EdgeURB: Edge-driven Unified Resource Broker for Real-time Video Analytics

Xiaojie Zhang[*], Amitangshu Pal[†], Saptarshi Debroy[*]
[*]City University of New York, New York, NY, USA; [†]Indian Institute of Technology Kanpur, Kanpur, India
Email: *xzhang6@gradcenter.cuny.edu, amitangshu@cse.iitk.ac.in, saptarshi.debroy@hunter.cuny.edu*

NOMS 2024-2024 IEEE Network Operations and Management Symposium | 979-8-3503-2793-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/NOMS59830.2024.10575131

Abstract—**Real-time video analytics applications are one of the driving forces towards adoption of edge computing due to the latter's ability to provide 'near cloud-scale' resources closer to the application site. However, striking a balance between system energy-efficiency and video quality satisfaction still remains a challenge. In this paper, we propose an edge-driven unified resource broker (URB) framework, viz., *EdgeURB* that seeks to find a trade-off between edge devices' energy-efficiency and video configuration adaptation, with an aim to satisfying the real-time latency requirements without compromising analytics accuracy. Particularly, we design a two-stage algorithm: 1) a centralized algorithm for resource allocation, frame resolution selection, and user device to sever assignment and 2) a strategic game to further improve the energy-efficiency. We evaluate the performance of *EdgeURB* framework using an edge hardware testbed prototype that demonstrates *EdgeURB*'s success in simultaneously satisfying application latency, analytics accuracy, and devices' energy consumption requirements. Also, through extensive simulations, we demonstrate *EdgeURB*'s schedulability and scalability improvement over baseline algorithms for a large number of devices and for varying edge resource availability.**

Index Terms—**Edge computing, resource management, real-time video analytics, energy efficiency, joint optimization.**

I. Introduction

In recent years, machine learning (ML) driven video analytics have become one of the killer applications in mission-critical fields such as, emergency management [1], [2], autonomous vehicles (AV) [3], and surveillance system [4]. Since these applications often have low/ultra-low latency requirements, processing such video data locally on user IoT devices, although desirable, is far from realistic. Nevertheless, massive data transmission and privacy requirements make cloud data-center based processing imparactical. Therefore, edge computing has recently emerged as an ideal compromise between local device and remote cloud based processing. However, effective edge-driven video analytics is fraught with design challenges.

Two such most important challenges are: a) energy-efficient resource management and b) video configuration adaptation, that can satisfy the analytics quality and real-time latency requirements. These challenges are more formidable when the system is required to optimize a set of often mutually diverging performance metrics such as, network delay, computation latency, accuracy of the analytics outcome, and system energy consumption. This is because: 1) Users' preferences over edge resources and video configurations are different as they can have diverse video application requirements; 2) The wireless channel quality and consequently radio transmission characteristics experienced by users (i.e., their devices) can vary greatly resulting in varying performance; and 3) Joint optimization of such often mutually diverging metrics is typically NP-hard and thus compute-intensive. *However, edge systems with their limited resource budgets can only support light-weight solutions that do not impede on hosted applications' performance.*

In this paper, we propose an edge-driven unified resource broker (URB) framework, viz., *EdgeURB* that can jointly address the two aforementioned challenges through energy-efficient edge resource management and video configuration adaptation. The proposed *EdgeURB* framework collects and uses the edge server utilization information, network bandwidth condition, application profiling results, and energy consumption statistics of user devices to run a two-stage algorithm with the aim of addressing the trade-off between system energy consumption and the accuracy of video analytics outcome under given latency constraints. In the first stage, a centralized algorithm utilizes linear programming relaxation and Lagrangian method to obtain a valid initial, yet sub-optimal solution for resource allocation, frame resolution selection, and sever selection problems. While in the second stage, *EdgeURB* formulates a strategic game to further improve the efficiency of the initial user-server assignment. This stage decouples the original problem into two interconnected sub-problems that solve resource allocation and user-server assignment, respectively. The resource allocation algorithm runs on individual edge servers and the optimized results are passed to the user-server assignment algorithm. Here, a bandwidth-efficiency based priority mechanism is adopted to update the server selection of the users. Finally, *EdgeURB* broadcasts the updated global strategy profile to the edge servers when the framework runs the resource allocation algorithm again. This inter-exchange between algorithms terminates and converges when the system reaches a Nash Equilibrium (NE) - thus achieving optimal energy-efficiency.

For evaluation, we design an edge hardware testbed prototype and implement the *EdgeURB* framework along with services such as, server discovery, user identification, control information exchange, and virtual-to-physical resource mapping [5]. Experiments validate *EdgeURB*'s effectiveness in terms of latency satisfaction, resource allocation, and resolution adaptation for varying edge resource availability. The results demonstrate the success of the framework in successfully satisfying the latency, application outcome accuracy, energy consumption requirements of heterogeneous applications. To add more scalability that the testbed can provide, we perform extensive simulations to demonstrate *EdgeURB* schedulability and scalability for a large number of user devices and for varying edge resource availability. *EdgeURB* comparison results against traditional uni-dimensional algorithms show that *EdgeURB* is able to increase the schedulability between 33.5% to 58% with average utility improvement of 10%, on a case to case basis. The results also demonstrate *EdgeURB*'s upto 36% analytics' accuracy improvement over baseline algorithms at the cost of merely 6% increase in energy consumption.

The rest of the paper is organized as follows. Section II presents the related work. Section III proposes the system model and problem formulation. Section IV presents the resource allocation algorithm. Section V discusses the user-server assignment. Section VI discusses testbed and simulation. Section VII concludes the paper.

This material is partially supported by the National Science Foundation under Award Number: CNS-1943338.

II. Related work

Real-time video analytics: The performance of video analytics is jointly determined by orchestrating resource allocation and configuration parameters and there exists a group of work seeking the trade-off between accuracy and the cost (i.e., latency and energy consumption). Authors in [6] optimize the selections of a variety of configurations, including frame sample rate, frame resolution, video bitrate, and model variant. In [7], the authors propose an edge-based orchestrator for Mobile Augmented Reality (MAR). This work aims to find the optimal server assignment and frame resolution in terms of optimizing the trade-off between the service latency and analytics accuracy. On the other hand, [6], [7] follow a simple network model and fair computation resource allocation. While [8] proposes a bandwidth-efficiency framework which jointly optimizes resource allocation and configuration adaptation, the authors only consider bandwidth allocation in single server use cases. *Above work mostly focus on configuration adaptation rather than resource allocation, whereas our problem aims at solving joint optimization of resource allocation and configuration adaptation in multi-server scenarios.*

Joint optimization in edge systems: Joint optimization problems on edge video analytics are usually NP-hard due to their discrete and combinatorial elements such as server selection, channel allocation, and application model selection [9]. Typically, it is challenging and time-consuming to find the optimal solution to such problems. Works such as [10], [11] utilize bi-level optimization method to decouple discrete elements from the original problems. Authors in [10] apply a two-sided matching game to implement sub-channel allocation, while [11] uses Hungarian method to find the optimal offloading strategy. The authors in [12] perform a device classification and priority determination strategy to solve task offloading. *However, few works have considered the characteristics of video analytics applications which is precisely our problem environment.*

III. System Model and Empirical Motivation

For this work, we assume an edge environment consisting of a set of edge servers $\mathcal{K} = \{1,2,...,K\}$ where each server has a certain amount of radio and computational resources. We assume that multiple end-users $\mathcal{N} = \{1,2,...,N\}$ equipped with camera enabled devices simultaneously capture videos of a target scene and live stream the video feed to chosen edge server(s) for real-time video analytics. The environment also consist of centralized URB, hosted by one of the edge servers and responsible for radio and computational resource allocation for the real-time video processing. In this paper, we describe the server selection problem as user-server assignment, i.e., $a_{n,k} = \{0,1\}$, where $a_{n,k} = 1$ indicates that the n-th user is assigned to the k-th edge server; otherwise, $a_{n,k} = 0$. Since one user can be assigned to only one server, we have $\sum_{k=1}^{K} a_{n,k} = 1, \forall n \in \mathcal{N}$. In addition, we consider a time-slotted optimized system where each time slot $t \in \{1,2,3,...T\}$ is further divided into transmission and computation periods.

