A Privacy-aware Computation Offloading Method for Virtual Reality Application

Kai Peng[1], Peichen Liu[2] and Tao Huang[3]

[1]College of Engineering, Huaqiao University, Quanzhou, China
[1]College of Engineering, Huaqiao University, Quanzhou, China
[2]School of Computer Science and Technology, Silicon Lake College, Suzhou, China

Abstract

As a new technology, virtual reality (VR) is constantly enriching people's experience. VR application has high requirements for the performance VR devices, but the computing resources of VR devices are limited. Mobile edge computing provides an effective solution to solve the above issue by offloading VR application to edge servers (ESs) for processing. Nevertheless, the resources of ESs are limited and heterogeneous. And thus, it is necessary to consider the load balancing of ESs. In addition, user privacy protection in the process of computation offloading is another issue that needs to take into consideration. In view of this, in this paper, we investigate the computation offloading for VR applications with privacy protection. Technically, we propose a privacy-aware computation offloading method based on multi-objective optimization genetic algorithm to obtain the optimal strategy for VR application. Finally, it is shown that the proposed method is effective through extensive experiments.

Keywords

LaTeX Mobile Edge Computing, Virtual Reality, Privacy Protection, Computation Offloading

1. Introduction

With the development of communication and networking, the number of mobile users is increasing in a staggering speed. According to Oracle, there are more than 10 billion mobile devices connected to the Internet, and that number will grow to 22 billion by 2025 [1]. The explosive growth of mobile devices leads to a large request and the promising application prospect [2]. New technologies are beginning to be applied to mobile devices, such as virtual reality (VR), artificial intelligence (AI) and the Internet of Vehicles (IoV) [3]. Among these, as a prospective technology, VR is constantly enriching people's experience ([4],[5]). In practice, the VR applications are usually run over a wearable VR devices (VDs), which can record information about the movement and activity of VR users (VUs). Equipment manufacturers of VD usually need to consider physical size constraint, which equips with a small CPU and GPU with low-power consumption and low-capacity battery [6]. However, due to VR application requires real-time processing, it will consume a lot of computing resources. This poses a big challenge to VDs.

Fortunately, mobile edge computing (MEC) is a promising distributed computing paradigm ([7],[8]). Computation offloading of MEC has been well studied in ([9]-[10]). Inspired by this, offloading the VR application to edge servers (ESs), the time and energy consumption of VD can be significantly reduced. However, computing resources of the ES are limited, it will increase the waiting time and energy consumption of VDs when the number of tasks performed exceeds computing capacity of the ES [11]. Meanwhile, the resources of ESs are heterogeneous, the tasks should be reasonably distributed on ES cluster to avoid some of the ESs are overload [12]. Meanwhile, when the VU interacts with the VD, the application will collect the VU's private information, such as location, posture [13]. If placing data with privacy conflicts on the same ES, it is easy to cause the leakage of VU privacy information [14]. Therefore, it is necessary to consider the load balancing and the privacy constraints of placement of computation offloading for VR application.

To address above issues, the computation offloading considering privacy protection for VR applications are investigated in this work. The contributions of this paper can be summarized as follows.

1) We model the computation offloading problem as a multi-objective optimization issue, where the motion-to-photons latency and energy consumption of VD, as well as load balancing of ES are considered as the optimization objectives, and privacy protection is considered as a constraint.

2) A computation offloading method for VR applications based on multi-objective optimization genetic algorithm, named MCOVR, is proposed to address the above issue.

Woodstock'21: Symposium on the irreproducible science, June 07–11, 2021, Woodstock, NY
pkbupt@gmail.com (K. Peng); 18734166683@163.com (P. Liu); nuisthuangtao@163.com (T. Huang)
https://yamadharma.github.io/ (K. Peng)
0000-0002-0877-7063 (K. Peng); 0000-0001-7116-9338 (P. Liu); 0000-0002-9421-8566 (T. Huang)

