group, through selection(making the outstanding individuals in the group have more opportunities to be passed on to the next generation), crossover(reflecting the information exchange between individuals in the group in nature), and mutation(in the group Genetic operations such as introducing new variants to ensure the diversity of information in the population) allow the population to evolve from generation to generation to better and better regions in the search space. Genetic algorithms include elements such as encoding, initial population generation, fitness assessment, selection, crossover, and mutation.

B. Introduction to Simulated Annealing Algorithm

The simulated annealing algorithm is an effective approximation algorithm for solving large-scale combinatorial optimization problems, especially NP complete combinatorial optimization [24]. The physical image and statistical properties of the solid annealing process are the physical background of the simulated annealing algorithm. The Metrolpis acceptance criterion makes the algorithm jump out of the "trap" of local optimum, and the reasonable selection of the cooling schedule is the premise of the algorithm application. Solid annealing is a thermodynamic process in which a solid is first heated to melt, and then slowly cooled to solidify into a regular crystal. From the point of view of statistical physics, as the temperature decreases, the energy of matter will gradually approach a lower state, and finally reach a certain balance. The solid temperature parameter T, the state transition process is repeated, and the acceptance probability $P(x)$ of the new state obeys the Gibbs distribution:

$$P(x) = \frac{1}{z}\exp\left(-\frac{E(x)}{T}\right) \tag{18}$$

In the formula (18), z is the probability regularization coefficient, and $E(x)$ is the energy of state x. It can be seen from the above formula that as the temperature parameter decreases, the acceptance probability also decreases, that is, the possibility of increasing the energy function also gradually decreases, and finally the system will converge to a state with the minimum energy. Simulating such a solid annealing process applies it to function optimization.

C. The Process of Routing Decision

Genetic Simulated Annealing Algorithm is an optimization algorithm that combines a genetic algorithm and a simulated annealing algorithm. The local search ability of the genetic algorithm is poor, but the ability to grasp the overall search process is strong; while the simulated annealing method has a strong local search ability, and can prevent the search process from falling into the local optimal solution, but the simulated annealing algorithm has no effect on the entire search process. Not much is known about the state of the search space. It is not convenient to make the search process enter the most promising search area so that the operation efficiency of the simulated annealing algorithm is not high. However, if the genetic algorithm and simulated annealing algorithm are combined to learn from each other, it is possible to develop a new global search algorithm with excellent performance. This is the basic idea of the genetic simulated annealing algorithm. We use the genetic simulated annealing algorithm to make routing decisions, the specific process is shown in Fig. 3.

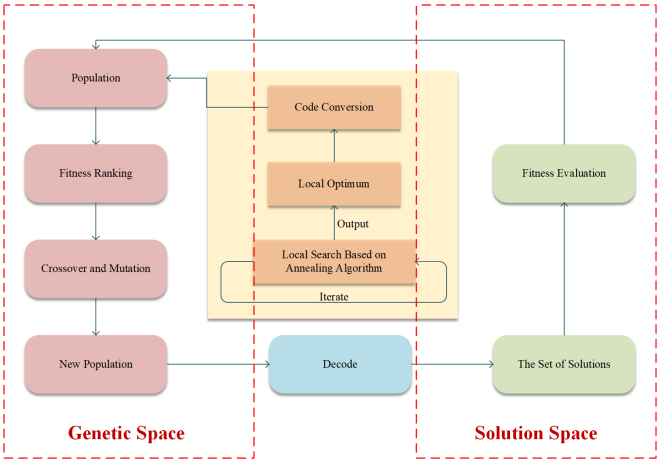

Fig. 3. Hybrid Genetic Simulated Annealing Algorithm Process

1) Population Modeling

For the encoding method of the genetic algorithm, we propose a two-tuple encoding method in combination with the scene characteristics of routing selection. Take producer P_b and consumer C_v as an example, the set of $\mathcal{O} = \{1, 2, ..., o, ...\}$ is the relay routing node. We extract two nodes from the relay routing set to form a two-tuple, for example, $[\mathcal{O}_i, \mathcal{O}_j]$, which is an individual in a genetic algorithm. A collection of randomly selected individuals is called a population, in which we generate N individuals. Thus the population is represented as $\mathcal{S} = \{s_1, s_2, ..., s_k, ..., s_N\}$.

