Title

Proffler: Toward Collaborative and Scalable Edge-Assisted Crowdsourced Livecast

Journal

IEEE Internet of Things Journal, 11(2)

ISSN

2372-2541

Authors

Zhang, Wenyi

Xu, Zihan

Wang, Fangxin

et al.

Publication Date

2024-01-15

DOI

10.1109/jiot.2023.3297843

Peer reviewed

Proffler: Toward Collaborative and Scalable Edge-Assisted Crowdsourced Livecast

Wenyi Zhang, Zihan Xu, Fangxin Wang, *Member, IEEE*, and Jiangchuan Liu, *Fellow, IEEE*

***Abstract*—In recent years, crowdsourced livecast has seen remarkable progress due to the interactivity and real-time nature, playing an essential role in multimedia applications in the post-epidemic era. Given the delay sensitivity, large viewing volumes, and heterogeneous viewing patterns, the traditional video streaming methods fail to provide the optimized Quality of Experience (QoE) for viewers using the minimum system cost over an edge-assisted service architecture. The emerging technology of mobile edge computing (MEC) offers a new perspective of reducing user latency and enhancing the quality of dispatched videos in a promising way. In this article, we propose `Proffler`, an integrated framework that addresses this problem through effective stream caching at the network edge server. We first examine the underlying correlations in viewing patterns across different regions and propose a novel transformer-based algorithm, Chili-TF, that achieves accurate viewer request prediction, even for regions with insufficient data. We then design a scalable algorithm, U2VR, that achieves near-optimal video stream allocation as well as viewer scheduling. Extensive real-data-driven experiments further confirm that `Proffler` can achieve improvements of 20%–55% in average QoE compared to state-of-the-art solutions.**

***Index Terms*—Request scheduling, video stream allocation, viewer request prediction.**

Manuscript received 25 November 2022; revised 8 June 2023; accepted 16 July 2023. Date of publication 24 October 2023; date of current version 8 January 2024. This work was supported in part by the Basic Research Projects of Hetao Shenzhen-HK S&T Cooperation Zone under Grant HZQB-KCZYZ-2021067; in part by NSFC under Grant 62293482 and Grant 62102342; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515012668; in part by the Shenzhen Science and Technology Program under Grant RCBS20221008093120047; in part by the National Key Research and Development Program of China under Grant 2018YFB1800800; in part by the Shenzhen Outstanding Talents Training Fund under Grant 202002; in part by the Guangdong Research Projects under Grant 2017ZT07X152 and Grant 2019CX01X104; in part by the Guangdong Provincial Key Laboratory of Future Networks of Intelligence under Grant 2022B1212010001; in part by the Shenzhen Key Laboratory of Big Data and Artificial Intelligence under Grant ZDSYS201707251409055; and in part by the Key Area Research and Development Program of Guangdong Province under Grant 2018B030338001. The work of Jiangchuan Liu was supported in part by CFI JELF/BCKDF Grant. *(Wenyi Zhang and Zihan Xu contributed equally to this work.) (Corresponding author: Fangxin Wang.)*

Wenyi Zhang and Zihan Xu were with the Future Network of Intelligence Institute and the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China (e-mail: wenyizhang@link.cuhk.edu.cn; zihanxu@link.cuhk.edu.cn).

Fangxin Wang is with the School of Science and Engineering and the Future Network of Intelligence Institute, The Chinese University of Hong Kong, Shenzhen 518172, China (e-mail: wangfangxin@cuhk.edu.cn).

Jiangchuan Liu is with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A1S6, Canada (e-mail: jcliu@sfu.ca).

Digital Object Identifier 10.1109/JIOT.2023.3297843

I. Introduction

CROWDSOURCED livecast, which involves real-time video streaming and content viewing from geo-distributed Internet users, gets benefits from the development of broadband mobile access [1] and smart mobile devices [2]. It has seen remarkable progress in recent years as it enables everyone to share customized videos lively and interactively. Such a wide variety of stream content attracts millions of viewers every day and brings billions of revenues for service providers [3]. The pandemic of Covid-19 further brings new opportunities for livecast, e.g., more companies choose to leverage livecast for product demonstration and sale promotion, and online classroom has become quite popular in more and more countries and regions. In the post-epidemic era and the foreseeable future, livecast will continue to play an essential role on the Internet and cultivate the rapid development of emerging new multimedia applications, such as virtual reality (VR) [4], Metaverse [5], etc.

Different from TV-based broadcast, the crowdsourced livecast is characterized by the sensitive streaming delay requirement, numerous geo-distributed viewers, heterogeneous viewer watching patterns, and various stream contents, rendering the conventional cloud-based streaming service ineffective to accommodate viewers' Quality of Experiences (QoEs). The readily deployed 5G and mobile edge computing (MEC) offers a new alternative, where computation-intensive stream transcoding and delivery for heterogeneous viewers can be offloaded to the distributed edge servers to provide services with shorter streaming delay and sufficient bandwidth [6], [7], [8]. Such architecture, however, faces two significant challenges. The first one lies in the highly dynamic and uncertain viewer requests in different regions, so it is challenging to accurately forecast the future viewing patterns of the geo-distributed viewers and make corresponding service allocations in advance. The second one comes from the mismatch between the distributed yet limited edge resources and the heterogeneous viewer requests, making it highly challenging to achieve optimal resource allocation and viewer scheduling in a scalable way toward high QoE and low system overhead.

Existing works are incapable of solving these problems concurrently. For the viewing pattern prediction, Li et al. [9] mainly focused on predicting the popularity of different video contents, ignoring the fast-changing popularity patterns and thus being unable to satisfy the highly dynamic live requests. Some other works [10], [11], [12] mainly focused on exploring the potential of the self-attention mechanism for time series

without a complete solution to solve this problem. For the edge resource allocation, Wang et al. [13] proposed an end-to-end learning-based solution for user scheduling, yet such a solution is generally a centralized algorithm and is insufficient to deal with the scalable condition in practical applications. Many other works [14], [15], [16] omit the heterogeneity of the edges and the users, which are not suitable for practical live-streaming scenarios. Thus, how to design an integrated framework to solve the problems remains challenging in crowdsourced livecast.

In this article, we propose `Proffler`,[1] an integrated framework that achieves collaborative viewing request prediction across different regions and scalable video stream allocation as well as viewer scheduling in edge-assisted crowdsourced livecast. `Proffler` consists of two major components, i.e., the request prediction module and the service scheduling module. The first module identifies the underlying common viewing patterns across different regions and extracts the similarity. We propose a novel transformer-based algorithm, Chili-TF, with local and regional similarity features that achieves accurate viewer request prediction, even for regions with insufficient data. In the second module, we design an optimization algorithm, i.e., U2VR, to solve this scheduling problem. U2VR first allocates video streams in a geographical popularity-based manner, and then it utilizes a utility-based heuristic to schedule users' requests. We have conducted extensive real-trace-driven experiments. The results demonstrate that our prediction algorithm outperforms the commonly used long short-term memory (LSTM) and state-of-the-art Transformer algorithm, achieving 0.03043 MSE on average. Besides, with the prediction results, our service scheduling algorithm is able to attain near-optimal stream allocation and user redirection, with up to 55.7% improvement.