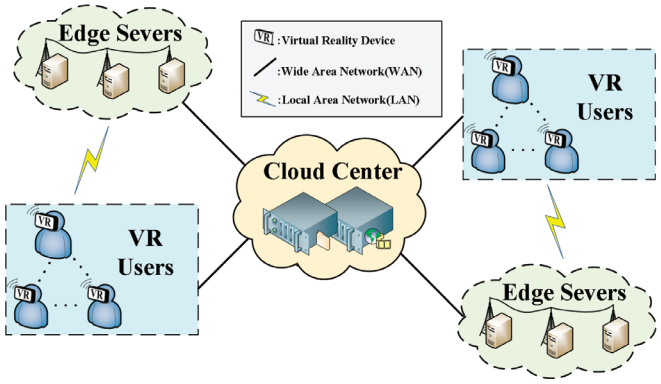


Figure 1: MEC-Enabled VR application architecture.

3) We conduct a large number of comparative experiments to prove that MCOVR can obtain the optimal computation offloading strategy for VR application.

The remaining of the paper is describe as follows. Firstly, section 2 describes the system model and defines the problem formulation. Secondly, section 3 introduces our proposed multi-objective computation offloading of VR application method. Then, section 4 demonstrates the experiment results. Finally, section 5 introduces the related work and section 6 describes the conclusion and the future work.

2. System Model and Problem Formulation

In this section, the architecture of MEC-enabled VR application is firstly introduced. Then, the model of VR application is established. Finally, the system model and problem formulation are described.

The MEC-enabled VR application architecture is shown in Figure 1. We assume that the cloud has infinite computing resources, and resources of ESs are finite and heterogeneous. These VR applications are executed by VDs or offloaded to the ES or to the cloud. VUs communicate with ESs via local area network (LAN) and with the cloud via wide area network (WAN).

Y is a set of VU, which is defined as $Y = \{1, 2, ..., y\}$. There are vn applications in V, which is defined as $V = \{v^1, v^2, ..., v^{vn}\}$. The task length requested by the v-th application of y-th user is denoted as $l_{y,v}$, and the task workload is denoted as $r_{y,v}$. Let X be the set of ES, denoted as $X = \{1, 2, ..., x\}$. In each ES, the computing resources of ES are represented as a number of i virtual machine(VM) xn, represented as $xn = \{xn^1, xn^2, ..., xn^i\}$. J is a set of computation offloading strategies, which is defined as $J = \{j_{1,1}, ..., j_{1,v}, j_{2,1}, ..., j_{y,v}\}$. And $j_{y,v}$ represents the offloading strategies of v-th application from y-th VU. $j_{y,v} = 0$ represents that the application is run by VD, $j_{y,v} \in (1, 2, ..., x)$ indicates the application is offloaded to ES, $j_{y,v} = x + 1$ represents the application is offloaded to cloud for processing.

2.1. Virtual Reality Application Model

In general, Augmented reality (AR) application can be represented as a directed acyclic graph(DAG) [15]. Similarly, VR can also be represented by a DAG. As shown in Figure 2, the VR model can be represented as follows. The VR application contains three independent operations, namely, rendering, calibration and processing. Thus, we use a layer containing three operations to represent each video frame, namely $node$. In addition, there are k video frames in a VR application, the $node_k$ represents the $k-th$ node in application.

2.2. Motion-to-Photons Latency Model

2.2.1. Executing latency

Execution latency is the time cost of processing VR applications on the three platforms, namely, VD, ESs, and cloud center. In this part, the executing time of v-th VR application of y-th user is represented as

$$S^e(j_{y,v}) = \begin{cases} \frac{r_{y,v}}{f_l} & j_{y,v} = 0 \\ \frac{r_{y,v}}{f_e} & j_{y,v} = 1 \ or \ 2, or \ldots or \ x \\ \frac{r_{y,v}}{f_c} & j_{y,v} = x + 1. \end{cases} \quad (1)$$

2.2.2. Transmission latency

The transmission latency is related to the computation offloading strategy of the two tasks. Let $P^c(c \in$

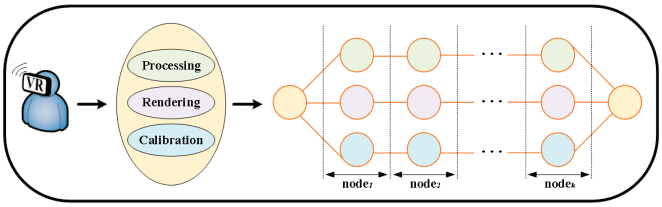


Figure 2: VR Application Model.