2) Fitness Function Design

We design a fitness function that can reflect the routing delay. First, the bandwidth between producer P_b and relay routing device $\mathcal{O}_i$ is maintained in the relationship matrix of P_b and $\mathcal{O}$. Similarly, the relationship matrix between $\mathcal{O}$ and $\mathcal{O}$ and the relationship matrix between $\mathcal{O}$ and C_v both maintain the bandwidth status between nodes. We design the fitness function based on the equation (13).

$$F([\mathcal{O}_i, \mathcal{O}_j]) = -\frac{Data_d}{B_{P_b,\mathcal{O}_i}} - \frac{Data_d}{B_{\mathcal{O}_i,\mathcal{O}_j}} - \frac{Data_d}{B_{\mathcal{O}_i,C_v}} \tag{19}$$

We evaluate individuals in the population and sort them according to this fitness function. According to the results of the fitness ranking, we use the strategy of combining the best individual retention and roulette selection for individual selection. Firstly, the individual whose fitness ranks first has the best performance in this generation, and it is directly copied to the next generation. The other N-1 individuals of

Algorithm 2: Simulated Annealing Algorithm

```
// Initialization
Initialize solution S = S[0], iteration step k = N
// Iteration Begin
while k >= 1 do
    1) Generate S' = S[k].
    2) Calculate ΔF = F(S') − F(S).
    3) Prob = min(1, e^{−ΔF/k})
    4) if Prob > random(0, 1) then
         S = S', F(S) = F(S')
       end
    5) k = k − 1
end
```

the next generation are generated by the roulette selection method according to the fitness of the previous generation population. Specifically, calculate the proportion of the fitness of the previous generation of individuals to the total fitness as equation (20).

$$Ratio_k = \frac{F_k}{\sum_{k=1}^{N} F_k} \tag{20}$$

This ratio is the probability of the individual surviving, which means the probability of being saved for the next generation. This selection method can ensure that the optimal individual can be preserved for the next generation, and at the same time, the individual with high fitness has a greater probability of surviving to the next generation.

3) Crossover and Mutation

Firstly, the crossover operation is introduced. For the new population generated by the selection operation in the previous step, in addition to the optimal individual ranked first, the other N-1 individuals are paired and cross-recombined with the probability of P_c. This probability P_c is the set hyperparameter. Since the individual genes in this scenario are in the form of binary groups $[\mathcal{O}_i, \mathcal{O}_j]$, the gene crossover operation between individuals is easy to implement. Next, introduce the mutation operation. I still retain the individual with the best fitness, and then generate gene mutations for the other N-1 individuals with the probability P_m. Specifically, a gene is randomly selected in the gene bank to replace one of the binary groups $[\mathcal{O}_i, \mathcal{O}_j]$ in the individual. It is worth noting that during the operation of crossover and mutation, individuals that cannot exist will be actively eliminated, such as two individuals with the same gene. Through the cross-mutation operation, while retaining the best performing individuals, it promotes the generation of species diversity.

4) Simulated Annealing Algorithm

This section introduces the simulated annealing algorithm, and its algorithm flow is shown in Algorithm 2. First, we obtain the individual with the highest fitness in the population, that is the individual $\mathcal{S}[0]$ that ranks first in the set $\mathcal{S}$. Then we iterate over the individuals in N-1 individuals, and use $S[k]$ to represent the k-th individual in the population. Calculate the fitness $F(S[k])$ of this individual and compare it with $F(S[0])$.

$$\Delta F = F(\mathcal{S}[k]) - F(\mathcal{S}[0]) \tag{21}$$

Then we judge that if $\Delta F \leq 0$, it means that the fitness of the k-th individual is better than the initial individual, so this k-th individual is replaced by the highest-ranked individual in the population set. If it is judged that $\Delta F > 0$, which means that the k-th individual is less adaptable, but in order to retain the exploration ability, a certain probability is still used to replace the k-th individual with the optimal solution. This probability is expressed as $e^{\frac{\Delta F}{k}}$. In summary, the replacement probability can be expressed as

$$Prob = \begin{cases} 1 & \Delta F \leq 0 \\ e^{\frac{\Delta F}{k}} & \Delta F > 0 \end{cases} \tag{22}$$

As shown in Algorithm 2, the value of k decreases from N to 1, which means that the replacement probability decreases gradually when $\Delta F > 0$. Then iterate until exiting the loop.