A. Application Model

In order to support live streaming, we assume that user devices split their video streams into a sequence of video segments/chunks. We also assume that the frames in each segment need to be transmitted and processed within a single time slot. We denote the number of frames in each segment as s_n. The latency requirement is measured by s_n/τ frames per second (fps) which indicates the constraint of the video output (e.g., 5 fps). Additionally, we assume that the users care about the accuracy of analytics outcome and device energy consumption. We assume that the users are requesting different ML models $m_n \in \mathcal{M} = \{1,2,...,M\}$ and video frame resolutions $u_n \in \mathcal{U} = \{128\times128,...,640\times640\}$ with some specific performance requirements. Ideally, these configurations should be adjusted according to the actual edge resource availability.

B. Application Benchmarking: An Edge System Prototype

To capture the characteristics of the video analytics' performance metrics in a realistic edge environment, we design an edge system prototype with one edge server implemented on a Dell desktop equipped with NVIDIA GeForce RTX 2060 (1920 CUDA cores). The edge server also has a wireless component, an eNodeB (LTE base station) installed on a Dell desktop equipped with Intel i7-10700k @3.8GHz and 32GB RAM with an Ettus B210 USRP device implementing the RF front-end using OpenAirInterface (OAI) and OpenAir-CN platforms. The edge server and the eNodeB are connected through a 1 Gbps Ethernet cable mimicking 'single-unit' setup. A Nokia 2.2 Android smartphone is used as the user device which continuously sends the video frames to the edge server through the allocated sub-channels (from eNodeB). For the video frames, we use COCO 2014 video dataset that is pre-installed on the end-device.

To study the accuracy of user selected ML models, the edge server runs a well-known object detection algorithm, viz., YOLOv5. For the benchmarking experiments, we test four such models of YOLOv5 (e.g., *v5x, v5l, v5m, v5s*). We obtain the voltage (V) and current (A) readings of the end-user device in real-time by using Android Battery Manager and estimate the energy consumption in Joules (J) during the transmission period.Overall, we measure the network delay and energy consumption of user devices when total channel bandwidth is adjusted to 1.8 MHz, 3.6 MHz, and 5.4 MHz. We also run the same experiments under a Wi-Fi/2.4 GHz (802.11n) environment (instead of LTE) where the bandwidth is 20 MHz in order to establish the universality of the results.

C. Analytics Accuracy and Computation Latency Models

We define analytics accuracy as the average precision $_{IOU_{0.5}}$ (intersection over union). The accuracy results of YOLOv5 against different frame resolutions (measured by $u_n \times u_n$) are shown in Fig. 1(a). As expected, higher resolution leads to better accuracy and for fixed frame resolution, different models have different accuracies. The figure demonstrates that the relationship between accuracy and video frame resolution can be fitted into a concave function. Here, the dashed lines indicate the fitted accuracy with mean squared error (MSE) less than 0.01. Based on the observations, we characterize the accuracy of a video segment for a given model m_n as:

$$\Phi^{i}(u_n;m_n) = s_n(\alpha^{i_1}_{m_n} - \frac{\alpha^{i_2}_{m_n}}{u_n^{\alpha^{i_3}_{m_n}}})$$

where the functions have different convergence features and maximum values. We also observe that the YOLOv5 computation latency increases as frame resolution increases with larger models consuming longer computation times (in Fig. 1(b)). Thus, such relationship can be fitted into a convex function. Thus, the fitted latency follows:

$$l^{t}_{n,k} = \begin{cases} s_n \frac{\Phi^{c}(u_n;m_n)}{y_{n,k}} & a_{n,k} = 1, y_{n,k} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\Phi^{c}(u_n;m_n) = \alpha^{c_1}_{m_n} u_n^3 + \alpha^{c_2}_{m_n}$ is the computational complexity w.r.t. the video frame resolution u_n and model m_n. The amount of shared computational resources is denoted by $y_{n,k}$ with constraint $\sum_{n\in\mathcal{N}} y_{n,k} = Y_k, \forall k \in \mathcal{K}$.

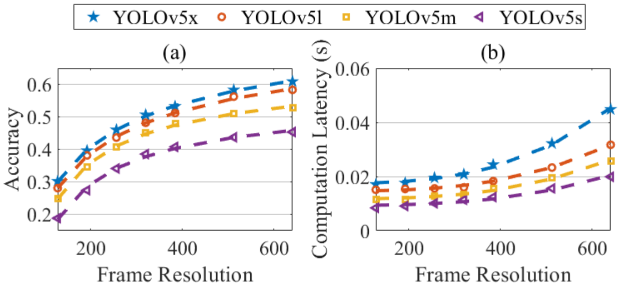

Fig. 1: (a) Model accuracyand (b) Computation latency

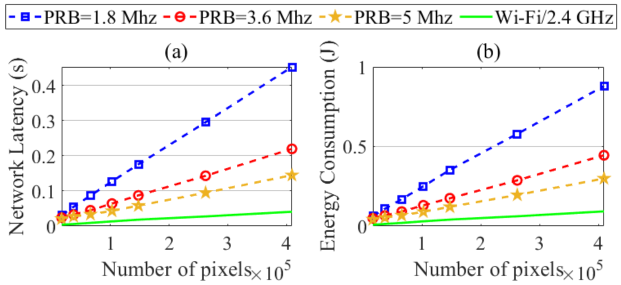

Fig. 2: (a) Network latency and (b) Energy consumption per frame

D. Network Latency and Energy Consumption Models

For modeling the network latency, we first define $\eta_{n,k}$ as the Signal-to-interference-plus-noise ratio (SINR) of the communication between the n-th user device and the k-th edge server (i.e., its eNodeB). Thus, the achievable data rate is:

$$r_{n,k} = x_{n,k} \log_2(1+\eta_{n,k}), \forall n \in \mathcal{N}, \forall k \in \mathcal{K} \qquad (2)$$

where the amount of shared network bandwidth is denoted by $x_{n,k}$ with the constraint $\sum_{n\in\mathcal{N}} x_{n,k} = X_k, \forall k \in \mathcal{K}$ (i.e., each edge server has different amount of available bandwidth X_k). We assume that for a device, the average data rate is fixed under the given bandwidth as long as the SINR between the user device and the edge server remains unchanged.