$\{1, 2, 3\}$) be the flag between two strategies, which are defined as

$$P^c = \begin{cases} Two\ applications\ are\ executed \\ by\ same\ platform & c = 1, \\ VD\ and\ ES \\ offloading & c = 2, \\ Cloud\ and \\ other\ platforms\ offloading & c = 3. \end{cases} \tag{2}$$

According to transmission strategy P^c, the transmission time $S^t(j_{y,v})$ of the v-th application of the y-th VU is shown as

$$S^t(j_{y,v}) = \frac{l_{y,v}}{B}, \tag{3}$$

where B represents the transmission bandwidth. The transmission bandwidth is divided into three cases which is denoted as

$$B = \begin{cases} \infty & P^c = 1, \\ B^e & P^c = 2, \\ B^c & P^c = 3. \end{cases} \tag{4}$$

2.2.3. Waiting latency

When the number of applications offloaded to the ES exceeds the number of VM xn, the newly coming tasks need to wait for the previous task to complete execution. The i-th VM in x-th ES is represented as a double-tuple $xn_i^x = (vmwk, tnum)$, where $vmwk$ represents the i-th VM collection and the $tnum$ represents the number of applications in VM. When the offloading strategy $j_{y,v} = x$, $r_{y,v}$ is offloaded to VM of x-th ES. Then, the $vmwk$ is updated by $vmwk = vmwk + l_{y,v}$ and the $tnum = tnum + 1$. Finally, the $S^w(j_{y,v})$ of v-th application from y-th VU is calculated as

$$S^w(j_{y,v}) = S^e(pre(j_{y,v})) \cdot \psi_{y,v,q}, \tag{5}$$

where $\psi_{y,v,q}$ is to determine whether $r_{y,v}$ needs to wait for the previous task, and the execution latency of previous task is represented as $S^e(pre(j_{y,v}))$.

2.2.4. Total latency

The total time $S(j_{y,v})$ is represented as the sum of all time, including the executing latency $S^e(j_{y,v})$, the transmission latency $S^t(j_{y,v})$, and the waiting latency $S^w(j_{y,v})$. The total latency is represented as

$$S(j_{y,v}) = S^e(j_{y,v}) + S^t(j_{y,v}) + S^w(j_{y,v}). \tag{6}$$

2.3. Energy Consumption Model

2.3.1. Executing energy consumption

The executing energy of MDs is related to the executing time, the executing time of v-th VR application of y-th user is represented as

$$S^e(j_{y,v}) = \begin{cases} \frac{r_{y,v}}{f_l} \cdot \wp^{active} & j_{y,v} = 0 \\ \frac{r_{y,v}}{f_e} \cdot \wp^{idle} & j_{y,v} = 1\ or\ 2, or \ldots or\ x \\ \frac{r_{y,v}}{f_c} \cdot \wp^{idle} & j_{y,v} = x + 1. \end{cases} \tag{7}$$

where $\wp^{active}$ represents active state of VD energy consumption and $\wp^{idle}$ represents idle state of VD energy consumption.

2.3.2. Transmission energy consumption

$N^t()$ represents transmission energy consumption, which can be obtained by multiplying the transmission power by the transmission time. $N^t()$ can be calculated by

$$N^t(j_{y,v}) = S^t(j_{y,v}) \cdot \wp^{trans}, \tag{8}$$

where $\wp^{trans}$ represents the energy consumption of transmission.

2.3.3. Waiting energy consumption

Waiting energy consumption is the energy consumption of VD when the task waits for the execution of the previous task. Waiting energy consumption is related to waiting time which can be obtained as

$$N^W(j_{y,v}) = S^W(j_{y,v}) \cdot \wp^{idle}. \tag{9}$$

2.3.4. Total energy consumption

The total energy consumption $N(j_{y,v})$ is represented as the sum of all energy consumptions which includes the executing energy consumption $N^e(j_{y,v})$, the transmission energy consumption $N^t(j_{y,v})$, and the waiting energy consumption $N^w(j_{y,v})$. The total energy consumption can be calculated by

$$N(j_{y,v}) = N^e(j_{y,v}) + N^t(j_{y,v}) + N^w(j_{y,v}). \quad (10)$$

2.4. Load Balancing Model

The ESs are heterogeneous means that the computing resources of ESs are not equal. Therefore, the workload of ESs needs to be balanced. And, ζ_v is a binary number to calculate the occupation of ES, which is calculated by

$$\zeta_v = \begin{cases} 1, & \text{if e}^{\text{x}} \text{ is occupied} \\ 0, & \text{Otherwise.} \end{cases} \quad (11)$$