VII. Performance Evaluation

In this section, we evaluate the performance of the proposed task offloading algorithm and routing algorithm under the computing power network-based live streaming transmission architecture. The experimental procedure is as follows: First initialize the network device status in the scene, including the definition of the video block requested by a user at the current moment, the computing resource occupancy of 10 consumer devices, and the bandwidth resource occupancy of a limited number of relay devices. Secondly, according to the task offloading algorithm proposed in Section V, the consumer device that performs the transcoding task is decided and its computing resources are allocated. Finally, on the basis of the above, a routing algorithm is proposed in Section VI to decide the relay path. Next in this section, we introduce the experimental environment of the simulation setup, and then present the experimental results and analyze them.

A. Simulation Environment

We use a PC with an Intel Core i5-8250 CPU 1.8 GHz and 20GB of RAM for numerical simulations. The simulated data is based on watching the live broadcast on the PC in the laboratory environment, and the bandwidth resource design in the simulation is based on 3GPP standardization. We use several most popular resolutions {2k, 1080p, 720p, 360p} as target resolutions. The corresponding bit rates and required computing resources are based on [25]. This is based on the fact that RTMP protocol is generally used for live streaming nowadays. For the convenience of experimental testing, we set a single chunk of live video streaming to $0.5s$. We consider a scene focusing on a region, as shown in Fig. 1.

Edge consumer settings: For a certain user, we set up 10 edge devices (i.e. consumers). As described in the computing resources in the second subsection of Section III, edge devices include CPU, memory RAM, and hardware disk status. In the simulation environment, for example, we simulate the parameters of the common intel Core i5 8300h processor on the market, the number of CPU cores is 4, that is, $N_{core} = 4$, and the number of threads is 8. The basic frequency of the processor is 2.30GHz, that is, $Fre_{core} = 2.30GHz$. The emulation setting memory is 16G. For the convenience of

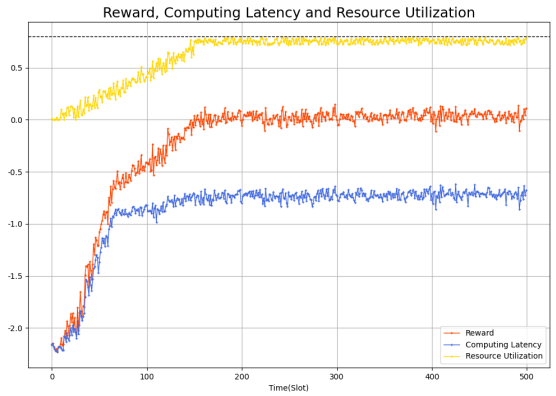

Fig. 4. The trend of training process

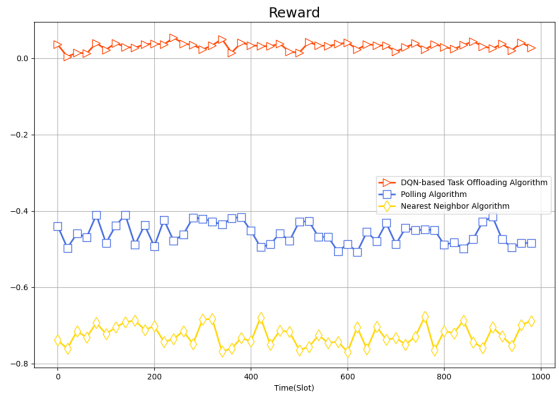

Fig. 5. Average Reward

experimental design, regardless of the capacity of the hard disk, here we reasonably assume that the hard disk space of the device will not be full during the live broadcast. In fact, the bottleneck of the usual live broadcast user experience will appear in the processing speed of the CPU in the edge server and the reading speed of the memory from the hard disk, but not in the capacity of the hard disk. The resource occupancy of edge devices is related to user requests. The frequency of available computing resources obeys a normal distribution at $\lambda = 2GHz$, and the processing frequency range is $[0, 4GHz]$. The number of CPU cores is chosen randomly on the set of [2,4,8,16]. It is worth noting that the upper limit of computing resource utilization mentioned in Section III is set to $u = 0.8$, and if the decision exceeds this upper limit, a huge penalty will be obtained.