The results of per frame network latency are shown in Fig. 2(a). The number of pixels is measured by u_n^2 (i.e., width × height). As expected, we notice that the network latency increases linearly with the number of pixels carried by each frame for a given data rate. Therefore, the average bits of frames to resize with resolution $u_n \times u_n$ can be represented as $\Phi^d(u_n) = \alpha_d^1 u_n^2 + \alpha_d^2$, where α_d^1 and α_d^2 are constants [8]. On the other hand, network latency is inversely proportional to the data rate $r_{n,k}$. Now since the user device transmits an entire video segment to the edge server in each time slot, the size of the segment s_n determines the overall networking latency. Based on these observations, our network latency model $l_{n,k}, \forall n \in \mathcal{N}$ is defined as:

$$l_{n,k}^t = \begin{cases} s_n \frac{\Phi^d(u_n)}{x_{n,k}\log_2(1+\eta_{n,k})} & a_{n,k}=1, x_{n,k}>0 \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

where we consider the value of SINR (i.e., $\eta_{n,k}$) as a constant during the optimization (e.g., users mobility is limited while sending the video frames). However, the SINR can change when the user device connects to other edge servers.

Nevertheless, energy consumption results, as shown in Fig. 2(b), share the same characteristics of network latency in terms of the number of pixels and the bandwidth. Since the energy consumption can be interpreted as the transmission power times the transmission time, we model energy consumption $e_{n,k}, \forall n \in \mathcal{N}$ as:

$$e_{n,k} = \begin{cases} s_n\left(\frac{\Phi^d(u_n)}{x_{n,k}\log_2(1+\eta_{n,k})} p_n\right) & a_{n,k}=1 \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

where p_n is the unit transmission energy consumption (i.e., $p_n = V_n \times A_n$) of the user device (in Watts). It is assumed that user devices can have different unit energy consumption.

E. Problem Formulation

Given a video analytics model m_n and frame resolution u_n, the end-to-end latency to process a video segment with s_n frames is expressed as:

$$l_{n,k} = \begin{cases} s_n\left(\frac{\Phi^d(u_n)}{x_{n,k}\log_2(1+\eta_{n,k})} + \frac{\Phi^c(u_n;m_n)}{y_{n,k}}\right) & a_{n,k}=1 \\ 0 & \text{otherwise} \end{cases} \qquad (5)$$

In order to model the trade-off between analytics accuracy and energy consumption, we introduce β as the desired balance factor between energy consumption and accuracy. The multi-objective optimization problem, thus, can be formulated as:

$$\min \sum_{n=1}^{N}\sum_{k=1}^{K} a_{n,k}\left(\beta e_{n,k} - \Phi^i(u_n;m_n)\right)$$

$$\text{s.t. } \mathbf{C1:} \sum_{n=1}^{N} a_{n,k} y_{n,k} = Y_k, \forall k \in \mathcal{K}$$

$$\mathbf{C2:} \sum_{n=1}^{N} a_{n,k} x_{n,k} = X_k, \forall k \in \mathcal{K}$$

$$\mathbf{C3:}\ a_{n,k} \in \{0,1\}, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}$$

$$\mathbf{C4:} \sum_{k=1}^{K} a_{n,k} = 1\ \forall n \in \mathcal{N}, \forall k \in \mathcal{K}$$

$$\mathbf{C5:} \sum_{k=1}^{K} a_{n,k} l_{n,k} \leq \tau, \forall n \in \mathcal{N}$$

$$\mathbf{C6:}\ u_n \in \{u^{min}, \ldots, u^{max}\}, \forall n \in \mathcal{N} \qquad \mathbf{(P1)}$$

In *EdgeURB*, we propose a two-stage solution for (**P1**) which is a classic NP hard Mixed-Integer Nonlinear Programming (MINLP) problem. In the 1^{st} stage, we convert **P1** into a convex integer relaxation and use a naive rounding method to achieve an initial user-server assignment. While in the 2^{nd} stage, we divide it into two sub-problems as:

$$\overbrace{\underset{\mathbf{a}}{\text{argmin}} \left\{ \underbrace{\underset{\mathbf{x,y,u}}{\text{argmin}} \sum_{n=1}^{N}\sum_{k=1}^{K} a_{n,k}\left(\beta e_{n,k} - \Phi^i(u_n;m_n)\right)}_{\textbf{Resource Allocation (RA)}} \right\}}^{\textbf{User-Server Assignment (UA)}} \qquad (6)$$

The inner sub-problem of joint network and computational resource allocation (RA) aims to find the optimal resource allocation and frame resolution strategies that minimizes the cost for all users. Whereas, the outer sub-problem of user-server assignment (UA) is a strategic game where users are modeled as selfish in nature and are always willing to select the best edge server to minimize their own cost. The solution to the 2^{nd} stage of **P1** is obtained by performing RA and UA alternatively and iteratively until convergence. We denote these two sub-problems as $\mathcal{P}(a,\mathbf{x},\mathbf{y},\mathbf{u})$ and $\mathcal{P}(\mathbf{a},x,y,u)$, respectively.

IV. RA: Joint Network and Computational

In this section, we first relax the video frame resolutions $u, \forall n \in \mathcal{N}$ to continuous variables $\hat{u}, \forall n \in \mathcal{N}$. Once the optimal $\hat{u}^*$ is obtained, we convert it back the nearest integer resolution (defined in $\mathcal{U}$) that does not violate the constraint **C5**. This makes $\mathcal{P}(a,\mathbf{x},\mathbf{y},\hat{\mathbf{u}})$ a convex optimization problem in terms of resource allocation and resolution selection:

$$\mathcal{P}(a,\mathbf{x},\mathbf{y},\hat{\mathbf{u}}) : \underset{x,y,\hat{u}}{\text{argmin}} \sum_{n=1}^{N}\sum_{k=1}^{K} a_{n,k}\left(\beta e_{n,k} - \Phi^i(\hat{u_n};m_n)\right)$$

$$s.t.\ \mathbf{C1}, \mathbf{C2}, \mathbf{C5}$$

$$\mathbf{C6}: \hat{u} \in [u^{min}, u^{max}]$$

A. Convexity of $\mathcal{P}(a,\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{u}})$

Once UA a is fixed, we observe that the Hessian matrix of $\mathcal{P}(a,\mathbf{x},\mathbf{y},\hat{\mathbf{u}})$, being positive semi-definite, makes the problem convex w.r.t. variables $x_{n,k}$, $y_{n,k}$, and $\hat{u_n}$. Therefore, applying the Karush-Kuhn-Tucker (KKT) conditions yields the optimal solution. We first introduce a set of Lagrangian multipliers Θ: $\{\boldsymbol{\lambda},\ \boldsymbol{\nu},\ \boldsymbol{\mu}\}$ that are associated with network resource allocation, computation resource allocation, and latency constraints respectively. Then, we formulate the following Lagrange function of $\mathcal{P}(a,\mathbf{x},\mathbf{y},\hat{\mathbf{u}})$:

$$\begin{aligned}\mathcal{L}(a,\mathbf{x},\mathbf{y},\hat{\mathbf{u}},\Theta)=\sum_{n=1}^{N}\sum_{k=1}^{K}a_{n,k}\left(\beta e_{n,k}-\Phi^i(\hat{u_n};m_n)\right)\\+\sum_{k=1}^{K}\lambda_k\left(\sum_{n=1}^{N}a_{n,k}x_{n,k}-X_K\right)+\sum_{k=1}^{K}\nu_k\left(\sum_{n=1}^{N}a_{n,k}y_{n,k}-Y_K\right)\\+\sum_{n=1}^{N}\mu_n\left(\sum_{k=1}^{K}a_{n,k}l_{n,k}-\tau\right)+\sum_{n=1}^{N}\sigma_n(\sum_{k=1}^{K}\ a_{n,k}-1)\end{aligned} \tag{7}$$

We first initialize $\hat{u}_n=u^{min}$. Once the resolution $\hat{u}_n$ is selected, the computation and network resource allocation can be adjusted. Subsequently, the value of $\hat{u}_n$ is updated based on the new resource allocation results.