And, η_v is a flag to represent whether the ES is occupied, which is represented as

$$\eta_v = \begin{cases} 1, & \text{if v existed in e}^{\text{x}} \\ 0, & \text{Otherwise.} \end{cases} \quad (12)$$

Firstly, the number of ES that are utilized is defined as $UE(j)$, which is represented as

$$UE(j) = \sum_{v=1}^{V} \zeta_v. \quad (13)$$

τ represent the amount of occupied ESs which is calculated by

$$\tau = \begin{cases} \frac{1}{o^i} \sum_{v=1}^{V} \sum_{y=1}^{Y} g_{y,v} \cdot \eta_v, & \zeta_v = 1 \\ 0, & \text{Otherwise.} \end{cases} \quad (14)$$

According to the resource utilization of each ES, the average resource utilization $\alpha(j)$ can be obtained by

$$\alpha(j) = \frac{1}{\mu} \cdot \sum_{v=1}^{V} \tau. \quad (15)$$

Finally, the load balancing of each ES is calculated by the squared difference between the resource utilization of each ES and the average resource utilization. The loading balancing $F(J)$ is calculated by

$$F(j) = \frac{1}{\mu} \cdot \sum_{v=1}^{V} [\alpha(j_{y,v}) - \tau(j_{y,v})]^2). \quad (16)$$

2.5. Privacy Model

As VR applications need to collect user information, VR privacy conflicts need to be considered. In this paper, we address privacy conflicts by placing applications on different ESs to avoid privacy leakage [14]. A graph $\hbar = (RW, CT)$ is used to describe the privacy conflicts, where RW represents the a set of computing services and CT denotes a set of privacy conflicting relation. A pair of conflict relations $(rw_y, rw_{y*})(rw_y, rw_{y*} \in RW)$ are used to indicate that VU information with privacy conflicts cannot be placed in the same ES. The conflict application of rw_v are represented as

$$rw_v|(rw_y, rw_{y*}) \in CT, y* = \{1, 2, ..., Y\}. \quad (17)$$

$HS = hs_1, hs_2, ..., hs_y(hs \in Y)$ represents the placement location ES for executing the applications. Afterwards, based on the acquired conflicting service set, the placed destination hs_y has a conflicting ES set, which is calculated by

$$h_y|(es_x \| es_x) \in rw_q, x = \{1, 2, ..., |rw_v|\}. \quad (18)$$

2.6. Problem Formulation

The main objectives of this study are to decrease the motion-to-photons latency and energy consumption of VDs, as well as the load balancing of ESs while preserving the VU's privacy. The problem formulations are defined as follows.

$$Min \sum_{y=1}^{Y} \sum_{v=1}^{V} S(j_{y,v}); \forall j \in \{1, 2, 3, \ldots, x\}. \quad (19)$$

$$Min \sum_{y=1}^{Y} \sum_{v=1}^{V} N(j_{y,v}); \forall j \in \{1, 2, 3, \ldots, x\}. \quad (20)$$

$$Min \sum_{y=1}^{Y} \sum_{v=1}^{V} F(j_{y,v}); \forall j \in \{1, 2, 3, \ldots, x\}. \quad (21)$$

$$s.t. j_{y,v} \in \{0, 1, 2, ..., x+1\}. \quad (22)$$

$$hs_y \notin h_y. \quad (23)$$

$$\sum_{y=1}^{Y} \sum_{v=1}^{V} S(j_{y,v}) \leq S^{QoS}. \quad (24)$$

3. Method Design

In this section, we describe the details of our proposed multi-objective computation offloading of VR application (MCOVR). MCOVR is based on MOMBI [16].

