Reinforcement Learning Based Offloading Framework for Computation Service in the Edge Cloud and Core Cloud

Nasif Muslim and Salekul Islam
United International University Dhaka, Bangladesh
Email: {nasif, salekul}@cse.uiu.ac.bd

Jean-Charles Grégoire
INRS-EMT, Montréal, Canada
Email: gregoire@emt.inrs.ca

***Abstract*—Smartphones have increasingly become indispensable tools of our everyday lives, with extensive applications beyond communications, from utilitarian to entertaining. As such, demands on the technology remain stringent, leading to a limitation in performance and battery lifetime, requiring approaches such as computation offloading to improve the user experience for computation-intensive tasks such as gaming. This work presents the use of Reinforcement Learning in an offloading framework that provides smartphones with the ability to decide whether to perform computations on the smartphone or on the remote Cloud (Edge and Core) to minimize process. Several scenarios have been used to produce simulations that demonstrate that the proposed algorithm can operate efficiently in a dynamic Cloud computing and networking environment.**

***Index Terms*—reinforcement learning, cloud computing, computation offloading**

I. Introduction

Breakthroughs in mobile computing and wireless technology have allowed smartphones to dramatically impact human and social life by ever-improving processor performance and connectivity. These devices now support diverse applications ranging from simple information display to computationally expensive voice and face recognition. Battery technology, however, remains a major limiting factor of the smartphone progress: the increase in energy capacity (i.e., mA-hour) of the prevailing lithium-ion batteries has not been able to keep up with increases in power consumption.

A promising solution to overcome many of the limitations of smartphone performance lies in computation offloading, the process of migrating compute-intensive tasks to the remote Cloud (Edge and Core). "Core Cloud" refers to commercial computing infrastructures (i.e., data centers) situated further away from clients or users. "Edge Cloud" points to commercial or private datacenter located closer to clients or users. Globally, access to a remote cloud is performed via the Internet core through high-performance access (e.g., cellular) infrastructures.

Application types and hardware configuration greatly influence the decision to offload. Thus, Kumar *et al.* [1] suggest that offloading a computation task to the Cloud may not always be beneficial. Experimental results show that computation offloading is more beneficial for a chess game than for an image retrieval process due to the high amount of computation that the former requires. This paper presents an in-depth study of the offloading process and analyzes how different parameters affect each other in decision-making. Furthermore, we apply Reinforcement Learning (RL) in our proposed framework, and we use Face recognition, a computation and data-intensive application, to evaluate the soundness of our offloading framework, extending in the process work presented earlier in [2]. Face recognition is nowadays commonly applied in authentication as well as for passive surveillance, and recent related research has exploited deep-learning-based approaches due to accuracy rates above 99.80% [3]. Yet executing deep learning algorithms is computationally intensive and especially taxing for smartphones; for example, [4], and [5] report processing rates of less than one video frame per second using Tensorflow's Inception deep learning model on a typical Android phone. There is a noted need for a flexible, distributed infrastructure to enable cooperation between end-user devices (e.g., smartphones) and the remote Cloud infrastructures (Edge and Core) that provide more computational power, independent of any specific configuration of Cloud (Edge and Core), end-user device, and face recognition technique [6], [7].

The main contributions of this paper are as follows:

- We propose an offloading framework that makes its decisions using RL. This framework can obtain the optimal policy for local or remote allocation of task processing from evolving environment parameters without any complicated computation of an offloading solution. Simulation results show

Manuscript received October 18, 2021; revised February 10, 2022.

© 2022 J. Adv. Inf. Technol.
doi: 10.12720/jait.13.2.139-146

that the offloading decision converges quickly, adapts to different conditions, and outperforms a random offloading decision algorithm.

- Unlike most earlier research, which has focused on one or two parameters for offload decision making [8], our proposed offloading framework includes considerations for the (monetary) cost of providing the computing resources rather than simply computation costs (i.e., compute cycles). The end-user device can make the decision to offload directly to the Edge Cloud or Core Cloud without relaying to the Computing Access Points (CAPs) [9]. Using the experiments, we show that the proposed framework can realize savings in the combination of these parameters (i.e., processing time, end-device energy consumption and monetary cost).