B. Network Resource Allocation

Based on the KKT conditions and Eq. (7), the optimal network resource allocation can be computed by solving:

$$\begin{aligned}\nabla\mathcal{L}(a,\mathbf{x},\mathbf{y},\hat{u},\Theta)|_{x_{n,k}}=\lambda_k a_{n,k}\\-a_{n,k}\frac{s_n(\beta p_n+\mu_n)\Phi^d(\hat{u_n})}{(x_{n,k})^2\log_2(1+\eta_{n,k})}=0\end{aligned} \tag{8}$$

where the optimal $x^*_{n,k}$ is obtained as:

$$x^*_{n,k}=\begin{cases}\sqrt{s_n(\beta p_n+\mu_n)\frac{\Phi^d(\hat{u_n})}{\log_2(1+\eta_{n,k})}}\times\sqrt{\frac{1}{\lambda_k}} & a_{n,k}=1\\ 0 & \text{otherwise}\end{cases}$$

Since the edge server allocates all its network resources to the users in $\mathcal{N}_k|_{a_{n,k}=1}$, it results in users choosing higher frame resolution u_n in getting more resources. Therefore:

$$x^*_{n,k}=\frac{a_{n,k}\times\sqrt{s_n(\beta p_n+\mu_n)\frac{\Phi^d(\hat{u_n})}{\log_2(1+\eta_{n,k})}}}{\sum\limits_{i\in\mathcal{N}|_{a_{i,k}=1}}\sqrt{s_i(\beta p_i+\mu_i)\frac{\Phi^d(\hat{u_i})}{\log_2(1+\eta_{i,k})}}}\times X_k,\forall k\in\mathcal{K} \tag{9}$$

We notice that $x^*_{n,k}$ is determined by the size of the video segment s_n, frame resolution $\hat{u_n}$, and transmission energy p_n.

C. Computation Resource Allocation

Similar to the analysis in Section IV-B, the optimal computation resource allocation for users can be expressed as:

$$\nabla\mathcal{L}(a,\mathbf{x},\mathbf{y},\hat{u},\Theta)|_{y_{n,k}}=a_{n,k}(\nu_k-\mu_n s_n\frac{\Phi^c(\hat{u_n};m_n)}{(y_{n,k})^2})=0 \tag{10}$$

where

$$y^*_{n,k}=\begin{cases}\sqrt{s_n\mu_n\Phi^c(\hat{u_n};m_n)}\times\sqrt{\frac{1}{\nu_k}} & \text{if}\, a_{n,k}=1\\ 0 & \text{otherwise}\end{cases}$$

Substituting multiplier ν_k leads to:

$$y^*_{n,k}=\frac{a_{n,k}\times\sqrt{s_n\mu_n\Phi^c(\hat{u_n};m_n)}}{\sum\limits_{i\in\mathcal{N}|_{a_{i,k}=1}}\sqrt{s_i\mu_i\Phi^c(\hat{u_i};m_i)}}\times Y_k,\forall k\in\mathcal{K} \tag{11}$$

We notice that user applications with larger DNN models, high frame resolution, and long video segment require more computation resources.

D. Video Frame Resolution

Next, we find the best frame resolution $\hat{u}^*_n$ in terms of minimizing the weighted cost without violating the latency constraint. This problem is defined as:

$$\hat{u}^*_n=\underset{\hat{u}\in[u^{min},u^{max}]}{\text{argmin}}\quad\mathcal{L}(a,x,y,\hat{\mathbf{u}},\Theta)$$

Here the first derivative of $\mathcal{L}$ *w.r.t.* $\hat{u}_n$ is defined by:

$$\begin{aligned}\nabla\mathcal{L}(a,x,y,\hat{\mathbf{u}},\Theta)|_{\hat{u}_n}=a_{n,k}\frac{s_n(\beta p_n+\mu_n)\nabla\Phi^d(\hat{u}_n)}{x_{n,k}\log_2(1+\eta_{n,k})}\\-a_{n,k}\nabla\Phi^i(\hat{u}_n;m_n)+a_{n,k}\frac{\nabla\Phi^c(\hat{u}_n;m_n)}{y_{n,k}}\end{aligned} \tag{12}$$

According to the fitted functions defined in Section III:

$$\begin{aligned}\nabla\Phi^d(\hat{u}_n)=2\alpha^1_d\hat{u}_n\\ \nabla\Phi^i(\hat{u}_n;m_n)=s_n\frac{\alpha^{i_2}_{m_n}}{(\hat{u}_n)^{\alpha^{i_3}_{m_n}+1}}\\ \nabla\Phi^c(\hat{u}_n;m_n)=3\alpha^{c_1}_{m_n}(\hat{u}_n)^2\end{aligned} \tag{13}$$

where $\alpha^{(\cdot)}$ are the fitted parameters. Combined with Eq. (12) and Eq. (13), the derivative function $\nabla\mathcal{L}(a,x,y,\hat{\mathbf{u}},\Theta)|_{\hat{u}_n}$ is monotonically increasing when $\hat{u}_n>0$. The optimal resolution $\hat{u}^*_n$ can thus be found with the following expression:

$$\hat{u}^*_n=\begin{cases}u^{min} & \nabla\mathcal{L}|_{\hat{u}_n}>0\\ \hat{u}_n & \nabla\mathcal{L}|_{\hat{u}_n=\hat{u}^*_n}=0\\ u^{max} & \nabla\mathcal{L}|_{\hat{u}_n}<0\end{cases}$$

The proposed bisection method terminates when $|u^{min}-u^{max}|\leq\epsilon$, where ϵ is a tolerance factor (e.g., 32×32).

V. UA: User-sever Assignment

In this section, we aim to address the other sub-problem UA, i.e., $\mathcal{P}(\mathbf{a},x,y,u)$, defined as:

$$\begin{aligned}\mathcal{P}(\mathbf{a},x,y,u):\underset{a}{\text{argmin}}\sum_{n=1}^{N}\sum_{k=1}^{K}a_{n,k}\left(\beta e_{n,k}-\Phi^i(u_n;m_n)\right)\\ s.t.\,\mathbf{C1},\mathbf{C2},\mathbf{C5}\end{aligned}$$

Here, the original problem **P1** reduces to $\mathcal{P}(\mathbf{a},x,y,u)$, which is a classic $0-1$ integer linear program. To solve $\mathcal{P}(\mathbf{a},x,y,u)$ efficiently, we utilize linear programming relaxation to obtain an initial assignment. Afterward, a strategic game is applied to improve the quality of that initial assignment.