Router settings: As described in Section III, the routers of the relay network have the attributes of bandwidth occupation and remaining available bandwidth. The available bandwidth resources obey the normal distribution with $\lambda = 5MHz$, and the bandwidth range is $[0, 10MHz]$ [26]. Considering the number of routers, we set up 9-81 relay nodes to try to test the influence of the number of routers on the routing decision algorithm.

Time setting: Each time slot is set to 0.5s, so one video chunk will be processed in one-time slot. The process of a chunk being processed is a step in reinforcement learning training.

B. Performance and Result Analysis

This section introduces the algorithm performance and result analysis. It is divided into two parts. First, we introduce the decision-making process of task offloading, and then introduce the routing selection process.

1) Decision-Making Process of Task Offloading

Training analysis: The reward is a linear combination of the Calculation Delay in the consumer and the Resource Utilization rate of the consumer, and the specific calculation is shown in equation (11). The basic trend is that the smaller the calculation delay and the greater the resource utilization cause the greater the total reward. The specific DQN-based Task Offloading Algorithm training performance is shown in Fig. 4. Computing delay calculation is represented by equation (10), and computing resource utilization is represented by equation (5). The DQN-based Task Offloading Algorithm is trained for 500 time slots. During training, the absolute value of the calculation delay in the consumer device gradually decreases with each iteration of the training step, and eventually converges at around -0.75. In contrast, the computing resource utilization starts low at the beginning of the task, but increases steadily with each iteration until it finally converges to a level close to 0.8. It's worth noting that we have set an upper limit of 0.8 for the utilization of computing resources, in order to ensure that the device's computing power can handle sudden surges in computing tasks. As a result, the reward during training also increases with the number of slots, until it eventually converges to around 0. During the training process, the reward may fluctuate, but this indicates that the DQN-based Task Offloading Algorithm is gradually learning from its environment through exploration, and ultimately selects actions that yield excellent returns.

Computing task processing delay: Fig. 6 describes the training trend of the specific two sub-elements in the computing task processing delay. Calculation latency is the processing time c_t of computing tasks in the CPU and the replacement time s_t for reading data from the hard disk when the memory space is insufficient. The latency of these two dimensions is also the average latency of the overall environment. The shaded part in Fig. 6 is the actual value interval of multiple pieces of training, and the solid line is the average value of multiple pieces of training. It can be observed that the overall trend of multiple pieces of training is consistent, and the CPU processing time and Storage Replacement Time both decrease as the time slot (number of training iterations) increases. Among them, the CPU processing time finally converges to about 0.75s, and the Storage Replacement Time can converge to about 0. It shows that with exploration and learning, the Consumer Decision module can determine a better consumer server through the DQN-based Task Offloading Algorithm. This better consumer server is most suitable for processing the current transcoding task. Specifically, the currently schedulable

