ISSN 2277-2685
IJESR/April-June. 2025/ Vol-15/Issue-2s/990-998
Akkaloori Yamini *et. al.*, / International Journal of Engineering & Science Research

Optimized VR Video Offloading via Deep Reinforcement Learning

AVS Radhika[1], Akkaloori Yamini[2], Dodla Sathwika[3]
[1]Associate professor, Department of CSE, Bhoj Reddy Engineering College for Women, India.
[2,3]B.Tech Student, Department of CSE, Bhoj Reddy Engineering College for Women, India.

ABSTRACT

In recent years, eXtended Reality (VR) applications have been widely used across diverse sectors such as tourism, healthcare, education, and manufacturing. Such apps are now available on mobile devices, wearable devices, tablets, and similar platforms. Mobile devices often have limitations regarding battery capacity and computing power, which restricts the variety of supported apps and diminishes the user experience. An viable solution to these challenges is to transfer the computation to cloud servers. The fundamental restriction of cloud computing is the considerable distance between the processing server and the end user, which might lead to unacceptable latency for several mobile XR applications. To address these limitations, Multi-access Edge Computing (MEC) is proposed to deliver mobile computing, network control, and storage services to the network peripheries (such as base stations and access points) to enable the deployment of computation-intensive and latency-sensitive applications on resource-constrained mobile devices. This study presents a Deep Reinforcement Learning-based offloading strategy for XR applications (DRLXR). The issue is articulated as an optimization equation for a utility function that considers both energy consumption and execution latency at devices, using the Markov Decision Process (MDP) paradigm for decision-making. The Deep Reinforcement Learning (DRL) approach is then used to train and ascertain the near-optimal offloading option for mobile XR gadgets. The proposed DRLXR system is evaluated in a simulated environment and compared with other innovative offloading techniques. The simulation findings demonstrate that our suggested approach surpasses its alternatives for overall execution delay and energy usage.

Keywords— eXtended Reality, Offloading, Multi-access Edge Computing, Deep Reinforcement Learning, Energy efficiency,
Quality of Service

INTRODUCTION

eXtended Reality (XR) apps use the newest advancements in 5G and subsequent network connectivity. XR is described as the integration of virtual three-dimensional objects with real-world content, experienced via smart devices such as portable smartphones or head-mounted displays. XR is classified as Augmented Reality (AR), Mixed Reality (MR), or Virtual Reality (VR) based on the equilibrium between virtual content and reality. Nonetheless, irrespective of the categorization, there is a significant exponential growth in XR applications across several domains, including healthcare, tourism, education, and manufacturing.

Figure 1 depicts a general XR system, with the following fundamental components:

• Input sensors that gather data using different types of integrated or auxiliary sensors, including gyroscopes, location sensors, cameras, etc. Bao Trinh Nguyen and

Gabriel-Miro Muntean are affiliated with the Insight SFI Research Centre for Data Analytics and the Performance Engineering Lab at the School of Electronic Engineering,

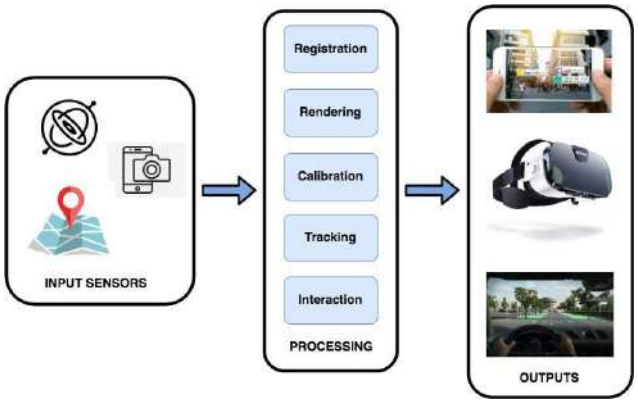

Figure 1: Components of an XR system [4]

Processing modules are tasked with handling the acquired data, executed either locally or by offloading to a cloud server, fog server, or edge server, depending upon the necessary computational complexity and available processing capacity.