A. Integer Relaxation

We first relax the binary variables $a_{n,k}$ to $\hat{a}_{n,k}\in[0,1]$, where $a_{n,k}$ can be interpreted as the fraction of time that the n-th user utilizes the resources on the k-th edge server. The relaxed optimization of $\mathcal{P}(\hat{\mathbf{a}},x,y,u)$ is defined as:

$$\hat{a}^*_{n,k}=\underset{\hat{a}\in[0,1]}{\text{argmin}}\mathcal{L}(\hat{\mathbf{a}},x,y,u,\Theta)$$

Then, according to the KKT conditions:

$$\hat{a}^*_{n,k}=\begin{cases}0 & \nabla\mathcal{L}|_{\hat{a}_{n,k}}>0\\ 0<\hat{a}_{n,k}<1 & \nabla\mathcal{L}|_{\hat{a}_{n,k}=\hat{a}^*_{n,k}}=0\\ 1 & \nabla\mathcal{L}|_{\hat{a}_{n,k}}<0\end{cases} \tag{14}$$

where $\mathcal{L}|_{\hat{a}_{n,k}}$ is the first derivative of the Lagrange function *w.r.t.* $\hat{a}_{n,k}$, which is express as:

$$\begin{aligned}\nabla\mathcal{L}(\hat{\mathbf{a}},x,y,u,\Theta)|_{\hat{a}_{n,k}}=U_{n,k}+L_{n,k}+\sigma_n\\ U_{n,k}=\beta e_{n,k}-\Phi^i(u_n;m_n)\\ L_{n,k}=\mu_n l_{n,k}\end{aligned} \tag{15}$$

where σ_n is a Lagrange multiplier that is associated to the constraint $\sum_{k=1}^{K}\hat{a}_{n,k}=1$, $U_{n,k}$ and $L_{n,k}$ indicate the weighted cost and the end-to-end latency. We consider $\hat{a}_{n,k}$ as the preference of the n-th user over the k-th edge server, which is jointly determined by $U_{n,k}$ and $L_{n,k}$. Combining Eq. (14)

with Eq. (15), we get:

$$\hat{a}^*_{n,k} = 1 - \frac{U_{n,k} + L_{n,k} + \sigma_n}{\rho} \tag{16}$$

where $\rho \gg 1$ is a positive value that enforces $\hat{a}^*_{n,k}$ to be bounded to 1. According to $\mathcal{P}(a,\mathbf{x},\mathbf{y},\mathbf{u})$, any viable μ_n and σ_n should facilitate all users to finish a video segment within a single time slot τ (i.e., best case scenario) and find an optimal edge server. Once optimized, we convert $\hat{a}^*_{n,k}$ to $a^*_{n,k}$ as:

$$a^*_{n,k} = \begin{cases} 1 & k = \text{argmax}\, \hat{a}_{n,k}, \forall k \in \mathcal{K} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

B. Initial UA: Integer Relaxation Algorithm

Next, we update μ_n and σ_n. We define the parameter update iteration $i \le I_{max}$ and the difference between the latency of the current iteration and the length of a time slot as:

$$\kappa^i_n = \sum_{k=1}^{K} s_n \Big(\frac{\Phi^d(u^t_n)}{x^t_{n,k} \log_2(1+\eta_{n,k})} + \frac{\Phi^c(u^t_n; m_n)}{y^t_{n,k}} \Big) - \tau$$

where $\kappa^i_n < 0$ indicates that the video segment from n-th user can be processed before the end of a time slot; otherwise, it violates constraint **C5**. With κ^i_n, the update rule of μ_n is defined as follows:

$$\mu^{i+1}_n = \Big[\mu^i_n + \Delta(t) \kappa^i_n \Big]^+ \tag{18}$$

As shown in Eq. (18), the diminishing step sizes are denoted by $\Delta(t)$ (e.g., $\{1/t\}^{\frac{1}{2}}$). Similarly, σ_n is updated by:

$$\sigma^{i+1}_n = \Big[\sigma^i_n + \Delta(t) (\sum_{k=1}^{K} a_{n,k} - 1) \Big]^+ \tag{19}$$

C. Enhanced UA: A Strategic Game

If the relaxed solution of problem $\mathcal{P}(\hat{\mathbf{a}},x,y,u)$ happens to have all variables with values of either 0 or 1, then the solution is optimal. However, in most cases, many of the relaxed variables are fractional values (between 0 and 1). In such cases, the rounded result does not guarantee the optimal solution. When this is true, the relaxed solution gives an upper bound of the original problem, that is, $\mathcal{P}(\hat{\mathbf{a}}^*,x,y,u) \le \mathcal{P}(\mathbf{a}^*,x,y,u)$. Generally, methods such as Branch and Bound (B&B) and Cutting-plane can be applied to effectively solve such discrete and combinatorial optimization problems when facing fractional values. However, such algorithms need to solve the entire sub-problem $\mathcal{P}(\hat{\mathbf{a}},x,y,u)$ repeatedly to evaluate candidate solutions. This may result in huge computational complexity when the edge environment contains a large number of user devices and edge servers.

Thus for *EdgeURB* framework, we propose an Enhanced UA where user devices are initially connected to the edge server according to the rounded solution given in Eq. (17). Afterward, we formulate a strategic game to further improve the overall performance. Strategic game is a powerful tool to transform centralized optimization problems (i.e., usually NP-hard) into decentralized problems [13]. It provides a sub-optimal solution with less computational overhead and guarantees that all players are mutually satisfied, which is key for real-world edge system implementation. Here, we define game iteration t and formulate the game as follows:

Players: For the strategic game, we define $\mathcal{N}$ as the set of players, i.e., user end-devices. It is assumed that players are inherently selfish such that they are only concerned about increasing their own utilities. It is a realistic assumption for real-world mission-critical use-cases (e.g., tactical scenarios, disaster response) where applications are managed by individual stakeholders (e.g., teams, agencies).

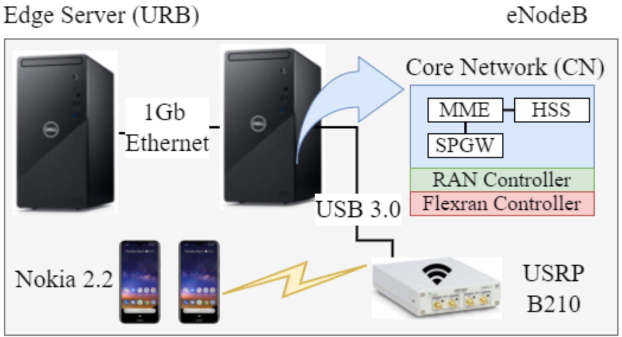

Fig. 3: *EdgeURB* framework hardware testbed prototype

Strategy: A player strategy $a_n = \{1,...,K\}, \forall n \in \mathcal{N}$ indicates the selection of an edge server, with *EdgeURB* global strategy denoted by $\mathcal{A}(t) = \{a_1(t),..,a_N(t)\}$. We define $\mathbf{a}_{-n}(t)$ as the set of strategies made by all other players except for player n.