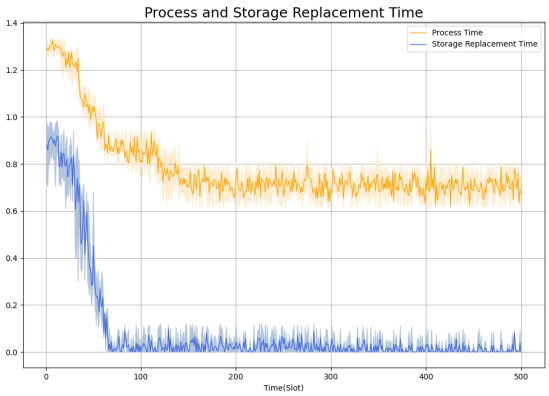

Fig. 6. The trend of Process Time and Replacement Time

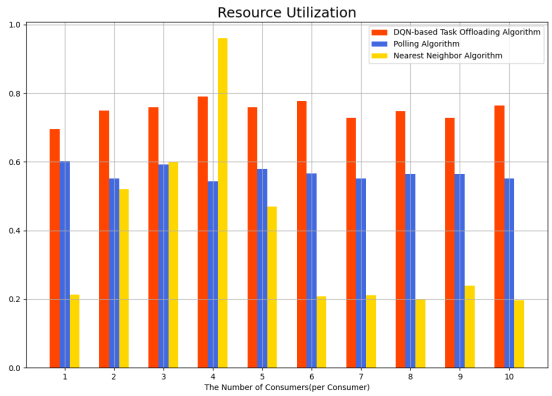

Fig. 7. The ratio of Resource Utilization

CPU resources on the Consumer device are most suitable for processing this task, which can minimize the average computing delay in the overall environment. At the same time, the convergence of the Storage Replacement Time to 0 also means that the currently selected Consumer device can rely on the memory space to complete the storage when processing the transcoding task. The data to be processed does not need to be stored on the hard disk without generating Storage Replacement Time. The changing trend of the overall Computing Latency in Fig. 4 can be combined with the processing delays of these two dimensions in one-to-one correspondence.

Comparative algorithm performance analysis: Fig. 5 depicts the performance comparison between two mainstream task offloading schemes and our proposed DQN-based Task Offloading Algorithm. To analyze the reward level more clearly, the average reward of 20 steps is depicted in this figure. The blue line represents the polling algorithm [27], which is a commonly used load-balancing strategy in task offloading scenarios. Specifically, the polling algorithm sequentially distributes computing tasks to consumer devices in the set of devices that can participate in providing services. The yellow line in Fig. (5) represents the reward performance of the Nearest Neighbor Algorithm. This algorithm is also a commonly used decision-making scheme in edge computing task offloading. In detail, the Nearest Neighbor Algorithm assigns computing tasks to the edge device closest to the user. In practice, the consumer device will be prioritized according to the geographical distance, and the device with the highest priority will provide computing services to the user first. When the amount of calculated data exceeds the load capacity of the device, the algorithm selects the second-priority device to perform the calculation task and then iterates with the same logic. Fig. 5 compares the performance of the three algorithms. The overall reward of the DQN-based Task Offloading Algorithm proposed by us is higher than the other two algorithms, and the volatility is lower than the other two algorithms. The reward level of the Polling Algorithm is higher than that of the Nearest Neighbor Algorithm, but they are all below the 0 levels. The DQN-based Task Offloading Algorithm proposed by us has been explored and learned in the early stage, and the final performance is more suitable for the dynamic live network conditions, and better consumer devices can be selected.

Fig. 7 records the average computing resource utilization of 10 consumers in a certain video stream. The red bar graph is the performance of the DQN-based Task Offloading Algorithm algorithm proposed by us. The computing resource utilization rate of each consumer will be higher than that of the other two algorithms. Among them, the utilization rate of the Nearest Neighbor Algorithm on the No. 4 consumer is particularly high, almost close to 1, which means that the No. 4 consumer device is almost overloaded beyond this time. Obviously, edge device No. 4 is the consumer closest to the user at this time, so the Nearest Neighbor Algorithm prioritizes almost all computing tasks of this flow to device No. 4. Secondly, the utilization rate of equipment No. 2, 3, and 5 is relatively high. However, the utilization rate of the rest of the consumer devices is very low, which leads to a low overall average computing resource utilization rate, and the computing pressure of some devices is very high. The overall distribution of computing resource utilization using the Polling Algorithm is uniform, but the level is significantly lower than our proposed scheduling algorithm. Mechanically cycling through all devices can balance the load status of the devices well, but because the actual dynamic live broadcast situation is not considered, some consumer devices with remaining processing capacity are not selected, resulting in longer overall processing time and certain waste of computing resources. In general, the DQN-based Task Offloading Algorithm proposed by us is both robust and efficient in computing resource utilization and is superior to the other two algorithms.