Outputs pertain to post-processing operations related to the display of XR material, including the streaming of high-definition video, the activation of actuators, and interaction with external devices. This phase employs head-mounted displays (HMDs) [6], portable displays [7], and/or devices such as haptic gloves and olfactory dispensers [8], among others.

Notwithstanding the recent rapid advancements in hardware design and development, mobile devices used for XR applications remain resource-constrained relative to desktops or servers. The trade-off for enhanced mobility and compactness is a decrease in battery capacity and processing power. Conversely, the intricate algorithms mostly used in video content processing need substantial computer resources for XR applications. A proficient method to address the difficulty of facilitating immersive XR applications on resource-constrained mobile devices is to offload computations to resource-abundant devices, such as cloud or edge servers, over the network. Cloud computing has emerged as a successful computer paradigm. The fundamental concept is the centralization of computation, storage, and network management inside the cloud, facilitated by data centers, backbone networks, and cellular core networks [9], [10]. To perform computations in the cloud, mobile devices and servers must use offloading frameworks, such as MAUI [11] or ThinkAir [12]. Recently, the functionality of cloud computing is progressively shifting towards the network edges, nearer to consumer devices [13]. By using the dormant computational power and storage capacity located at network edges, enough resources are provided for XR apps to execute computation-intensive and latency-sensitive operations on user mobile devices. This notion underlies the Multi-Access Edge Computing (MEC) paradigm, whereby mobile devices may interact with and receive assistance from MEC servers using several wireless communication protocols, including LTE, 5G, WiFi, or their combinations. Figure 2 depicts the overarching architecture of a MEC system.

In a MEC-enhanced cloud computing environment, the problem persists in determining which XR processing activities should be offloaded and to which location, in order to optimally balance XR application demands while efficiently using device, MEC, and cloud resources.

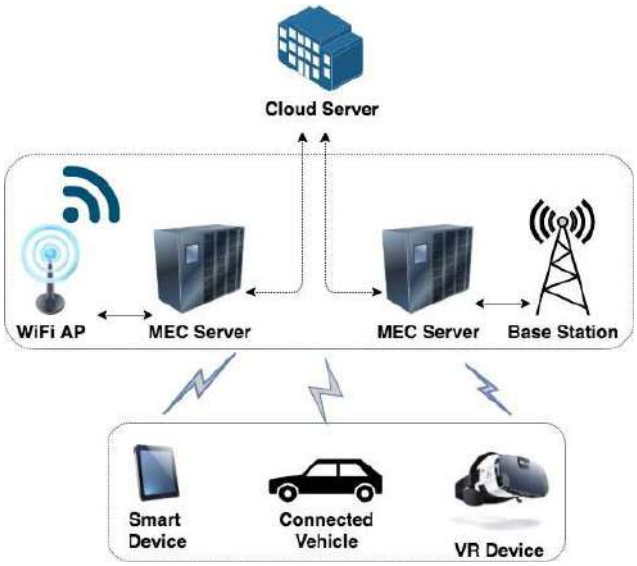

Figure 2: General architecture of a MEC-enhanced cloud computing system

Conversely, computational, storage, and network resources. This is not straightforward, and several solutions have been offered using heuristic or complicated optimization methods [16].

This study presents a Deep Reinforcement Learning-based offloading method for XR applications (DRLXR) that allocates compute across the device, MEC, and cloud to optimize XR application performance and energy efficiency within specified network resource restrictions. This publication presents the following contributions:

• A tri-layer architecture for XR systems is proposed, emphasizing the energy-efficient computation offloading to minimize overall power consumption while adhering to the stringent delay requirements of XR applications. • The issue is articulated through the Markov Decision Process (MDP) framework, with near-optimal offloading decision-making derived using a Deep Reinforcement Learning (DRL) technique. The XR applications are segmented into discrete tasks and modeled using Graph Theory. The suggested DRLXR solution is assessed utilizing the Network Simulator NS-3 and the Open Gym AI library, and it is compared with other innovative offloading methodologies.

The remainder of this paper is structured as follows: Section II examines several innovative offloading techniques identified in the academic literature. Section discusses the technological underpinnings of Deep Reinforcement Learning (DRL).