Utility: The utility of the players represents the negative weighted cost. Thus, the utility function for player n can be formulated as:

$$\phi_n(a_n(t),\mathbf{a}_{-n}(t)) = -U_{n,k}|_{k=a_n(t)} \tag{20}$$

The Enhanced UA problem is formulated as a strategic game with individual utilities calculated by Eq. (20). Given $\mathbf{a}_{-n}(t)$, each player chooses a best strategy $a_n(t)$ to maximize its own utility (in a selfish manner) where $\forall n \in \mathcal{N}$,

$$a^*_{n,k} = \begin{cases} 1 & k = \underset{a_n(t) \in \{1,2,...,K\}}{\text{argmax}} \phi_n(a_n(t),\mathbf{a}_{-n}(t)) \\ 0 & \text{otherwise} \end{cases} \tag{\textbf{P3}}$$

It is to be noted that (**P3**) can be treated as $N \times K$ parallelizable resource allocation problems. Thereby the computational complexity is reduced to polynomial time. In order to solve **P3**, *EdgeURB* assigns a 'Helper' process to each user device (randomly or from the nearest edge node). The helpers apply the Best Response Strategy algorithm in response to **P3**. This is done by iterating all the edge servers under a given profile $\mathbf{a}_{-n}$ and finding the best update strategy $a_n(t-1) \rightarrow a_n(t)$ for utility maximization.

Nash equilibrium: Since any unilateral strategy update may affect the preferences of other players and may lead to continuous updating of individual strategies, *EdgeURB* aims to find a Nash equilibrium (NE) for the proposed game:

Definition 1: A strategy profile $\mathcal{A}^* = \{a^*_1, a^*_2, ..., a^*_N\}$ is a NE of a strategic game, if at the equilibrium $\mathcal{S}^*$, no player can further increase its utility by unilaterally altering its strategy, i.e.,

$$u_n(a^*_n, \mathbf{a}^*_{-n}) \ge u_n(a'_n, \mathbf{a}^*_{-n}), \forall a'_n \in \mathcal{A}, n \in \mathcal{N}$$

In accordance to ***Definition*** (1), the URB broadcasts the current global strategy profile $\mathcal{A}(t-1)$ to all the 'Helpers'. The 'Helpers' solve **P3** and propose the best update request $a_n(t-1) \rightarrow a_n(t)$ to the *EdgeURB*. At each iteration t, the users in need submit their update requests to the URB sequentially. Upon each submission, the updated global strategy profile $\mathcal{A}(t)$ is again forwarded to the 'Helpers' when the URB waits for the next update request. When there is no new update request, the process terminates and returns the final outcomes of UA and RA.

VI. Performance Evaluation

In this section, we present a testbed based implementation and experimental results, followed by extensive simulation results. The hardware limitations of the testbed allows the system to be evaluated for only a small number of edge servers and user devices. Thus, via simulation we evaluate *EdgeURB*'s performance against a larger set of system components.

A. Testbed Design, Implementation, and Experiment Results

In order to evaluate *EdgeURB* performance on 'near real-world' edge systems, we use the hardware prototype described in Section III-B, as shown in Fig. 3. In the testbed, the URB and the edge server are placed on the same physical machine, while other framework and eNodeB components (e.g., core network, ran controller, and flexran controller) are installed on another. Since server/service discovery [14] is one of the key issues in the implementation of edge systems, we consider access point name (APN) as an edge server identifier and its associated access point (either LTE eNodeB or Wifi AP). We assume that the URB contains a list of APNs that are available in the system and the user devices are initially connected to the URB (i.e., the APN of the URB is known in advance).

Application Profiling: Since most well-known video analytics models are publicly available, we assume that the pre-trained models are pre-cached in edge servers and their accuracy and computation latency models are known, as in Fig. 1. Once user devices are connected to the URB, the URB will respond by sending the existing APN list. Based on received APN list, the user devices start to establish a short connection period with each edge server in the list to obtain network quality (i.e., $\log_2(1+\eta_{n,k})$ in Eq. 2). Afterwards, the URB starts to collect the user information including signal strengths, energy consumption (e.g., voltage and current), application requirement (i.e., the selected model and the length of the video segment) and user identification (i.e., International Mobile Subscriber Identity). Upon completion, the URB performs the optimization and delivers the resource allocation results to each edge server. In the meantime, the APN of the edge server that is assigned to the user will be sent to the user device as the final user-server assignment.

Resource Mapping: For resource allocation, we need to convert the virtual resources (i.e., $\mathbf{x}$ and $\mathbf{y}$) to the actual radio and computational resources. In the *EdgeURB* prototype LTE setup, the value of $\mathbf{x}$ indicates the amounts of allocated bandwidth. Therefore, we map $\mathbf{x}$ to the number of allocated sub-channels. Given that the sub-channel bandwidth is 180 KHz, the number of allocated sub-channels is computed by $\lfloor \mathbf{x}/180 \rfloor$. As for computational resources, we consider the GPU-enabled (NVIDIA GeForce RTX 2060) edge system for real-time video processing. *EdgeURB* computational resource allocation scheme follows [15], where the GPU is shared by multiple video streams in a time-division manner i.e., the system grants the exclusive GPU usage to different analytics models during consecutive time periods. This method avoids the model interference and reload overhead caused by multiple video streams using a single GPU concurrently. In order to preserve the real-time processing characteristics, the GPU usage time frame T_g is set to a very small value (i.e., 300 ms). Here, we measure the computational complexity mentioned in Section III-C as the pure GPU inference time and convert $\mathbf{y}$ to the length of GPU usage time by the following mapping function:

$$y_{n,k} \rightarrow \frac{y_{n,k}}{\sum_{i=1}^{N} y_{i,k}} \times T_g, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}$$

Testbed Experiment Results: The results of latency satisfaction (**C5** in **P1**), resource allocation, accuracy and energy consumption are shown in Fig. 4 with two bandwidth settings: $X = 3.6$ MHz and $X = 7.2$ MHz (i.e., 20 and 40 LTE sub-channels). In this experiment, we let User 1 select model YOLOv5x and require 3 fps (computationally intensive), while User 2 chooses model YOLOv5s and needs 6 fps (bandwidth-hungry). Here, we choose $\beta = 0.1$ as the balance factor (more discussions on β are given in Section VI-B). The time slot is configured as $\tau = 1$ s. Fig. 4(a) and Fig. 4(b) show the network (NT) and computation (CT) latency. We notice that the network latency dominates the overall latency in both cases. The figure also demonstrates that all the video segments from the two users can be processed within the pre-defined time slot $\tau = 1$ (the dotted line).

We also observe that when there are insufficient bandwidth as shown in Fig. 4(d), User 2 dominates the network resource while the User 1 dominates the computation resource. This is due to differences in the computational complexity and fps requirements of their analytics models. However, with sufficient bandwidth condition (i.e., 7.2 MHz), the overall resource allocation shows a proportional distribution. As shown in Fig. 4(e), User 1 occupies 40% of the bandwidth and GPU usage time and User 2 takes the remaining 60%. Such distribution is determined by the trade-off between accuracy and energy consumption. The results of accuracy and energy consumption are shown in Fig. 4(c) and Fig. 4(f). When the bandwidth increases from 3.6 MHz to 7.2 MHz, the accuracy increases while the energy consumption decreases. The above experiment results demonstrate the success of *EdgeURB* in handling video analytics with heterogeneous requirements under varying edge resource availability.