3.1. Initialization

The initialization of the algorithm includes the definition of key parameters and the generation of the initial population. Firstly, the first-generation population Z_0 of size t^z is generated randomly. Secondly, MCOVR defines some parameters, such as crossover probability L^c, mutation probability L^m, the number of iterations D_{max}, and current iteration index d. Finally, the algorithm defines a size of t^q achieve set Q_0 that keeps the achieve solution. Moreover, tournament selection, crossover, as well as mutation is executed on Z_0 to generate new population Z_0^*.

3.2. Crossover and Mutation

In the crossover operation, the algorithm randomly exchanges a certain point value of two offloading strategies. Through crossover operation, the algorithm can obtain diversity solutions. In the mutation operation, mutations will slightly change certain values on the offloading strategy. And mutation operator can reduce the occurrence of local optimal and avoid early convergence.

3.3. R2 Ranking and Reference Points

In the MCOVR, algorithm divides the population through the R2 indicator to achieve non-dominated sorting, namely, R2 ranking. In R2 indicator, the objectives are measured based on the weighted Tchebycheff method and processed by normalization. W is represented as the weight of each object, A is represented the Pareto solution set, and ϑ represents the utility functions. Therefore, the R2 indicator is defined as follows.

$$R2i(A, W) = -\frac{1}{|W|} \sum_{\omega \in W} \min_{a \in A} \vartheta(a). \tag{25}$$

Additionally, in Equation (25), when each population is calculated by the R2 indicator, the reference point of the ϑ needs to be updated. The maximum value c_d^{max} and minimum value c_d^{min} of the objective need to be found. Through Equation (25) and reference point updating policy, the r2 ranking can be calculated as

$$rank_{Z_d} = \bigcup_{\omega \in W} \min_{a \in A/B_k} \{ \max_{i \in 1,\ldots,t} w_i | \frac{f_d(j_{y,v}) - c_d^{min}}{c_d^{max} - c_d^{min}} | \}, \tag{26}$$

where $f_d(j_{y,v})$ represents the objective function in d-th iteration.

3.4. Tournament Selection

The two populations Z_d and Z_d^* generated by the algorithm need to select the number of t^z as the next-generation population Z_{d+1}, where d<D_{max}. In addition, based on R2 ranking of Z_d, by comparing the objective function values of the two solutions and select the optimal value of the objective function as the parent solution.

3.5. Method Overview

This method aims to reduce the time consumption and energy consumption of VDs as well as load balancing of ESs jointly. The flow chart represents the execution sequence of our proposed MCOVR is shown in Figure 3.

4. Experiment Evaluation

In this section, we present the experimental evaluation of our method. Firstly, the experiment setting is introduced, which includes comparative methods and key parameters. Then, the experimental results and discussion are described.

4.1. Experimental Setting

Two comparative computation offloading methods are proposed. The details of the two methods is as follows.

Benchmark: All applications are offloaded to the three platforms for processing randomly, named as Benchmark.

First Come First Service (FCFS): All applications are offloaded to the three platforms for processing orderly, named as FCFS.

The key parameter value is described in Table 1. We execute these offloading methods by JAVA over Win10 64 OS with 8 Intel Core i7-10850H 3.60 GHz CPU processors, Nvidia RTX2070 GPU processors, and 32GB RAM.

4.2. Experiment evaluation result and discussion

We design three different kinds of comparative experiments to verify the effectiveness of MCOVR. All the experiments are performed 10 times, and the result is the average value.

The comparison of motion-to-photons latency is presented in Figure 4. Figure 4(a) testifies the motion-to-photons latency of multi-VU condition. As the number of VUs increases, the difference of the latency consumed among the three methods is getting larger. Our proposed