LITERATURE REVIEW

This section examines advanced offloading strategies suggested in the research literature. Four primary categories of offloading methods are identified depending on their kind of offloading: i) binary offloading, ii) partial offloading, iii) stochastic model-based offloading, and iv) deep learning-based offloading.

.

Reinforcement Learning-based Offloading

Due to the scarcity of training data and the continuous emergence of innovative applications, supervised learning poses challenges for feature extraction. While unsupervised learning has potential for using network traffic attributes, achieving real-time processing remains a challenge [28]. Conversely, the reinforcement learning paradigm may be used without the need of a pre-existing dataset for training. Training

may occur via direct contact with the learning agent and its environment.

[29], [30], and [31] suggested using reinforcement learning and/or integrating it with deep learning to develop varied offloading strategies for MEC-enhanced Internet of Vehicle (IoV) systems. In [29], Li et al. developed an online reinforcement learning approach using feedback and traffic patterns to equilibrate traffic loads. To achieve effective traffic management, a combination challenge of communication, caching, and computation was examined in [30]. [31] suggested an offloading strategy that addressed the trade-off between energy usage and latency in the IoV system. The reinforcement learning-based method was then used to design an offloading technique for Internet of Vehicles nodes.

The authors of [32] examined IoT nodes powered by energy harvesting. The suggested approach enabled IoT devices to pick the edge server and offloading rate according on the current battery level and previously recorded radio transmission rate. Deep Reinforcement Learning was used to enhance offloading performance inside a very intricate state space.

Wang et al. [33] reformulated the original combined computation offloading and content caching problem into a convex optimization problem and then resolved it in a distributed and efficient manner. Hao et al. [34] examined the offloading issue, including the constraints of CPU and storage capacity of mobile devices while minimizing long-term delay. The proposed method was developed using DRL, and the suggested approach demonstrated significant improvements in convergence time and latency reduction.

Wang et al. [35] presented a Meta Reinforcement Learning-based technique (MRLCO) to optimize offloading decisions for User Equipment (UE). Mobile apps are structured as Directed Acyclic Graphs (DAG). The author use Meta Reinforcement Learning (MRL) to identify near-optimal offloading options for User Equipments (UEs) to minimize latency. UE applications are segmented into many sub-tasks. Each sub-task is thereafter determined to be executed locally or delegated to a virtual machine at the MEC server. MRLCO surpasses the other baseline algorithms for average latency. The primary drawback of MRLCO is the absence of considerations for UE mobility and energy usage.

Despite exploring several routes, the majority of current works have failed to adopt a comprehensive strategy that addresses the intricacies of contemporary applications, particularly those related to XR. These applications consist of several little jobs, and their performance is collectively affected by network conditions and energy usage. This article addresses this gap.

TECHNICAL BACKGROUND

This section succinctly outlines the history pertaining to Markov Decision Processes (MDP) and Deep Reinforcement Learning (DRL), methodologies used in the proposed solution.

Deep Reinforcement Learning

A hybrid system that integrates the benefits of value function and policy search techniques, known as Actor-Critic [38], was presented. The Actor-Critic approach integrates a value function with a direct representation of the policy, leading to the development of actor-critic methods, as seen in Figure 3. The actor (policy) acquires knowledge via feedback from the critic (value function). Actor-Critic approaches use the value function as a reference for policy gradients, distinguishing them from other baseline methods by their incorporation of a learned value function. Actor-Critic approaches have many advantages: i) they need little computational resources for action selection compared to other methods; ii) they are capable of

learning an explicitly stochastic policy or optimum probability for action selection. The Actor-Critic technique is used as a decision-making framework for job offloading in XR devices due to its benefits. This will be further upon in the subsequent section.

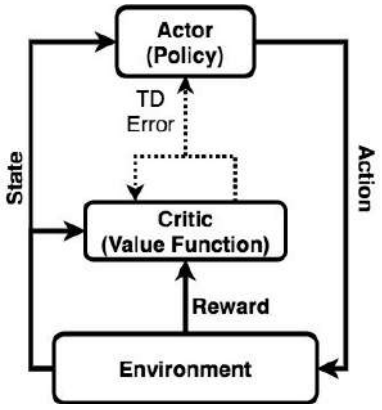

Figure 3: The Actor-Critic paradigm [38]. The actor (policy) selects an action based on the state received from the environment. Simultaneously, the critic (value function) obtains the state and reward from the previous encounter. The critic uses the estimated TD mistake to update both itself and the actor.