II. Related Work

A. Time Series Prediction

Time series prediction is a research field in which data regularity is sorted based on historical data, and future nonoccurrence time is made. There are many existing methods to capture the periodicity of sequential time sequence through neural network-related algorithms [17], [18], [19], [20]. Recently, Recurrent Neural Network has been a popular research area in time series forecasting. LSTM [21] is proposed to solve the problem of gradient disappearance and gradient explosion in long sequence training. Sagheer and Kotb [22] do the prediction of petroleum production using deep LSTM recurrent networks and achieve good performance. Li et al. [23] designed an evolutionary attention-based LSTM for time series prediction. Karevan and Suykens [24] investigated two weighting schemes based on the cosine similarity between the training samples and the test point.

Transformer is another popular model to solve the sequence-to-sequence problem [12]. It is designed for natural language processing (NLP) task, which has some structural limitations for the long-time series prediction task. Recently, a few studies have tried to exploit the potential of Transformer for time series forecasting [11], [25]. However, no study has introduced Transformer into the live video requests forecast yet. The user's request is multidimensional related to geographical location, holidays, and time points, with rapid real-time change and an extensive variation range. This poses a significant challenge to the Transformer model application.

[1]"Proffler" is the combination of "Prophet" and "Niffler." Prophet can predict the future, representing our prediction model. Niffler is a greedy creature in the myth, representing our utility-based scheduling algorithm.

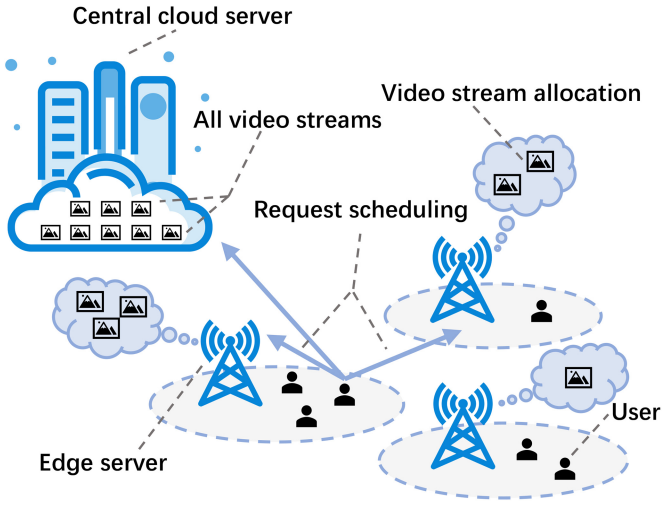

Fig. 1. Edge network system model.

B. Service Placement

Shareable resources between edge servers enable users to have access to different edges, opening up the problem of service placement. Researchers have proposed various algorithms for jointly placing service on edge servers and assigning users to edge servers with different objectives (e.g., minimizing delay [26], cost [27], or maximizing the number of users assigned to edges [28]). For example, an approximation algorithm with a greedy heuristic is used to guarantee the lower bound of the cost [14]. Yu et al. [16] proposed a decentralized algorithm to transform the placement problem into a matching problem between edge servers and users. Another way is to use the reinforcement learning-based method to solve the problem. According to Talpur and Gurusamy [29], RL-based dynamic service placement can effectively reduce service delay and utilize resources.

The problem of service placement and request scheduling can usually be formulated as an integer linear problem (ILP) or mixed ILP (MILP). Due to the complexity of the problem, the works [14], [15] that focus on optimally solving the problem omit the heterogeneity of the edge servers and the users. In addition, most works [30], [31], [32] typically consider scenarios, such as content placing, computing scheduling, and cache placement. However, few studies have worked on the live-streaming case, leaving an important void that needs to fill.

III. System Model

A. Framework Architecture

We consider an edge network framework as in Fig. 1, which consists of a central cloud server c with enough capacity to

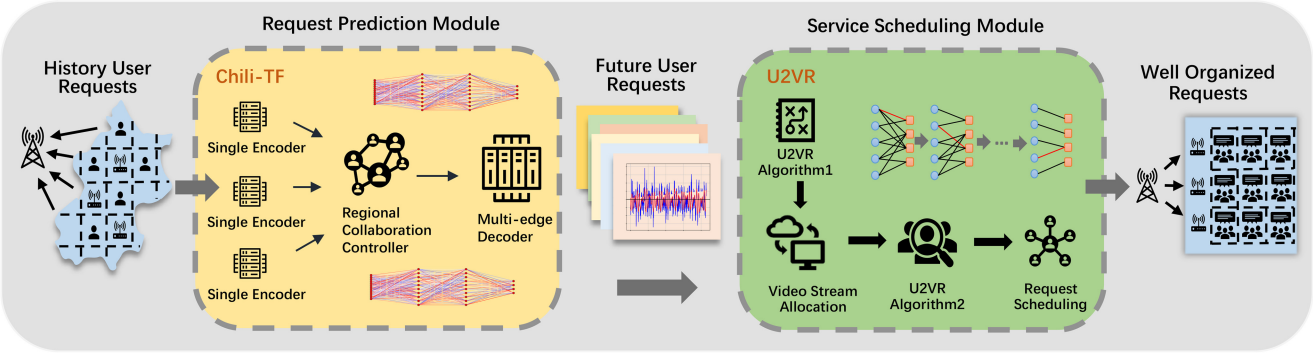

Fig. 2. System framework of Proffler. The system consists of two major modules. The request prediction module takes in the history user requests and makes a prediction. The predicted future requests are fed into the service scheduling module to output optimized scheduling.

store all video streams S, a set E of edges with some streams to provide better service delivery, and a set of users U distributed within the cover range of each edge. The streaming delay perceived by a user is much smaller if served by an edge than by the cloud server, and so is the bandwidth cost since the resource price at an edge is usually cheaper than at the cloud. Practically, each edge e has an inbound bandwidth constraint C_e^{in} and an outbound bandwidth constraint C_e^{out}, which limits the number of video streams an edge can take and the number of users an edge can serve. The inbound bandwidth take-up and outbound bandwidth take-up of an edge e are defined as $\widetilde{C_e^{\text{in}}}$ and $\widetilde{C_e^{\text{out}}}$, respectively.

B. Problem Formulation

Our Proffler system is composed of two major components, as illustrated in Fig. 2, including the request prediction module and the service scheduling module. The live-streaming scenario is highly dynamic, changing the requests in every new time window. The rapidly changing request requires the system to actively adjust the video stream allocation and request scheduling based on the request information.

1) Request Prediction Module: We propose Chili-TF, a module that constructs an improved transformer-based neural network to predict the next-time window users' requests for each edge based on the history of user requests with common viewing patterns across different regions. We formulate the live-video request prediction for a specific video stream in a particular edge e as a multidimensional and multistep supervised machine learning task.

Among K adjacent edges of E, we choose H edges to use their data to do the regional data collaboration and get a time series containing N data points $x_{t-N+1}, \ldots, x_{t-1}, x_t$, for M step ahead prediction, the input X of the supervised ML model is $\mathrm{x}_{t-N+1}, \ldots, \mathrm{x}_{t-M}$, and the output Y is $\mathrm{x}_{t-M+1}, \mathrm{x}_{t-M+2}, \ldots, \mathrm{x}_t$. Each data point x_t can be a scalar or a vector containing the features of multiple user request series.