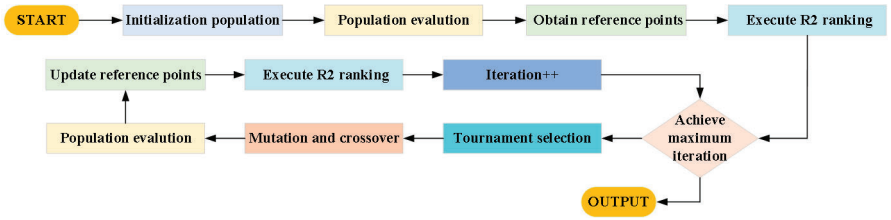


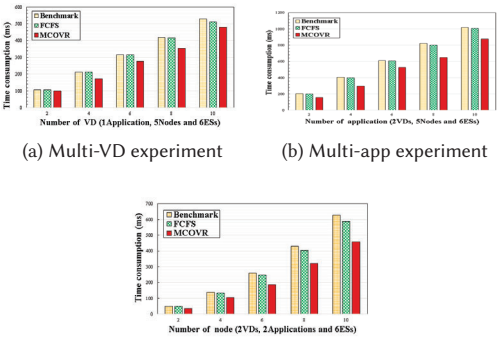
Figure 3: Flow chart of MCOVR.

Table 1
Experimental Parameter.

Key Parameter	Value
Idle power of MDs	0.05W
Active power of MDs	0.6W
Transmitting power of MDs	0.6W
The processing frequency of MD	500MHZ
The processing frequency of ES	2000MHZ-2400MHZ
The processing frequency of ES	5000MHZ
The bandwidth of LAN	220kb/s
The bandwidth of WAN	150kb/s
Number of VMs for each ES	15-20

MCOVR achieves the minimum latency consumption among the three methods. The latency consumption experimental result of the influence of different applications and different nodes are represented in Figure 4(b) and Figure 4(c). MCOVR reduces latency significantly and shows the slowest growth trend among the three methods. It can be deduced that MCOVR outperforms FCFS and Benchmark in terms of different situations.

The energy consumption is obtained by Equation



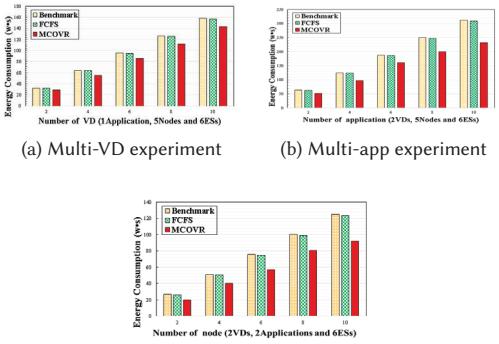
(a) Multi-VD experiment (b) Multi-app experiment

(c) Multi-node experiment

Figure 4: Comparison of the motion-to-photon latency.

(10). The energy consumption is related to time consumption which has the same growth trend with time consumption. Figure 5 demonstrates the comparison result of energy consumption. The multi-VD, multi-application and multi-node comparison of energy consumption is shown as Figure 5(a)-Figure 5(c). It is can be seen that MCOVR is more energy-efficient than the other two methods.

The load balancing can be obtained by Equation



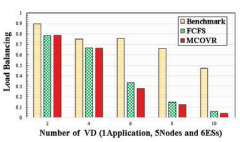
(a) Multi-VD experiment (b) Multi-app experiment

(c) Multi-node experiment

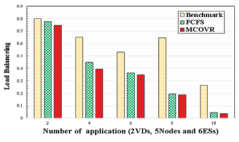
Figure 5: Comparison of the energy consumption.

(16). It is a negative indicator, that means the low value of load balancing represents balanced tasks allocation in the ES cluster. The load balancing comparison result of three methods is shown in Figure 6. As shown in Figure 6(a) and Figure 6(b), with the number of tasks increases, MCOVR and FCFS show a downward trend and the difference between the two methods is little. Due to the offloading strategies of Benchmark are generated randomly. Therefore, the results of Benchmark are unpredictable and irregular. As shown in Figure 6(c), as the number of tasks increases, more task are offloaded to the ESs, and the load balancing value decreases of these three methods.

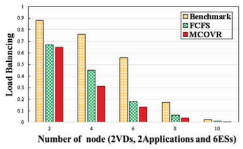
Above all, it is observed that our proposed method can obtain the optimal solution in comparison to the three experiments.



(a) Multi-VD experiment



(b) Multi-app experiment



(c) Multi-node experiment

Figure 6: Comparison of the load balancing.