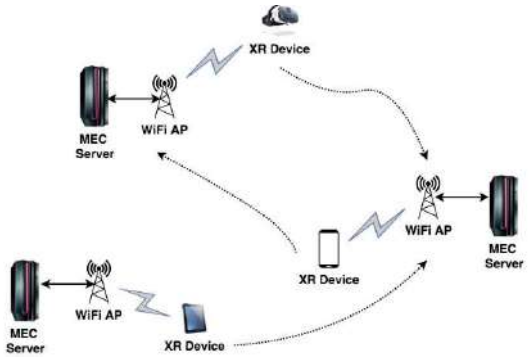

Figure 4: Testing Topology

PROBLEM STATEMENT

This section addresses the planned offloading mechanism. The system architecture is first delineated, followed by an exposition of the issue formulation grounded on Markov Decision Processes (MDP). The DLR-based offloading mechanism is thoroughly introduced. All acronyms used in this study are shown in Table I.

To assess and compare our suggested methods with other algorithms, we use the following metrics:

• Mean energy usage (in Joules) across all devices

• Mean total duration for job completion

A. System Architecture

The overarching architecture of the MEC-enhanced network system has three tiers: core network, edge network, and XR devices, as seen in Figure 5.

The Operations Support System (OSS) and Multi-Access Edge Orchestration (MEO) are positioned at the apex of the core network hierarchy. The OSS block is tasked with receiving client requests, assessing their approval, and forwarding them to MEO. The MEO oversees the MEC-based system, being aware of the available resources, services, and installed MEC hosts, while also monitoring the topology. MEO also identifies optimal hosts for application deployment, taking into account resource availability, service

accessibility, and restrictions like as latency. The primary components at the Edge Network level are the MEC Server and the MEC Platform. The latter oversees the life cycle management of both applications and MEC platforms, notifying the MEO of any pertinent events. The MEC platform manager facilitates platform configuration and application lifecycle management operations. Ultimately, at the base are XR devices executing computation-intensive applications, such as deep learning-based object identification and 360° video streaming, which need the offloading of some activities to MEC servers.

The following discussion pertains to the block diagram of the MEC server and XR devices,.

• On mobile XR devices, the Application Monitor block oversees the simultaneous operation of all apps. An energy consumption monitor block indicates the residual battery level and the pace of depletion. The Channel State Information (CSI) module continuously monitors the signal strength, quantified by the Received Signal Strength Indicator (RSSI). All three blocks provide information to the Local Trainer for data collection and to the Deep RL-based choice Maker for determining the offloading choice. The calculation is either directed to the Offloading Scheduler module and then sent over the Radio Transmission Unit to the MEC server, or it is performed locally at the Local Executor block.

At the MEC server, the Data Aggregation component gathers all requests from devices connected to the Radio Transmission Unit in the neighborhood and then transmits them to the Traffic Management block. The Traffic Management block oversees all Virtual Machines (VM) and allocated resources for the respective demands of mobile devices. All queries are then processed, and the responses are sent back to XR devices using the Remote Execution Service block. The MEC server is also connected to Remote Cloud servers; however, this article disregards the impact of such exchanges.