2) Service Scheduling Module: Once the predicted potential viewing requests are obtained, the system can proceed to the second module, i.e., *the service scheduling module*. We propose a utility-based two-step joint video stream allocation and request scheduling (U2VR) algorithm in the second module to determine which video streams to allocate and whether to redirect local users to another edge in order to achieve a low overall streaming latency bandwidth cost. U2VR first allocates video streams on edges based on the predicted requests. When the new time window arrives, the system can instantly apply the changes to the request scheduling.

We aim to optimally schedule each edge, maximizing the QoEs (i.e., minimizing the streaming delay and bandwidth cost) of users within the resource constraints. In order to solve the problem, we should not only determine which video streams to allocate on each edge, i.e., *video stream allocation*, but also should determine how to assign each user to edges containing the streams requested, i.e., *request scheduling*.

We define $x_s^e \in \{0, 1\}$ $(s \in S, e \in E \cup \{c\})$ to indicate the allocation of service s at edge e (central cloud server is treated as a special edge where $x_s^c = 1, \forall s \in S$), and $y_u^e \in \{0, 1\}$ $(u \in U, e \in E \cup \{c\})$ to indicate whether user u is assigned to edge e. Hence, the video stream allocation can be denoted by set $X = \{x_s^e | s \in S, e \in E \cup \{c\}\}$, and the request scheduling can be denoted by set $Y = \{y_u^e | u \in U, e \in E \cup \{c\}\}$.

We assume a user can request a video stream from either the local edge or the nonlocal edges with a relatively small streaming delay in proportion to the geographical distance from the request edge to the local edge. We adopt a linear function to represent the streaming delay. If no edge can satisfy a user's request, the user will be assigned to the central could server, suffering a huge streaming delay. The streaming delay $\mathcal{L}_{\text{loc}(u),e}$ can be represented as follows:

$$\mathcal{L}_{\text{loc}(u),e} = \begin{cases} \alpha \cdot ||\text{loc}(u), e||, & e \neq c \\ \beta, & e = c \end{cases} \tag{1}$$

where $\text{loc}(u)$ indicates the local edge of the user u. $||\text{loc}(u), e||$ denotes the geographical distance between the user's local edge and the requested edge. α and β are weighted parameters to tune the delay.

Consider that the bandwidth cost of the central cloud server is higher than the edges, the bandwidth cost $\mathcal{W}_{u,e}$ is

$$\mathcal{W}_{u,e} = B_{r_u} \cdot I_e \tag{2}$$

where B_{r_u} is the bandwidth size a video stream r_u takes up and I_e is the unit bandwidth cost of server e $(e \in E \cup \{c\})$.

Combining the variables mentioned above, we have the following optimization objective that minimizes the overall

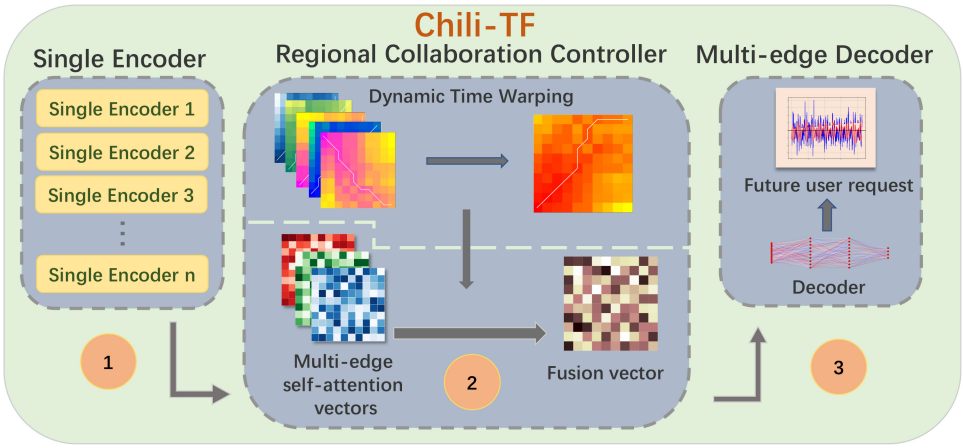

Fig. 3. Overall structure of the Chili-TF module. The output vectors from different Single-Edge Encoders are aggregated through the Regional Collaboration Controller based on the DTW algorithm and then input into the corresponding Multiedge Decoder for the future user requests prediction.

streaming delay and bandwidth cost:

$$\min \quad \sum_{u\in U}\sum_{e\in E\cup\{c\}} \mathcal{L}_{\mathrm{loc}(u),e}\cdot y_u^e + \mathcal{W}_{u,e}\cdot y_u^e \tag{3}$$

$$\text{s.t.} \quad \sum_{s\in S} x_s^e \cdot B_s \le C_e^{\mathrm{in}} \quad \forall e\in E \tag{4}$$

$$\sum_{u\in U} y_u^e \cdot B_{r_u} \le C_e^{\mathrm{out}} \quad \forall e\in E \tag{5}$$

$$x_s^c = 1 \quad \forall s\in S \tag{6}$$

$$\sum_{e\in E\cup\{c\}} y_u^e = 1 \quad \forall u\in U \tag{7}$$

$$y_u^e \le x_{r_u}^e \quad \forall e\in E\cup\{c\}, u\in U \tag{8}$$

$$x_s^e, y_u^e \in \{0,1\} \quad \forall s\in S, e\in E\cup\{c\}, u\in U. \tag{9}$$

Equation (4) guarantees that the video streams allocated on one edge will not exceed its inbound bandwidth constraint. Equation (5) guarantees that the number of users served by one edge will not exceed its outbound bandwidth constraint. Equation (6) indicates that the central cloud server has unlimited resources, which can accept any request. Equation (7) indicates that a user can only be assigned to one edge. Equation (8) ensures that a user can only be assigned to the edge that contains the requested video stream. Equation (9) shows that all the decision variables are indicators.

IV. User Request Prediction

Regions with different geographical locations are composed of different users, which means that potential users' viewing preferences and patterns are also different in the sequence requested by users in the region. In order to maximize the selection of data to participate in the training to obtain higher accuracy prediction, we design Chili-TF, a user request prediction model using a multiedge collaborative learning algorithm. Based on the analysis of user requests' time series among different regions, Chili-TF selects particular regions with similar user composition and user request patterns to participate in the training model under the heterogeneous user data quantity.

A. Overall Module Architecture

The Chili-TF model is a multiregional cooperative transformer-based model. According to Fig. 3, it consists of three components: 1) Single-Edge Encoder; 2) Regional Collaboration Controller; and 3) Multiedge Decoder. The overall function of each component is listed below.

1) *Single-Edge Encoder:* Each edge has a Single-Edge Encoder, which uses the self-attention mechanism to extract time series' periodicity and self-dependence. The input for each edge is the history user requests data in its region, and the output of the Single-Edge Encoder is a vector that contains the extracted temporal features and the regional user video requests pattern.
2) *Regional Collaboration Controller:* The Regional Collaboration Controller compares the self-attention vectors from different edges' Single-Edge Encoder and selects a specific number of edges whose regional user characteristics are most similar based on the dynamic time warping algorithm. Based on the dynamic time warping distance of different regions, Regional Collaboration Controller chooses particular regions with smaller dynamic time warping distances between their self-attention vectors to participate in the model training process. The Regional Collaboration Controller assigns different weights to these vectors according to their similarity and merges them into one vector. The fused vector is transferred to the Multiedge Decoder.
3) *Multiedge Decoder:* The Multiedge Decoder is trained according to the data information from multiple similar edges. It decodes the fused vector output from the Regional Collaboration Controller using the multihead mechanism constructed in the Single-Edge Encoder to do the multidimensional search. Finally, the future user request predicted value for a particular edge is output from the Multiedge Decoder.