B. Simulation Results

Next, we show simulation results to validate *EdgeURB* schedulability and scalability when facing a large number of heterogeneous applications and requirements due to the testbed prototype's hardware limitations. In the simulation, the users use YOLOv5 algorithm with different analytics models and fps requirements (e.g., the length of a time slot τ is set to 1s and the fps varies from 5 to 15). We allow applications to choose a resolution from 128×128 to 640×640 which is a multiple of 32×32. The user device energy consumption parameters are obtained from testbed experiments as they are representative of common smartphone settings. The signal strength of the wireless channels is measured in dB. The available bandwidth for each edge server is randomly selected between {5MHz, 10MHz, 15MHz, 20MHz}. Such SNR and bandwidth values together model heterogeneous wireless resources. On the other hand, the GPU speed of edge servers is accelerated upto 10X based on the computing capacity of NVIDIA GeForce RTX 2060. We randomly create 50 test cases for each simulation. Through such massive simulations, we demonstrate the validity and efficiency of *EdgeURB* in handling different real-world scenarios.

Baseline algorithms: For fair comparison against *EdgeURB*, we choose the following set of traditional de-factor edge resource management algorithms.

1) $maxSNR$: User devices are connected to the edge server which gives the maximum signal strength.
2) $randLoad$: User devices are evenly assigned to edge servers, i.e., each server hosts at most $\frac{N}{K}$ applications.
3) $fairAlloc$: Under fair allocation strategy, both the network and computational resources are evenly allocated to the connected users.

The impact of β: We first study the impact of the balance factor β in handling the trade-off between energy consumption and accuracy (normalized to $0-1$). The results are shown in Fig. 5 with 16 users and 4 edge servers. Both the energy consumption and accuracy converge to baseline performance (i.e., $maxSNR$ with fixed resolution) for larger $\beta \geq 30$, where the energy consumption for transmitting each frame is 1.07×10^{-2} J and the accuracy is 0.25. When $\beta = 0$ (i.e., indicating that the users are not concerned about the energy consumption at all), the maximum accuracy is bounded to the computation

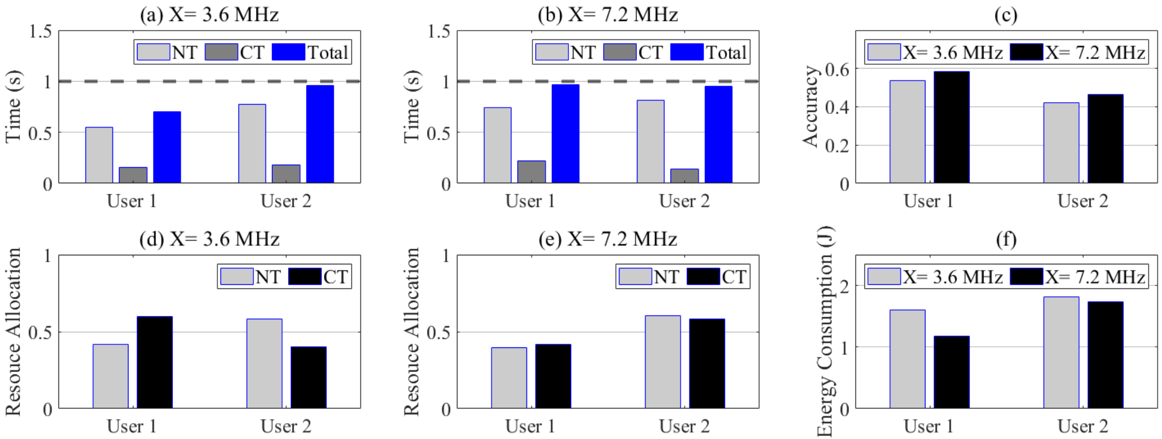

Fig. 4: Testbed experiment results on latency satisfaction, resource allocation, accuracy and energy consumption.

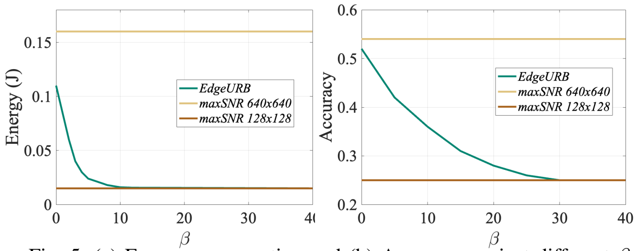

Fig. 5: (a) Energy consumption and (b) Accuracy against different β

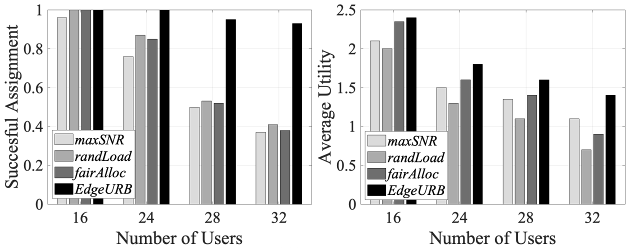

Fig. 7: Comparison among different user-server association strategies

capacity. In Fig. 5, when the video frame resolution is fixed at 640×640, the maximum accuracy is 0.54. However, the actual accuracy is limited to 0.52 due to the processing constraint. That is, if the users continue to increase the resolution, edge servers start to fail in completing the processing of all video segments in time. As shown in Fig. 6, for the values in between these two extremes $\beta=0$ and $\beta=40$, e.g., at $\beta=8$, the proposed *EdgeURB* is able to improve the accuracy by 36% at the cost of only 6% increase in energy consumption. Consequently in the following simulations and experiments, we choose $\beta=8$ as the balance factor to achieve balanced accuracy and energy comparison.

Schedulability vs. scalability: It is critical that *EdgeURB* framework is able to handle practical real-world applications with heterogeneous user requests (e.g., different fps and analytics models). We demonstrate the importance of user-server assignment (UA) algorithm towards the overall *EdgeURB* performance and scheduling of a large number of users. For the evaluation, we introduce a new performance indicator, viz., the probability that *EdgeURB* can successfully schedule user-server assignment. A successful assignment signifies that all video segments can be processed in time at least with a minimum resolution of 128×128. The results are shown in Fig. 7 with 4 edge servers and $\beta=8$.

We observe that as the number of users increases (i.e., $N=16$ to $N=32$), the competition for resources among users becomes increasingly fierce as expected. For $randLoad$ algorithm where edge servers accept equal amount of users, schedulability is ensured when there are a fewer users ($N\leq16$). However, with the arrival of more users, the schedulability starts to decline significantly. In addition, $randLoad$ produces a low average utility since it ignores the impact of the signal strength $\eta_{n,k}$ to the energy efficiency. In comparison to $randLoad$, our proposed *EdgeURB* solution improves the average utility by 42% while increasing the average schedulability by 27.5% and upto 56% based on test cases.

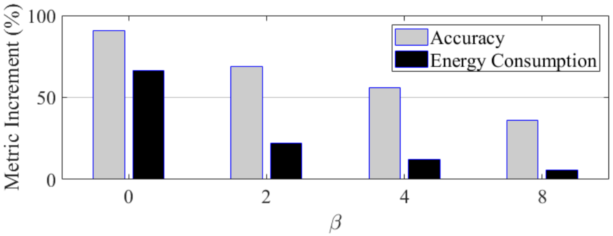

Fig. 6: The accuracy and energy comparison against different β.