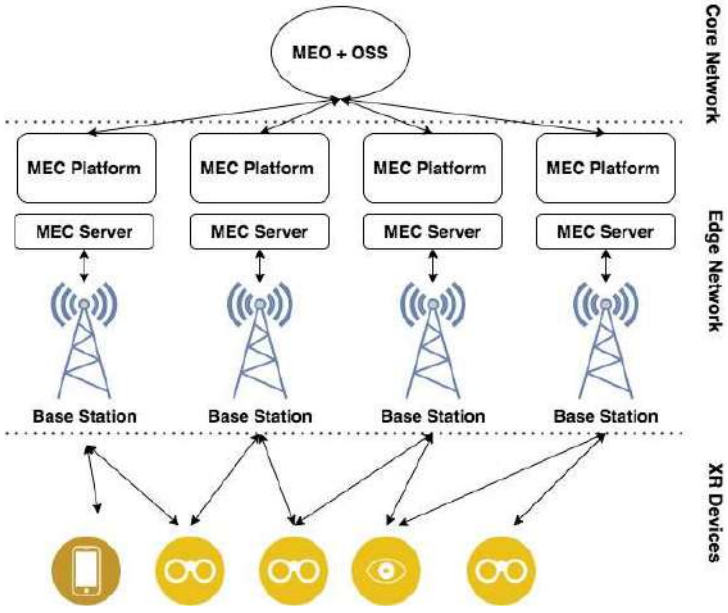

Figure 5: System Architecture of a MEC-based Network System

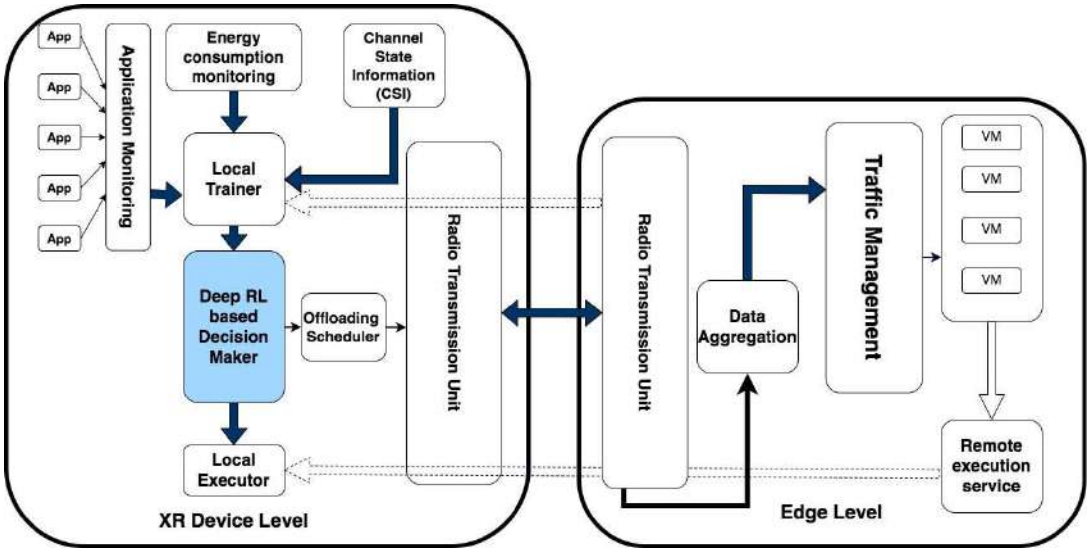

Figure 6: Block diagram of the proposed solution

CONCLUSIONS AND FUTURE WORKS

This study presents and formulates the Deep Reinforcement Learning-based Offloading method for XR devices (DRLXR) inside a MEC-enabled network context. A three-tier hierarchical network design is proposed. The workload offloading issue at the XR device is articulated by Deep Reinforcement Learning (DRL). The XR devices use an Actor-Critic technique for training and decision-making about task offloading, informed by observed data on radio signal quality, energy consumption, and the state of running applications. The suggested DRLXR system is assessed in a simulated setting and juxtaposed with other offloading techniques. The simulation findings demonstrate that DRLXR surpasses other options for average energy usage and overall completion time. Future endeavors will concentrate on a unified system that integrates the proposed offloading strategy with resource management at the MEC server, accommodating diverse QoS needs.

REFERENCES

[1] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: stateof-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[2] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62, 2010.

[3] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *2012 Proceedings IEEE Infocom*, pages 945–953. IEEE, 2012.

[4] Mung Chiang and Tao Zhang. Fog and iot: An overview of research opportunities. *IEEE Internet of Things Journal*, 3(6):854–864, 2016.

[5] Andrei OJ Kwok and Sharon GM Koh. Covid-19 and extended reality (xr). *Current Issues in Tourism*, pages 1–6, 2020.