5. Related Work

Many efforts have been make to solve the computation offloading in MEC. Bi et al. [9] considered energy consumption with the maximum number of memory constraint in MEC. An efficient algorithm based on a penalty function method is proposed to handle constraint. Xu et al. [10] jointly considered both IoV energy consumption and load balancing of ES problem. They proposed a genetic algorithms-based method to solve it. Feng et al. [17] considered the computation performance for offloading in the wireless powered MEC. Li et al. [18] studied a joint optimization problem of computation offloading and privacy preservation in the MEC system and proposed a privacy-aware online learning algorithm to solve this challenge. Ko et al. [19] studied the computation offloading problem in smart cities based on the privacy entropy model. A constrained Markov decision process method is developed to solve location privacy leakage. The above-mentioned studies provided a good view of the computation offloading in MEC and inspired us to study the computation offloading of VR applications.

A few studies take into account MEC-enabled VR applications. Zhang et al. [20] proposed an edge-cloud architecture of VR application that frame rendered on ES or cloud. They proposed a service placement algorithm for multi-user VR applications that makes placement decisions, based on QoS and VUs mobility patterns. Zhu et al. [21] focused on dynamic rendering-module placement problem for VR streaming. To reduce such rendering latency, they proposed a prediction-based pre-rendering mechanism at the ES. In [22], the authors investigated the problem of providing service using the MEC network architecture with a frame rendering. They solved an online service placement problem by leveraging model predictive control and overcoming conflicting system objectives. In [23], the authors focused on the online offloading interactive VR application. They considered the constraints of the delay requirements, the maximum frame per second demands, the total bandwidth and rendering resources limits and proposed programming to optimize execution efficiency of VR application.

Different from the above studies, we investigates how to make privacy-aware computation offloading decisions to reduce the motion-to-photons latency and energy of VR applications and load balancing of ESs while considering VUs privacy-preserving. Correspondingly, we solve this issue using a multi-objective optimization method.

6. Conclusion

We studied the problem of multi-objective computation offloading of VR applications. Correspondingly, we propose a method named MCOVR to minimize the motion-to-photons latency and energy consumption of VD and the load balancing of ESs jointly. We also considered privacy conflicts in VR applications and using the privacy placement model to protect user data information security. Finally, the experimental results have shown that our proposed method is effective.

In future work, we will study mobility-aware computation offloading for augmented reality applications in smart cities.

7. Acknowledgments

Acknowledgments

This work is supported by the National Science Foundation of China (Grant No.61902133), the Natural Science Foundation of Fujian Province (Grant No.2018J05106), the Fundamental Research Funds for the Central Universities(ZQN-817), Quanzhou Science and Technology Project(No.2020C050R).

References

[1] Mustafa, T., & Varol, A. (2020, June). Review of the Internet of Things for Healthcare Monitoring. In 2020 8th International Symposium on Digital Forensics and Security (ISDFS) (pp. 1-6). IEEE.

[2] Wijethilaka, S., & Liyanage, M. (2021). Survey on network slicing for Internet of Things realization in 5G networks. IEEE Communications Surveys & Tutorials, 23(2), 957-994.

[3] Liu, J., & Zhang, Q. (2020). To improve service reliability for AI-powered time-critical services using imperfect transmission in MEC: An experimental

study. IEEE Internet of Things Journal, 7(10), 9357-9371.

[4] Chakareski, J. (2020). Viewport-Adaptive Scalable Multi-User Virtual Reality Mobile-Edge Streaming. IEEE Transactions on Image Processing, 29, 6330-6342.

[5] Hu, F., Deng, Y., Saad, W., Bennis, M., & Aghvami, A. H. (2020). Cellular-connected wireless virtual reality: Requirements, challenges, and solutions. IEEE Communications Magazine, 58(5), 105-111.

[6] Hsu, Y. H., Cheng, J. H., Liao, K. Y., Wang, Y. S., Chen, T. H., Chen, H. Y., ... & Liao, W. (2020, September). NTU Smart Edge for Wireless Virtual Reality. In 2020 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan) (pp. 1-2). IEEE. 1

[7] Sukhmani, S., Sadeghi, M., Erol-Kantarci, M., & El Saddik, A. (2018). Edge caching and computing in 5G for mobile AR/VR and tactile internet. IEEE MultiMedia, 26(1), 21-30.