B. Single-Edge Encoder

As is shown in Fig. 4, the encoder is composed of a convolution layer, a positional encoding layer, a timestamp encoding

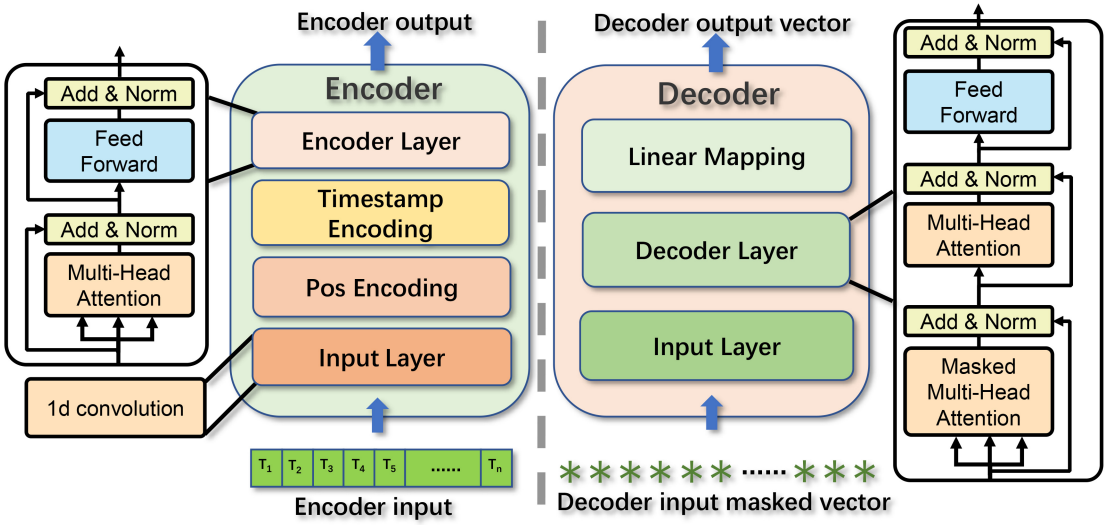

Fig. 4. Structure of the Single-Edge Encoder and the Multiedge Decoder.

layer, and a stack of three encoder layers. The convolution layer transforms the time series data to a vector of dimension d-model through a fully connected network and a $1d$ convolution network. The $1d$ convolution network makes the input time series vector V focus on the neighborhood time step information. The single term $\tilde{v}_t$ for resulting vector is

$$\tilde{v}_t = \text{Conv1}D(v_t). \tag{10}$$

The resulting vector $\tilde{v}_t$ is fed into positional encoding layer. This step is essential for the model to employ a multihead attention mechanism. The positional encoding algorithm used to encode the sequential information of time series data is

$$\text{PE}(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \tag{11}$$

$$\text{PE}(\text{pos}, 2i+1) = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right). \tag{12}$$

However, positional encoding cannot sufficiently capture the periodicity features from highly dynamic user requests in the live streaming scenario. Therefore, we propose a timestamp encoding method that can help the network better capture the periodic features hidden behind the dynamically changing data by adding the value of the periodic sine function concerning the position. The representation for a timestamp encoding vector TE is

$$\text{TE}(t) = \sin\left(\frac{2 * \pi * t}{\text{time\_cycle}}\right)/1000. \tag{13}$$

By combining the positional encoding and the timestamp encoding, the embedded vector X is element-wise addition of the input vector V with a positional encoding vector PE and a timestamp encoding vector TE. The representation for x_t for a time step t is

$$x_t = \tilde{v}_t + \text{PE}(t) + \text{TE}(t). \tag{14}$$

The resulting vector x_t fed into encoder layers consists of a self-attention sublayer and a fully connected feed-forward sublayer. A normalization layer follows each sublayer. The encoder outputs a d model-dimensional vector, which Regional Collaboration Controller uses.

C. Regional Collaboration Controller

Due to the highly dynamic live streaming and the diverse context-dependent request patterns in different edges, it is difficult to capture the hidden features of user requests. However, the user request patterns for live streaming among neighbor edges may have the same features because of the geographical similarity and user mobility among adjacent edges. Therefore, training the prediction model with multiedge data can better capture the user's request characteristics than only using a local request history.

Assume we have E edges in the whole area. We separate a certain edge e from E, and define its feature vector as X. We define the feature vector of the remaining $E-1$ edges e_i as $X_i(i = 1, 2, 3, \ldots, E-1)$. These processed vectors contain the features of self-attention and user request pattern information gained from the raw time series data in different regions. For a certain edge e, the Regional Collaboration Controller selects top N edges whose $X_n(n = 1, 2, 3, \ldots, N)$ is the closest to the X based on the dynamic time warping distance $D(X, X_n)$ to fuse their vectors. The algorithm of counting dynamic time warping distance $D(X, X_n)$ is introduced in the following part.

Suppose Q and C have length n and m, respectively

$$Q = q_1, q_2, \ldots, q_i, \ldots, q_n \tag{15}$$

$$C = c_1, c_2, \ldots, c_j, \ldots, c_m. \tag{16}$$

We constructs an n-by-m matrix, where the (ith, jth) element of the matrix contains the distance $d(q_i, c_j)$ between the two points q_i and c_j. The absolute distance between the values of two sequences is calculated using the Euclidean distance computation

$$d(q_i, c_j) = (q_i - c_j)^2. \tag{17}$$

Each matrix element (i, j) corresponds to the alignment between the points q_i and c_j. Then, the accumulated distance is measured by

$$D(i, j) = \min\left[D(i-1, j-1), D(i-1, j), D(i, j-1)\right] + d(i, j). \tag{18}$$

Based on the value of $D(X, X_n)$, We define the similarity ρ_n of two edges e and e_n as follows:

$$\rho_n = \frac{1}{(1 + D(X, X_n))}. \tag{19}$$

The fusion vector $\tilde{V}$ for edge e which contains the multiedge data feature is

$$\tilde{V} = \rho_1 X_1 + \rho_2 X_2 + \cdots + \rho_n X_n. \tag{20}$$

The fusion vector $\tilde{V}$ is transferred to the Multiedge Decoder for further model training.

D. Multiedge Decoder

The future user request predicted value for a particular edge is output from the Multiedge Decoder by using the fusion vector $\tilde{V}$ containing the multiedge features related to rules of regional user request pattern.

As shown in Fig. 4, the Multiedge Decoder comprises the input, decoder, and output layers. The fusion vector $\tilde{V}$ and the input vector, processed by look-ahead masking, are input into the decoder layers. The decoder layer consists of a ten-head self-attention sublayer, a fully connected feed-forward sublayer, and a normalization layer. Finally, an output layer maps the decoder layer output to the target time sequence with a fully connected network.

V. Joint Video Stream Allocation and Request Scheduling Optimization

A. Problem Hardness

We analyze the complexity of the video stream allocation and request scheduling problems in the heterogeneous case.

Theorem 1: The video stream allocation problem is NP-hard.