Compared to $randLoad$, the users under $maxSNR$ algorithm are always connected to the edge server that provides the maximum signal strength $i.e.,k=\text{argmax}\eta_{n,k}$. The $maxSNR$ algorithm guarantees the most effective bandwidth utilization, however edge servers with good signals nevertheless face offloading and computation congestion problems caused by too many users selecting the same edge server. Especially for limited edge resources and a large number of users ($N\geq24$), the user-server assignment of $maxSNR$ is even worse than $randLoad$. Unlike $maxSNR$, our proposed *EdgeURB* assigns users to the edge server that are most sought after while avoiding congestion problems. This is achieved by bringing the heterogeneous user requirements and the resources availability of edge server into the optimization of user-server assignment in problem **P1**.

The same figure also compares *EdgeURB*'s performance against a fair resource allocation strategy, viz., the $fairAlloc$. We observe that when there are enough edge resources for small number of users to use, the performance of $fairAlloc$ and *EdgeURB* are very close. However, $fairAlloc$ becomes less effective when facing more users and its schedulability starts to drop greatly once $N\geq24$. Compared to $fairAlloc$, the proposed *EdgeURB* adapts the frame resolution and resource allocation according to the changes of the edge resource availability. In addition, *EdgeURB* jointly considers the computational complexity of the user application and achievable data rate along with the energy consumption characteristics of a user device (i.e., Eq. (9) and Eq. (11)). This yields a more efficient resource sharing strategy.

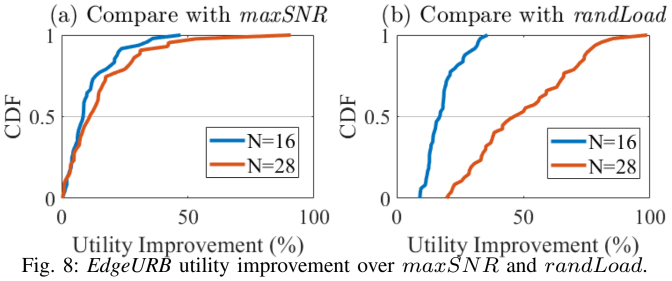

Fig. 8: *EdgeURB* utility improvement over $maxSNR$ and $randLoad$.

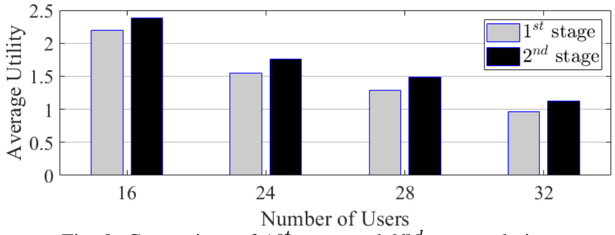

Fig. 9: Comparison of 1^{st} stage and 2^{nd} stage solutions

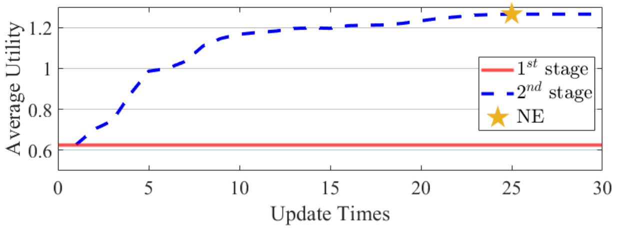

Fig. 10: An exemplary of strategy update and NE point

Probability of improvement: From the results we observe that the utility between different test cases fluctuates greatly (with some values in the negative region) as calculating the improvement through average utility of all the test cases does not reveal the actual difference. Therefore, in Fig. 8, we demonstrate the cumulative distribution function (CDF) of such improvements that allows a more detailed comparison between *EdgeURB* and the other competing algorithms. It is to be noted that we do not show the case with $N = 32$ because of its low schedulability that makes its utility value meaningless, in other words it violates the processing constraint. We notice that when $N = 16$, the probability that the average utility has increased by more than 10% are 0.92 and 0.38 in compared to algorithms $randLoad$ and $maxSNR$ respectively. Evidently such probabilities will increase when there are more users. In Fig. 8, we see that at $N = 28$, *EdgeURB* is able to increase the average utility by more than 20% for almost all the test cases in comparison to $randLoad$ while that value is 0.25 when compared to $maxSNR$.

Evaluation and convergence of 2-stage algorithm: Fig. 9 shows the comparison between the solutions of the 1^{st} and 2^{nd} stages in terms of average utility. The average improvement against different number of users is around 9%. We also show an example of strategy update where the system has 4 edge servers and 32 users. The results of convergence are shown in Fig. 10. We notice that the 2^{nd} stage converges after the 25-th update. This convergence is guaranteed by the finite improvement property (FIP) of the proposed game [16], [17]. Therefore, *EdgeURB*'s 2^{nd} stage algorithm, using the best response strategy to solve **P3**, should result a NE within finite updates. Under constrained resources, we can conclude that naive strategies such as $fairAlloc$, $randLoad$, and $maxSNR$ (from Figs. 7, 8) fail to effectively address applications which have strict processing constraints. Inferring from all the results, we can state that *EdgeURB* exhibits highly adaptive and efficient user-server assignment when facing heterogeneous user requests and edge resources.

VII. Conclusions

In this paper, we presented *EdgeURB*, a framework for real-time video analytics at edge. We showed that unlike existing solutions, *EdgeURB* jointly optimizes edge resource management and video configuration adaptation towards satisfying the real-time latency requirements of video analytics applications. We demonstrated how the proposed framework can find the trade-off between device energy consumption and video analytics accuracy under given latency constraints using a two-stage algorithm. We implemented *EdgeURB* on a hardware testbed prototype to validate the framework's utility towards latency satisfaction and resolution adaptation under varying edge resources. We also performed extensive simulations to demonstrate *EdgeURB*'s improvements towards schedulability and scalability over other baseline strategies.

References

[1] X. Zhang, A. Pal, and S. Debroy, "Effect: Energy-efficient fog computing framework for real-time video processing," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2021, pp. 493–503.

[2] X. Zhang, M. Li, A. Hilton, A. Pal, S. Dey, and S. Debroy, "End-to-end latency optimization of multi-view 3d reconstruction for disaster response," in *2022 10th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 2022, pp. 17–24.

[3] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[4] J. Chen, K. Li, Q. Deng, K. Li, and S. Y. Philip, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Transactions on Industrial Informatics*, 2019.

[5] Q. Liu, T. Han, and E. Moges, "Edgeslice: Slicing wireless edge computing network with decentralized deep reinforcement learning," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 234–244.

[6] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1421–1429.

[7] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 756–764.

[8] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 257–266.

[9] X. Zhang and S. Debroy, "Migration-driven resilient disaster response edge-cloud deployments," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, 2019, pp. 1–8.

[10] Y. Chen, B. Ai, Y. Niu, Z. Zhong, and Z. Han, "Energy efficient resource allocation and computation offloading in millimeter-wave based fog radio access networks," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[11] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-mec server systems," in *2018 IEEE international conference on communications (ICC)*. IEEE, 2018, pp. 1–6.

[12] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE access*, vol. 4, pp. 5896–5907, 2016.

[13] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19 324–19 337, 2018.

[14] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, "Selective offloading in mobile edge computing for the green internet of things," *IEEE Network*, vol. 32, no. 1, pp. 54–60, 2018.

[15] W.-J. Kim and C.-H. Youn, "Lightweight online profiling-based configuration adaptation for video analytics system in edge computing," *IEEE Access*, vol. 8, pp. 116 881–116 899, 2020.

[16] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[17] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.