[8] Shi, S., Gupta, V., Hwang, M., & Jana, R. (2019, June). Mobile VR on edge cloud: a latency-driven design. In Proceedings of the 10th ACM Multimedia Systems Conference (pp. 222-231).

[9] Bi, J., Yuan, H., Duanmu, S., Zhou, M. C., & Abusorrah, A. (2020). Energy-optimized Partial Computation Offloading in Mobile Edge Computing with Genetic Simulated-annealing-based Particle Swarm Optimization. IEEE Internet of Things Journal.

[10] Xu, X., Gu, R., Dai, F., Qi, L., & Wan, S. (2020). Multi-objective computation offloading for internet of vehicles in cloud-edge computing. Wireless Networks, 26(3), 1611-1629.

[11] Violos, J., Psomakelis, E., Tserpes, K., Aisopos, F., & Varvarigou, T. (2019, May). Leveraging user mobility and mobile app services behavior for optimal edge resource utilization. In Proceedings of the International Conference on Omni-Layer Intelligent Systems (pp. 7-12).

[12] Xu, X., Liu, X., Xu, Z., Wang, C., Wan, S., & Yang, X. (2019). Joint optimization of resource utilization and load balance with privacy preservation for edge services in 5G networks. Mobile Networks and Applications, 1-12.

[13] O'Brolcháin, F., Jacquemard, T., Monaghan, D., O'Connor, N., Novitzky, P., & Gordijn, B. (2016). The convergence of virtual reality and social networks: threats to privacy and autonomy. Science and engineering ethics, 22(1), 1-29.

[14] Xu, Z., Gu, R., Huang, T., Xiang, H., Zhang, X., Qi, L., & Xu, X. (2018). An IoT-oriented offloading method with privacy preservation for cloudlet-enabled wireless metropolitan area networks. Sensors, 18(9), 3030.

[15] Chen, X., & Liu, G. Energy-Efficient Task Offloading and Resource Allocation via Deep Reinforcement Learning for Augmented Reality in Mobile Edge Networks. IEEE Internet of Things Journal. 2021.

[16] GÓmez, R. H., & Coello, C. A. C. (2013, June). MOMBI: A new metaheuristic for many-objective optimization based on the R2 indicator. In 2013 IEEE Congress on Evolutionary Computation (pp. 2488-2495). IEEE.

[17] Feng, J., Pei, Q., Yu, F. R., Chu, X., & Shang, B. (2019). Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint. IEEE Wireless Communications Letters, 8(5), 1320-1323.

[18] Li, T., Liu, H., Liang, J., Zhang, H., Geng, L., & Liu, Y. (2020, September). Privacy-Aware Online Task Offloading for Mobile-Edge Computing. In International Conference on Wireless Algorithms, Systems, and Applications (pp. 244-255). Springer, Cham.

[19] Ko, H., Lee, H., Kim, T., & Pack, S. (2020). LPGA: Location Privacy-Guaranteed Offloading Algorithm in Cache-Enabled Edge Clouds. IEEE Transactions on Cloud Computing.

[20] Zhang, W., Chen, J., Zhang, Y., & Raychaudhuri, D. (2017, October). Towards efficient edge cloud augmentation for virtual reality mmogs. In Proceedings of the Second ACM/IEEE Symposium on Edge Computing (pp. 1-14).

[21] Zhu, Z., Jiang, N., Guo, T., & Wei, S. (2019, November). Virtual reality streaming at the edge: a power perspective: poster. In Proceedings of the 4th ACM/IEEE Symposium on Edge Computing (pp. 316-318).

[22] Zhang, Y., Jiao, L., Yan, J., & Lin, X. (2019). Dynamic service placement for virtual reality group gaming on mobile edge cloudlets. IEEE Journal on Selected Areas in Communications, 37(8), 1881-1897.

[23] Zhu, H., Li, Y., Chen, Z., & Song, L. (2021, March). Mobile Edge Resource optimization for Multiplayer Interactive Virtual Reality Game. In 2021 IEEE Wireless Communications and Networking Conference (WCNC) (pp. 1-6). IEEE.