Proof: The original problem of video stream allocation is complex since video stream allocation is dependent on request scheduling. We consider a simplified case with a given request scheduling. Consider each video stream as an item and each edge as a knapsack. The simplified problem can be considered as putting items (video streams) of different sizes into multiple knapsacks (edges) with different capacities (bandwidth constraints), maximizing the total profit. We reduce the problem into a multiple knapsack problem (MKP), which is an NP-hard problem. Hence, the video stream allocation problem is NP-hard. ■

Theorem 2: The request scheduling problem is NP-hard.

Proof: Under heterogeneous resource constraints, the request scheduling problem is to assign users to different edges in order to maximize the user's QoE. Consider a special case of the problem that each edge contains all video streams, i.e., the user can be assigned to every edge within the bandwidth constraint. The simplified problem can be considered as putting different items (users) into knapsacks (edges) with different capacities (bandwidth constraints), maximizing the total profit. The simplified problem can be considered as an MKP, which indicates that the original request scheduling problem is NP-hard. ■

Algorithm 1: U2VR

Input: Input parameters of (3) ∼ (9)
Output: Video stream allocation X and user request scheduling Y

$X \leftarrow \emptyset$, $Y \leftarrow \emptyset$;
Sort E from the edge having the most local requests to the edge having the least local requests;
for *each* $e \in E$ **do**
  Sort S from the most requested stream to the least requested stream;
  for *each* $s \in S$ **do**
    if $\widetilde{C_e^{\text{in}}} + B_s \leq C_e^{\text{in}}$ **then**
      $X \leftarrow X \cup x_s^e$;
      $\widetilde{C_e^{\text{in}}} \leftarrow \widetilde{C_e^{\text{in}}} + 1$;
while $K \notin \emptyset$ **do**
  $y^* \leftarrow \emptyset$, $\mathcal{K}^* \leftarrow \emptyset$;
  for *each* $\mathcal{K}_{u,e} \in K$ **do**
    if $\mathcal{K}_{u,e} \geq \mathcal{K}^*$ *and* $\widetilde{C_e^{\text{out}}} + B_{r_u} \leq C_e^{\text{out}}$ **then**
      $\mathcal{K}^* \leftarrow \mathcal{K}_{u,e}$;
      $y^* \leftarrow y_u^e$;
  if $y_u^{e*} \neq \emptyset$ **then**
    $Y \leftarrow Y \cup y^*$;
    Update K;
  else
    Break;

B. Algorithm Design

The proposed U2VR algorithm (Algorithm 1) consists of two steps, decomposing the problem into two subproblems.

1) Step One (Video Stream Allocation): We first resolve the video stream allocation problem in a geographical popularity-based greedy method. Since users served by the local edge would gain the smallest delay, the algorithm will first allocate video streams on the most popular (i.e., with the largest number of requests) edge. The program iteratively allocates video streams on edge from the most popular edge to the least popular one. In each iteration, we allocate the most popular (i.e., the most requested) video streams until the edge reaches its inbound bandwidth constraint. After allocating the video stream, the original problem is now converted to a single variable optimization.

2) Step Two (Request Scheduling): After step one, the problem can be considered as assigning users to different edges within a limited resource constraint. We propose utility gain $\mathcal{K}_{u,e}$ as a measure of the profit the system gains by assigning user u to edge e

$$\mathcal{K}_{u,e} = \lambda \Delta \mathcal{D}_u + \nu \Delta \mathcal{W}_u \tag{21}$$

where $\mathcal{D}_u$ is the streaming delay of user u, and $\mathcal{W}_u$ is the bandwidth cost of user u. By default, $\mathcal{D}_u$ and $\mathcal{W}_u$ are the value gained by the user assigned to the central cloud server. $\Delta \mathcal{D}_u$ and $\Delta \mathcal{W}_u$ represent the improvement of assigning users to a

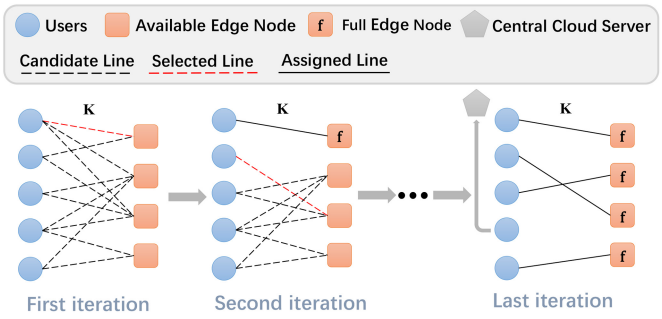

Fig. 5. Auxiliary graph $\mathcal{G}$ for a given video allocation X.

specific edge. The coefficients λ and ν are weighted parameters to tune the utility gain.

Given the utility function, we construct an auxiliary graph $\mathcal{G}$ as in Fig. 5. The nodes in $\mathcal{G}$ consist of a set of nodes U representing each user and a set of nodes E representing each edge. In the beginning, each user is connected to edge nodes containing the video stream that the user requested by dotted lines. The dotted lines are considered as candidate lines, indicating that the connected edge is qualified for serving the connected user. Each candidate line has an attribute $\mathcal{K}_{u,e}$, i.e., the utility gain if assign user u to edge e. We define K as the set containing all the candidate lines. Then, the program starts to iteratively select the dotted line with the highest utility gain. In one iteration, once a line is chosen, it becomes a solid line, while other candidate lines originated from that user node are deleted. Then, the program updates the graph status and starts a new iteration. Note that if multiple candidate lines have the same highest utility gain, the program will choose the first one. If the edge connected to the selected candidate line reaches its resource constraint, the program will select the candidate line with the second-highest utility gain, so on and so forth. When all the edges reach their resource constraints, the rest of the unassigned users will be redirected to the central cloud server, having zero utility gain.

3) Complexity Analysis: Recall that we define S as the video streams, and E as the edge servers. For the first step, the inner for loop is repeated $O(|E| \times |S|)$ times. For the second step, K reaches its maximum when all the users can be assigned to all the edges, i.e., $U \times E$. In the worst case, the edges have infinite capacity, which means no edge will be eliminated during each iteration. Therefore, the inner for loop is repeated $O([(|U|^2 + |U|)/2] \times |E|)$ times, and thus the complexity is $O(|U|^2 \times |E|)$. Therefore, the overall complexity of the algorithm is $O(|E| \times (|S| + |U|^2))$.

VI. Evaluation

This section compares our algorithm with the state-of-the-art algorithms in terms of Request Prediction and Service Scheduling. Our experiment is mainly divided into two components, one is the evaluation of the request prediction module, and one is the evaluation of the integrated framework performance based on the prediction result. We test our algorithm on real data sets.

A. Data Set

We collaborate with iQIYI and collect the user requests for different video streams every 6 min for two weeks in Beijing. We take Haidian District as an example and divide it into 100 edge regions, with each edge covering a 1 km $\times$ 1 km square area. At the same time, each region contains more than 100 video channels, and each channel contains 3360 attributes. To gain more training data, we also use the sliding window method to extend the training data sets. The whole data set is divided into three parts for training, validating, and testing with the scale of 6:2:3.

B. Baseline and Setup

We compared our Chili-TF with the Vanilla Transformer (Vanilla-TF) and LSTM to evaluate the user request prediction performance for a particular edge based on the data sets mentioned in Section V-A, the structure and setup of the baselines and the Chili-TF are listed.