2) Routing Selection Process

Iteration process: SAGA algorithm based on generation iterative optimization. Fig. 8 describes the fitness change trend of SAGA, traditional Genetic Algorithm, and Simulated Annealing Algorithm. The fitness calculation, such as Equation (19), is a direct description of the transmission delay in the relay network. The horizontal axis of Fig. 8 is the number of iterations, and the vertical axis is the fitness, which also represents the opposite number of transmission delays. The

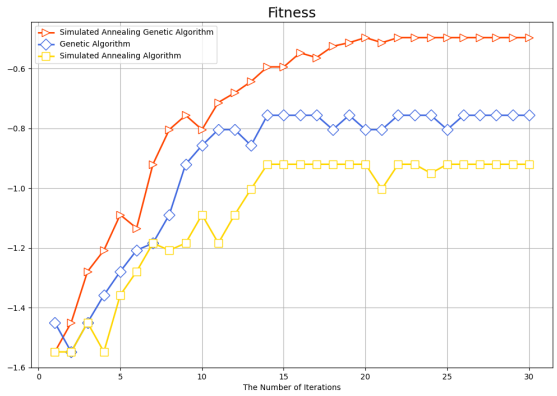

Fig. 8. The trend of Fitness

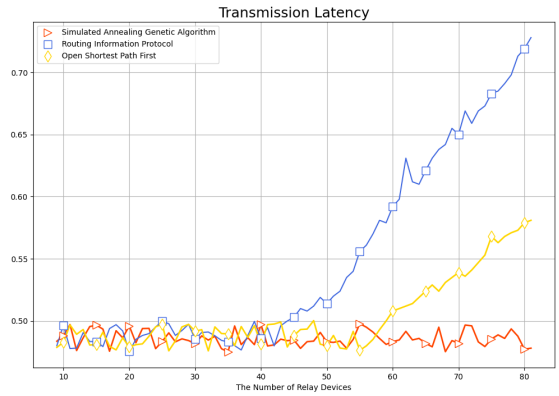

Fig. 9. The trend of Transmission Latency

fitness of the optimal individuals determined by the three algorithms will increase as the number of iterations increases, and will eventually converge to the optimal solution that the algorithm can find. Obviously, under the same number of iterations, SAGA can find individuals with better fitness than GA and SAA. SAGA combines the advantages of GA and SAA. GA benefits from the search of the entire population, and the general trend of combinatorial optimization is more inclined toward individuals with greater fitness. In the early stage, it is easy to spread individual good individuals to the entire population, but in the later stage of GA iteration, the fitness of each individual tends to be consistent, which makes iterative optimization of the population stagnant. On the contrary, SAA has a strong local search ability, but it is easy to fall into the dilemma of local optimum and cannot produce individuals with higher fitness. In summary, SAGA has a stronger ability to generate individuals with higher fitness and can find routing paths with lower transmission delays more effectively.

Comparative algorithm performance analysis: Fig. 9 shows the performance comparison between the SAGA algorithm and the traditional routing decision algorithm Routing Information Protocol (RIP) [28] and Open Shortest Path First (OSPF) [29]. The results of the curves in the figure are the optimal results of the algorithm iteration. The horizontal axis is the number of router devices in the current scene, and the vertical axis is the transmission delay of the relay network. The basic idea of the RIP algorithm is greedy, and each hop selects the link with the optimal bandwidth resource. OSPF maintains the routing table in the relay device and automatically calculates the update of the topology state of the route with a small weight. As shown in Fig. 9, when the number of optional relay routers in the scenario is relatively small, such as 9-40, all three algorithms can determine the optimal routing selection, and the overall performance is almost the same. This is because the overall environment is simple and the algorithm is easy to calculate the optimal solution. However, as the number of optional relay devices increases, the routing environment becomes complex, and both RIP and OSPF are gradually unable to calculate the optimal routing path solution. At least in the case of 81 routing devices, our proposed SAGA algorithm can still obtain the optimal solution through iteration. It is worth admitting that SAGA is a centralized calculation, so the calculation speed is faster, but iteration is required, and the cost will be relatively higher. On the other hand, OSPF decentralized computing has slow computing speed and inferior performance to SAGA, but the foreseeable cost is lower.