[6] Dimitris Chatzopoulos, Carlos Bermejo, Zhanpeng Huang, and Pan Hui. Mobile augmented reality survey: From where we are to where we go. *Ieee Access*, 5:6917–6950, 2017.

[7] Abid Yaqoob, Ting Bi, and Gabriel-Miro Muntean. A survey on adaptive 360° video streaming: Solutions, challenges and opportunities. *IEEE Communications Surveys & Tutorials*, 22(4):2801–2838, 2020.

[8] Fabio Silva, Mohammed Amine Togou, and Gabriel-Miro Muntean. An innovative algorithm for improved quality multipath delivery of virtual reality content. In *2020 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–6. IEEE, 2020.

[9] Daniel Wagner and Dieter Schmalstieg. Handheld augmented reality displays. In *IEEE Virtual Reality Conference (VR 2006)*, pages 321– 321. IEEE, 2006.

[10] John Patrick Sexton, Anderson Augusto Simiscuka, Kevin Mcguinness, and Gabriel-Miro Muntean. Automatic cnn-based enhancement of 360° video experience with multisensorial effects. *IEEE Access*, 9:133156– 133169, 2021.

[11] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[12] SangSu Choi, Kiwook Jung, and Sang Do Noh. Virtual reality applications in manufacturing industries: Past research, present findings, and future directions. *Concurrent Engineering*, 23(1):40–63, 2015.

[13] Ravi Pratap Singh, Mohd Javaid, Ravinder Kataria, Mohit Tyagi, Abid Haleem, and Rajiv Suman. Significant applications of virtual reality for covid-19 pandemic. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, 14(4):661–664, 2020.

[14] Suzhi Bi and Ying Jun Zhang. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Transactions on Wireless Communications*, 17(6):4177– 4190, 2018.

[15] Yi-Hsuan Kao, Bhaskar Krishnamachari, Moo-Ryong Ra, and Fan Bai. Hermes: Latency optimal task assignment for resource-constrained mobile computing. *IEEE Transactions on Mobile Computing*, 16(11):3056– 3069, 2017.

[16] Umber Saleem, Yu Liu, Sobia Jangsher, and Yong Li. Performance guaranteed partial offloading for mobile edge computing. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2018.

[17] Umber Saleem, Yu Liu, Sobia Jangsher, Xiaoming Tao, and Yong Li. Latency minimization for d2d-enabled partial computation offloading in mobile edge computing. *IEEE Transactions on Vehicular Technology*, 69(4):4472–4486, 2020.

[18] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28(13):2009, 2009.

[19] Bao Trinh, Liam Murphy, and Gabriel-Miro Muntean. An energyefficient congestion control scheme for mptcp in wireless multimedia sensor networks. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–7. IEEE, 2019.

[20] Pavel Mach and Zdenek Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, 19(3):1628–1656, 2017.

[21] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava. A survey of computation offloading for mobile systems. *Mobile networks and Applications*, 18(1):129–140, 2013.

[22] Weiwen Zhang, Yonggang Wen, Kyle Guan, Dan Kilper, Haiyun Luo, and Dapeng Oliver Wu. Energy-optimal mobile cloud computing under stochastic wireless channel. *IEEE Transactions on Wireless Communications*, 12(9):4569–4581, 2013.

[23] Yuvraj Sahni, Jiannong Cao, Lei Yang, and Yusheng Ji. Multi-hop multitask partial computation offloading in collaborative edge computing. *IEEE Transactions on*

Parallel and Distributed Systems, 32(5):1133– 1145, 2020.

[24] Sung-Tae Hong and Hyoil Kim. Qoe-aware computation offloading scheduling to capture energy-latency tradeoff in mobile clouds. In *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2016.

[25] Jiao Zhang, Li Zhou, Qi Tang, Edith C-H Ngai, Xiping Hu, Haitao Zhao, and Jibo Wei. Stochastic computation offloading and trajectory scheduling for uav-assisted mobile edge computing. *IEEE Internet of Things Journal*, 6(2):3688–3699, 2018.

EU project TRACTION.