1) *LSTM:* The depth of the LSTM network is 30 Layers. Its activation function is *Relu*. *MSE* is used as the loss function. The batch size is 32 and trained for 60 epochs. The sliding window's length is 100, and the stride is 1.
2) *Vanilla-TF:* The Vanilla Transformer (Vanilla-TF) follows the original structure discussed in this article [12]. It comprises a ten-head encoder and decoder without using the timestamp encoding. Its optimizer is the *AdamW* optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $lrate = 0.00001$. A dropout rate of 0.1 is used for each sublayer. *MSE* is used as the loss function. The batch size is 32 and trained for 60 epochs. The sliding window's length is 100, and the stride is 1.
3) *Chili-TF:* The Chili-TF uses the structure introduced in Section IV. It comprises a ten-head encoder and decoder with a $1d$ convolution layer and a timestamp encoding layer. its optimizer is the *AdamW* optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $lrate = 0.00001$. A dropout rate of 0.1 is used for each sublayer. *MSE* is used as the loss function. The batch size is 32 and the model is trained for 60 epochs. The sliding windows length is 100, and the stride is 1.

In addition to evaluating the predictive accuracy performance of Chili-TF within a certain edge, we also conduct further ablation experiments on regional collaboration strategies. For the edges of a particular region, we adopt four different training strategies.

1) *The Single-Edge Strategy (Single):* It uses only one edge data to train the user request prediction module.
2) *The Random Selection Strategy (RSS):* It randomly selects some user requests data from edges in the region to participate in the model training process.
3) *The Neighborhood Selection Strategy (NSS):* To participate in the model training process, it selects the time series data from edges with adjacent edges with similar geographical locations.
4) The dynamic time warping distance-based selective training strategy (DSS) is introduced in Section IV-C.

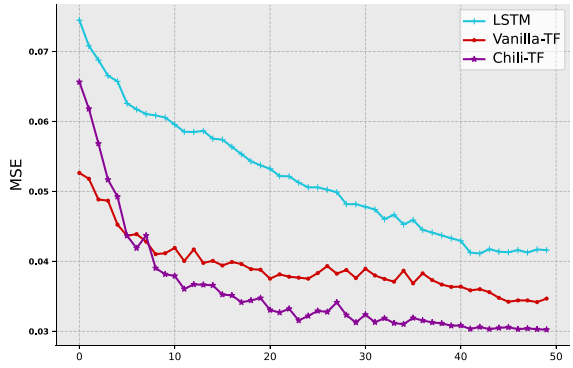

Fig. 6. *MSE* of time series predictions in a single edge.

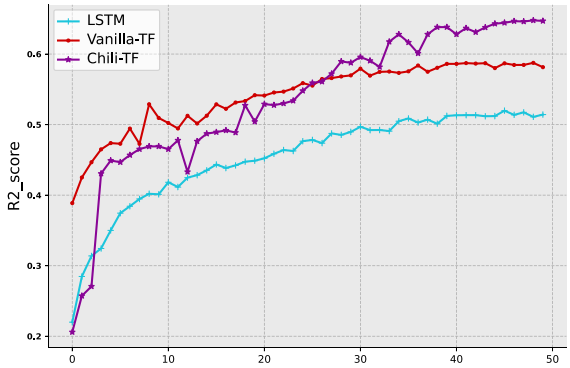

Fig. 7. R^2score of time series predictions in a single edge.

At the same time, we also discuss the influence of the amount of participating edge with our Chili-TF's DSS. We define three types of regions: 1) *small-region* (16 edges); 2) *mid-region* (49 edges); and 3) *large-region* (100 edges). 25% edges closest to the certain edge is chosen, i.e., choosing $16 \times 25\% = 4$ edges trained together for a *small-region*.

Since most existing works resolve the courdsource livecast problem without considering request prediction and service scheduling as a whole, we need to conduct an ablation study by replacing the service scheduling module of the framework. We evaluate `Proffler` against two state-of-the-art algorithms and one other benchmark on top of our Chili-TF prediction module.

1) *Greedy Video Stream Allocation With Greedy Request Scheduling (GSP-GRS):* The GSP-GRS is a state-of-the-art algorithm proposed by He et al. [28]. Within each iteration, the algorithm allocates an additional video stream that serves the maximum number of the previously unserved users using the residual resource. The users that are previously served by the edge remain unchanged.
2) *Greedy Video Stream Allocation With Shadow Request Scheduling (GSP-SS):* GSP-SS is a state-of-the-art algorithm proposed by Farhadi et al. [14]. In each iteration, the algorithm first resolves the shadow scheduling problem of maximizing served users using the linear program method. Then, the algorithm greedily allocates video streams until the edges are full.
3) Random scheduling (RS), which randomly allocates video streams and assigns users to edges.

C. Evaluation on Request Prediction Module

We use two statistical parameters *MSE* and R^2score to measure the performance of our model.

1) *MSE*, which is a measure of the degree of difference between an estimator and an estimator. The closer *MSE* is to zero, the accuracy is better

$$MSE = \frac{1}{n}\sum_{i=1}^{n} w_i(y_i - \tilde{y}_i)^2 \tag{22}$$

where y_i is the true value and $\tilde{y}_i$ is the predicted value. $w_i > 0$, n is the number of samples.

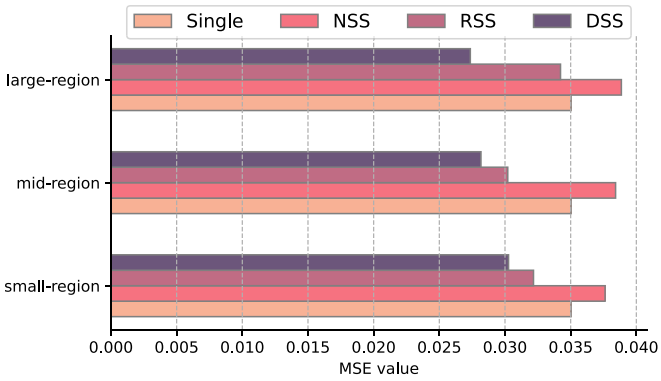

Fig. 8. *MSE* value of Chili-TF with different training strategy.

2) R^2score, also named the coefficient of determination, is the proportion of the variation in the dependent variables that is predictable from the independent variables. The closer R^2score is to one, the accuracy is better

$$R^2\text{score} = 1 - \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y_i})^2} \tag{23}$$

where y_i is the true value, $\tilde{y}_i$ is the predicted value. $\overline{y_i}$ is the mean value of y_i, n is the number of samples.

1) Result of Prediction Accuracy: For the accuracy of time series predictions in a single edge, as shown in Fig. 6, with the gradual convergence of model loss, our Chili-TF could eventually reach a lower *MSE* value than Vanilla Transformer and LSTM network on the test data set, which is as low as 0.03043 on average. The *MSE* is 26.81% lower than LSTM and 15.16% lower than Vanilla Transformer. Fig. 7 shows that the R^2score of Chili-TF is better than the Vanilla-TF and the LSTM network. Even on the high dynamic data, the correlation coefficient can still reach 0.6409, achieving a 24.67% increase over LSTM, and a 9.5% increase over Vanilla-TF.

Our Chili-TF is more accurate than the state-of-the-art time-series forecasting algorithm from the experimental data. It can significantly improve prediction accuracy in the mean square error and correlation coefficient of the predicted time series value. Therefore, it can better solve the problem of user request prediction in a particular region.