VIII. Conclusion

This paper introduces a novel architecture for live stream transmission based on a computing power network to address the challenge of real-time scheduling in live broadcast scenarios. We approach content delivery of live media streaming by breaking it down into two subproblems: transcoding and compiling, and routing path selection in relay networks. To enhance the viewing experience of audiences, we propose two optimization algorithms to solve these problems. Firstly, we suggest a DQN-based task offloading algorithm that considers the computing power of edge devices to optimize computing delay and computing resource utilization. Secondly, we design a simulated annealing genetic algorithm that takes into account node and link bandwidth resources to determine the routing path. The proposed algorithm is rigorously tested and validated through simulation experiments, and compared with a benchmark algorithm. The results demonstrate that our solution outperforms current common solutions in terms of computing processing delay, computing resource utilization, and transmission delay. Our work in this paper lacks validation of algorithm deployment loads, which should be considered additionally in future work.

Acknowledgment

This work is supported by the Postdoctoral Science Foundation of China under grant No. 2022M720518 and by the National Natural Science Foundation of China (NSFC) under grant No. 62225105 and 62001057

REFERENCES

[1] Cisco, "Cisco Annual Internet Report (2018–2023) White Paper," March 2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[2] M. Dai, Z. Su, R. Li and S. Yu, "A Software-Defined-Networking-Enabled Approach for Edge-Cloud Computing in the Internet of Things," in IEEE Network, vol. 35, no. 5, pp. 66-73, September/October 2021.

[3] X. Wang, Z. Ning, L. Guo, S. Guo, X. Gao and G. Wang, "Online Learning for Distributed Computation Offloading in Wireless Powered Mobile Edge Computing Networks," in IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 8, pp. 1841-1855, 1 Aug. 2022.

[4] N. Hu, Z. Tian, X. Du, N. Guizani and Z. Zhu, "Deep-Green: A Dispersed Energy-Efficiency Computing Paradigm for Green Industrial IoT," in IEEE Transactions on Green Communications and Networking, vol. 5, no. 2, pp. 750-764, June 2021.

[5] J. Li, Z. Li, R. Lu, K. Xiao, S. Li, J. Chen, J. Yang, C. Zong, A. Chen, Q. Wu, C. Sun, G. Tyson, and H. Liu, "LiveNet: A Low-Latency Video Transport Network for Large-Scale Live Streaming," in Proc. ACM SIGCOMM, New York, NY, USA, Aug. 2022, pp. 812-825.

[6] E. Kwong, K. Pan, C. Lafata, and R. Puri, "Ingesting Live Video Streams at Global Scale," Twitch, Apr. 26, 2022. [Online]. Available: https://blog.twitch.tv/zh-tw/2022/04/26/ingesting-live-video-streams-at-global-scale/.

[7] Q. Jia, R. Ding, H. Liu, C. Zhang, and R. Xie, "Survey on research progress for compute first networking," Chinese Journal of Network and Information Security, vol. 7, no. 3, pp. 1-12, Mar. 2021.

[8] M. Król, S. Mastorakis, D. Oran and D. Kutscher, "Compute First Networking: Distributed Computing meets ICN," 2019 6th ACM Conference on Information-Centric Networking (ICN), Macao, SAR, China, 2019, pp. 67-77.

[9] P. Rahimzadeh et al., "SPARCLE: Stream Processing Applications over Dispersed Computing Networks," 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), 2020, pp. 1067-1078.

[10] Q. Chen, F. R. Yu, T. Huang, R. Xie, J. Liu and Y. Liu, "Joint Resource Allocation for Software-Defined Networking, Caching, and Computing," in IEEE/ACM Transactions on Networking, vol. 26, no. 1, pp. 274-287, Feb. 2018.

[11] X. Xu, J. Liu and X. Tao, "Mobile Edge Computing Enhanced Adaptive Bitrate Video Delivery With Joint Cache and Radio Resource Allocation," in IEEE Access, vol. 5, pp. 16406-16415, 2017.

[12] C. Liang and S. Hu, "Dynamic video streaming in caching-enabled wireless mobile networks," arXiv preprint arXiv:1706.09536, 2017.

[13] H. A. Pedersen and S. Dey, "Enhancing Mobile Video Capacity and Quality Using Rate Adaptation, RAN Caching and Processing," in IEEE/ACM Transactions on Networking, vol. 24, no. 2, pp. 996-1010, April 2016.