2) Ablation Study: As for the ablation study of regional collaborative strategy, the result is shown in Fig. 8. By calculating and ranking the dynamic time warping distance using

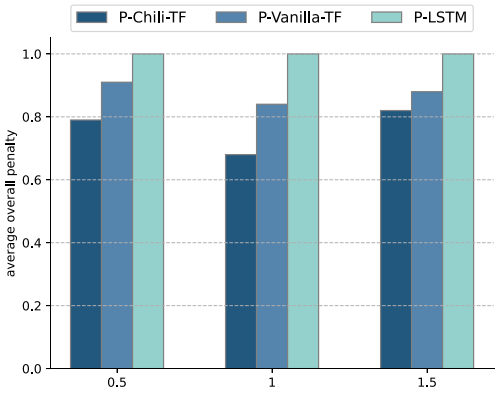

Fig. 9. Normalized overall penalty under different prediction model.

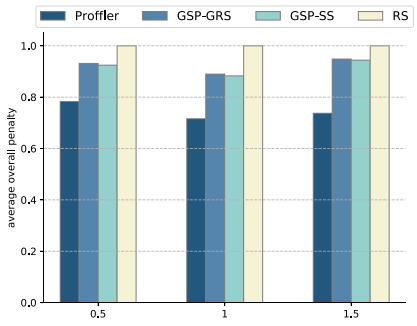

Fig. 10. Normalized overall penalty under different utility ratios.

the time sequence obtained by different edges, our regional collaboration framework can obtain the lowest *MSE* in both three scale conditions, which means it performs with the highest accuracy.

Our DSS can achieve an average MSE of 0.03027, 0.02817, and 0.02736 under *small-region* (16 edges), *mid-region* (49 edges), and *large-region* (100 edges), which has an average *MSE* error decline of 13.59%, 19.58%, and 21.90%, respectively, compared with Single, an average *MSE* error decline of 19.54%, 26.68%, and 29.59%, respectively, compared with RSS, and an average *MSE* error decline of 5.91%, 6.75%, and 20.06%, respectively, compared with NSS.

3) Impact on Overall Performance: To demonstrate the performance of Chili-TF on the integrated framework, we assess the QoE provided by `Proffler` with different prediction models (i.e., P-Chili-TF, P-Vanilla-TF, and P-LSTM). We select a region containing 16 edges and 2000 users and evaluate the overall penalty under different utility ratios (detailed explanation of utility ratio in Section VI-D). To better compare the results, we set the LSTM prediction model as the baseline and normalize the results of other algorithms accordingly. Fig. 9 shows that `Proffler` integrated with Chili-TF achieves the best penalty reduction among other prediction models, resulting in an average penalty reduction of 23.67% compared to the baseline.

The result proves that our user request prediction module, which trains Chili-TF by using DSS, does an excellent job in the prediction of time series between an extensive range of edges, achieving higher accuracy than other existing state-of-the-art algorithms. Chili-TF achieves a reduction of 29.59% *MSE* and an increase of 24.67% R^2score. Our model also has a better performance improvement with the rise of the regional scale. In addition, our overall framework integrated with Chili-TF achieves better QoE than existing prediction models. Therefore, Chili-TF, the user request prediction module of `Profller`, is proved to be robust and successful in time series prediction tasks for user requests in multidata regions.

D. Evaluation on Integrated Framework

We carry out evaluations to analyze the performance of `Proffler` under different settings.

TABLE I
CAPACITY OF DIFFERENT EDGES

Edge	C_e^{in}	C_e^{out}
small-edge	45∼55Mbps	350∼450Mbps
mid-edge	55∼65Mbps	450∼550Mbps
large-edge	65∼75Mbps	550∼650Mbps

1) Impact of Utility Ratio: Different ratios between the tuning parameters of streaming delay gain and bandwidth cost gain (i.e., λ/ν) will influence the performance of `Proffler`. A higher λ/ν ratio will result in higher priority for reducing streaming delay and vice versa. We select a region containing 16 edges and 2000 users as the default region. In this experiment, we define each edge as *small-edge*. To better compare the results, we set the RS algorithm as the baseline and normalize the results of other algorithms accordingly.

Fig. 10 shows the average overall penalty normalized by the random selection algorithm. We can obtain that `Proffler` outperforms the other three algorithms over all different settings, reducing the overall penalty by 21.7%, 28.4%, and 26.3% when λ/ν is 0.5, 1, and 1.5, respectively. The ratio corresponding to the highest performance indicates that the incentive of choosing different types of penalties should be balanced in order to achieve the best performance.

2) Impact of Edge Capacity: Apart from the utility ratio, the edge capacity can also impact the performance of the system. We consider three different settings of edge capacities, *small-edge*, *mid-edge*, and *large-edge* (the capacity of each edge is shown in Table I). We focus on the default region and set $\lambda/\nu = 1$.

According to Fig. 11, `Proffler` shows notably better performance than other algorithms under all edge settings. From *small-edge* to *large-edge*, U2VR reduces overall penalty by 21.7%, 30.8%, and 35.6% compared with the baseline, respectively. The result shows that an edge with a larger capacity enables U2VR to assign more users with higher utility, further lowering the overall penalty.

3) Impact of Scale: We compare regions with different scales to evaluate the scalability of `Proffler`. We select three regions covering different scales of area, i.e., *small-region*, *mid-region*, and *large-region* (the scale of each region is shown in Table II). We consider all edges as *small-edge* and set $\lambda/\nu = 1$.

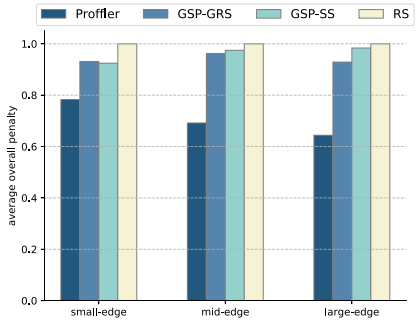

Fig. 11. Normalized overall penalty under different edge capacities.

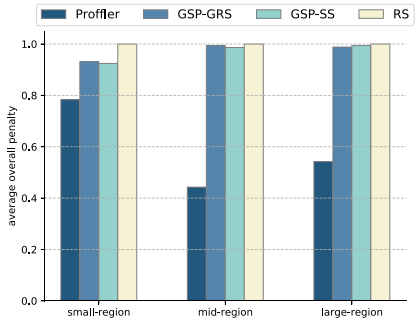

Fig. 12. Normalized overall penalty under different scales.

TABLE II
SCALE OF DIFFERENT REGIONS

Region	Edge numbers	Edge capacity
small-region	16	small-edge
mid-region	49	small-edge
large-region	100	small-edge

Fig. 12 shows that `Proffler` achieves superior performance among other algorithms, reducing the overall penalty by 21.7%, 55.7%, and 45.8% under different scales. From the figure, we can observe that as the scale of the region increases, GSP-GRS and GSP-SS can hardly reduce the overall penalty, while `Proffler` performs even better on large-scale regions, proving the scalable nature of `Proffler`. Since the scale of the live streaming environment is highly diverse under different circumstances, `Proffler` is proven to be an effective method of solving the service scheduling problem.