[14] Z. Wang, L. Sun, C. Wu, W. Zhu, Q. Zhuang and S. Yang, "A Joint Online Transcoding and Delivery Approach for Dynamic Adaptive Streaming," in IEEE Transactions on Multimedia, vol. 17, no. 6, pp. 867-879, June 2015.

[15] Y. Zheng, D. Wu, Y. Ke, C. Yang, M. Chen and G. Zhang, "Online Cloud Transcoding and Distribution for Crowdsourced Live Game Video Streaming," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 27, no. 8, pp. 1777-1789, Aug. 2017.

[16] T. X. Tran, A. Hajisami, P. Pandey and D. Pompili, "Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges," in IEEE Communications Magazine, vol. 55, no. 4, pp. 54-61, April 2017.

[17] T. Zahavy, N. Ben-Zrihem, and S. Mannor, "Graying the black box: Understanding DQNs," in Proceedings of The 33rd International Conference on Machine Learning, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48, New York, NY, USA, PMLR, 2016, pp. 1899-1908.

[18] G. Zhang, B. Tang, J. Sun and J. Li, "Ant colony routing strategy based on distribution uniformity degree for content-centric network," Journal on Communications, vol. 36, no. 6, pp. 2015126-1-2015126-12, Jun. 2015.

[19] F. Muchtar, A. A. Hanan, H. Suhaidi, K. A. Tajudin, and Z. K. Zuhairi. "Energy conservation of content routing through wireless broadcast control in NDN based MANET: A review," Journal of Network and Computer Applications, vol. 131, pp. 109-132, 2019, ISSN 1084-8045.

[20] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and M. Stiemerling, "Real-Time Streaming Protocol Version 2.0," in IEEE, RFC 7826, DOI 10.17487/RFC7826, Dec. 2016, pp. 1-102.

[21] K. Spiteri, R. Urgaonkar and R. K. Sitaraman, "BOLA: Near-Optimal Bitrate Adaptation for Online Videos," in IEEE/ACM Transactions on Networking, vol. 28, no. 4, pp. 1698-1711, Aug. 2020.

[22] A. Tamar, D. Di Castro, and S. Mannor, "Temporal Difference Methods for the Variance of the Reward To Go," in Proceedings of the 30th International Conference on Machine Learning, Sanjoy Dasgupta and David McAllester, Eds., vol. 28, no. 3, Atlanta, Georgia, USA, 17-19 Jun. 2013, pp. 495-503.

[23] J. H. Holland, "Genetic Algorithms and the Optimal Allocation of Trials," SIAM Journal on Computing, vol. 2, no. 2, pp. 88-105, 1973.

[24] R. Benedetti, M. M. Dickson, G. Espa, F. Pantalone and F. Piersimoni, "A simulated annealing-based algorithm for selecting balanced samples," in Computational Statistics, vol. 37, no. 1, pp. 491-505, Mar. 2022.

[25] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon, "Transcoding Live Adaptive Video Streams at a Massive Scale in the Cloud," in Proceedings of the 6th ACM Multimedia Systems Conference, MMSys '15, Portland, Oregon, 2015, pp. 49-60.

[26] Z. Zhang, R. Wang, F. R. Yu, F. Fu and Q. Yan, "QoS Aware Transcoding for Live Streaming in Edge-Clouds Aided HetNets: An Enhanced Actor-Critic Approach," in IEEE Transactions on Vehicular Technology, vol. 68, no. 11, pp. 11295-11308, Nov. 2019, doi: 10.1109/TVT.2019.2942629.

[27] V. M. Vishnevsky, O. V. Semenova, and D. T. Bui, "Investigation of the Stochastic Polling System and Its Applications to Broadband Wireless Networks," Automation and Remote Control, vol. 82, no. 9, pp. 1607-1613, Sep. 2021.

[28] G. Malkin, "RIP Version 2," STD 56, RFC 2453, DOI: 10.17487/RFC2453, Nov. 1998.

[29] J. Moy, "OSPF Version 2," in STD 54, RFC 2328, Apr. 1998.