VII. CONCLUSION

Crowdsourced livecast is a promising way of improving the fast-developing immersive multimedia community. However, it invokes the problems of optimal video stream allocation and request scheduling. In this article, we proposed `Proffler`, an efficient framework combining a novel prediction model and a utility-based algorithm that achieves collaborative viewing request prediction across different regions and scalable video stream allocation as well as viewer scheduling in edge-assisted crowdsourced livecast. We first proposed the Chili-TF prediction module to predict future user requests based on the history requests in different edges. Then, we formulated a QoE-driven optimization problem that minimizes the overall streaming delay and bandwidth cost. We then proposed the U2VR algorithm to resolve the service scheduling problem based on the accurate request prediction. Extensive experiments showed that the algorithm can achieve superior performance.

REFERENCES

[1] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.

[2] A. Kamilaris and A. Pitsillides, "Mobile phone computing and the Internet of Things: A survey," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 885–898, Dec. 2016.

[3] G. Appel, L. Grewal, R. Hadi, and A. T. Stephen, "The future of social media in marketing," *J. Acad. Market. Sci.*, vol. 48, no. 1, pp. 79–95, 2020.

[4] Z. Lv, D. Chen, R. Lou, and H. Song, "Industrial security solution for virtual reality," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6273–6281, Apr. 2020.

[5] Y. Han et al., "A dynamic hierarchical framework for IoT-assisted digital twin synchronization in the metaverse," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 268–284, Jan. 2023.

[6] J. Nightingale, P. Salva-Garcia, J. M. A. Calero, and Q. Wang, "5G-QoE: QoE modelling for ultra-HD video streaming in 5G networks," *IEEE Trans. Broadcast.*, vol. 64, no. 2, pp. 621–634, Jun. 2018.

[7] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 2, pp. 112–121, Apr. 2014.

[8] G. Premsankar, M. D. Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[9] C. Li, J. Liu, and S. Ouyang, "Characterizing and predicting the popularity of online videos," *IEEE Access*, vol. 4, pp. 1630–1641, 2016.

[10] S. Li et al., "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 5243–5253.

[11] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," 2020, *arXiv:2001.08317*.

[12] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[13] F. Wang et al., "DeepCast: Towards personalized QoE for edge-assisted crowdcast with deep reinforcement learning," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1255–1268, Jun. 2020.

[14] V. Farhadi et al., "Service placement and request scheduling for data-intensive applications in edge clouds," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 779–792, Apr. 2021.

[15] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE INFOCOM*, 2018, pp. 207–215.

[16] N. Yu, Q. Xie, Q. Wang, H. Du, H. Huang, and X. Jia, "Collaborative service placement for mobile edge computing applications," in *Proc. IEEE GLOBECOM*, 2018, pp. 1–6.

[17] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Commun. Mag.*, vol. 57, no. 6, pp. 114–119, Jun. 2019.

[18] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, "GeoMAN: Multi-level attention networks for geo-sensory time series prediction," in *Proc. IJCAI*, 2018, pp. 3428–3434.

[19] R. Geng, X. Wang, N. Ye, and J. Liu, "A fault prediction algorithm based on rough sets and back propagation neural network for vehicular networks," *IEEE Access*, vol. 6, pp. 74984–74992, 2018.

[20] U. Kose and A. Arslan, "Forecasting chaotic time series via ANFIS supported by vortex optimization algorithm: Applications on electroencephalogram time series," *Arab. J. Sci. Eng.*, vol. 42, no. 8, pp. 3103–3114, 2017.

[21] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.
[22] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing*, vol. 323, pp. 203–213, Jan. 2019.
[23] Y. Li, Z. Zhu, D. Kong, H. Han, and Y. Zhao, "EA-LSTM: Evolutionary attention-based LSTM for time series prediction," *Knowl.-Based Syst.*, vol. 181, Oct. 2019, Art. no. 104785.
[24] Z. Karevan and J. A. K. Suykens, "Transductive LSTM for time-series prediction: An application to weather forecasting," *Neural Netw.*, vol. 125, pp. 1–9, May 2020.
[25] H. Zhou et al., "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI*, 2021, pp. 11106–11115.
[26] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *Proc. IEEE INFOCOM*, 2019, pp. 1468–1476.
[27] S. Pasteris, S. Wang, M. Herbster, and T. He, "Service placement with provable guarantees in heterogeneous edge computing systems," in *Proc. IEEE INFOCOM*, 2019, pp. 514–522.
[28] T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein, "It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources," in *Proc. IEEE 38th ICDCS*, 2018, pp. 365–375.
[29] A. Talpur and M. Gurusamy, "Reinforcement learning-based dynamic service placement in vehicular networks," in *Proc. IEEE 93rd (VTC-Spring)*, 2021, pp. 1–7.
[30] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. IEEE INFOCOM*, 2019, pp. 1459–1467.
[31] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.
[32] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu, "Delay-aware virtual network function placement and routing in edge clouds," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 445–459, Feb. 2021.

Wenyi Zhang received the B.Eng. degree in computer engineering from The Chinese University of Hong Kong, Shenzhen, China, in 2023. He is currently pursuing the M.S. degree in ECE with the University of California San Diego, San Diego, CA, USA.

His research interests include multimedia streaming, volumetric video, satellite networks, and edge computing.

Zihan Xu received the B.Eng. degree in computer engineering from The Chinese University of Hong Kong, Shenzhen, China, in 2023. He is currently pursuing the M.S. degree in ECE from Carnegie Mellon University, Pittsburgh, PA, USA.

His research interests include multimedia streaming, edge computing, satellite networks, and volumetric video.

Fangxin Wang (Member, IEEE) received the B.Eng. degree in computer science and technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2013, the M.Eng. degree in computer science and technology from Tsinghua University, Beijing, in 2016, and the Ph.D. degree in computer science and technology from Simon Fraser University, Burnaby, BC, Canada, in 2020.

He is an Assistant Professor with The Chinese University of Hong Kong, Shenzhen (CUHKSZ), China. Before joining CUHKSZ, he was a Postdoctoral Fellow with the University of British Columbia, Vancouver, BC, Canada. He lead the Intelligent Networking and Multimedia Lab, CUHKSZ. He has published more than 30 papers at top journals and conference papers, including INFOCOM, *Virtual Reality*, *Multimedia Journal*, IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, and IEEE INTERNET OF THINGS JOURNAL. His research interests include multimedia systems and applications, cloud and edge computing, deep learning and big data analytics, and distributed networking and system.

Dr. Wang served as the Publication Chair of IEEE/ACM IWQoS, a TPC Member of IEEE ICC, and a Reviewer of many top conference and journals, including INFOCOM, *Multimedia Journal*, IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, and IEEE INTERNET OF THINGS JOURNAL.

Jiangchuan Liu (Fellow, IEEE) received the B.Eng. degree (cum laude) from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from Hong Kong University of Science and Technology, Hong Kong, in 2003.

He is currently a Full Professor with the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada.

Prof. Liu is a co-recipient of the Test of Time Paper Award of IEEE INFOCOM in 2015, the ACM TOMCCAP Nicolas D. Georganas Best Paper Award in 2013, and the ACM Multimedia Best Paper Award in 2012. He is a Steering Committee Member of IEEE TRANSACTIONS ON MOBILE COMPUTING, and an Associate Editor of IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON BIG DATA, and IEEE TRANSACTIONS ON MULTIMEDIA. He is a Fellow of the NSERC E. W. R. Steacie Memorial and the Canadian Academy of Engineering.