The rest of the paper is organized as follows: Section II presents an overview of previous work related to Edge Cloud computing and offloading frameworks. In Section III, we present the offloading problem and a solution based on multi-armed bandits. In Section IV, we present and discuss simulation results under synthetic scenarios. Finally, in Section V, we elaborate on possible extensions of our framework for future work.

II. Related Works

We have witnessed a proliferation of data centers, not only increasing the availability of low latency, high-speed service connectivity and cheap compute cycles but also decreasing the cost of storage and overall supporting the rapid progress of Cloud computing. Nowadays, the Edge Cloud deployed at the edge of the Internet in direct co-operation with ISPs consists of a number of multiple smaller, generic clouds run by various operators. The simplified Edge Cloud architecture [10] is shown in Fig. 1.

On the application side, computationally expensive models such as Deep Neural Networks (DNNs) have become popular for tasks like face detection, language translation, autonomous driving, etc. More specifically, Chemodanov *et al.* [11] have proposed a novel policy-based computational offloading scheme based on the trade-off between performance and cost parameters, which decides to offload either to an Edge, Core cloud, or apply function-centric computing- decomposing applications into smaller functions that can be deployed on to the Edge and Core Cloud–for real-time video analytics. This work has not taken into consideration computation time variations of distinct DNN based object recognition systems, as well as random network delay and server load. Chinchali *et al.* [12] have proposed an offloading framework that accounts for both local computation accuracy and offloading latency using a Deep Reinforcement Learning algorithm. The authors' proposed solution is, however, based on static communication ad computational delays. In our previous work, Nasif *et al.* [2] proposed an offloading framework based on Deep Reinforcement Learning that provides smartphones with the ability to make decisions for offloading. Shakarami *et al.* [13] proposed a deep learning-based hybrid approach for computation offloading. However, training DRL-based offloading is time and resource expensive compared with the currently proposed RL-based offloading framework.

Shahidinejad *et al.* [14] proposed a hybrid approach for computation offloading. In this paper, the smartphone makes the decision to offload in the Edge Cloud using Learning Automata (LA), which is closely related to the Reinforcement Learning (RL) approach. Additionally, the master Edge Cloud collects resources information from the Edge information system and Cloud information system. Using this information, the master Edge Cloud makes the decision for processing in the Edge Cloud or Core Cloud. However, in our proposed offloading framework, the smartphone makes the ultimate decision for offloading, which saves additional processing in the master Edge Cloud.

Shahidinejad *et al.* [15] also proposed a computation offloading technique based on DRL. The Federated learning technique is used to improve the offloading decision model inside the smartphone. The cost for implementing federated learning is higher because it requires significantly more smartphone processing power and memory to train the model. Moreover, Federated learning requires periodic communication between the smartphone and the server during the learning process, which requires high bandwidth connections.

Most research on offloading decisions to date has focused on optimizing the energy and/or time consumption of offloaded applications while generally ignoring decision-making algorithms overhead. Instead, we focus here on making offloading decisions dynamically, in time-varying environments, and on reducing the computational complexity of the offloading decision algorithm. This objective differs from earlier investigations by focusing on the combination of three parameters (processing time, energy consumption of the smartphone, monetary cost) to decide if and where to offload.

III. Model Offloading Problem and Solution Based on Multi-armed Bandits

We assume that devices and datacenters are connected to or reachable from wireless networks. However, we do not assume prior knowledge of the end-devices hardware configuration, infrastructures underlying either form of Cloud, or network-related information (e.g., bandwidth, hop-count, round-trip-time). The smartphone itself must decide whether to process tasks locally or remotely, either in the Edge or Core Cloud. Choosing the best-suited location to process a task is not a trivial task, mainly when the required information is not available. The problem becomes more challenging in a dynamic environment where computation time and network characteristics vary over time.

© 2022 J. Adv. Inf. Technol.

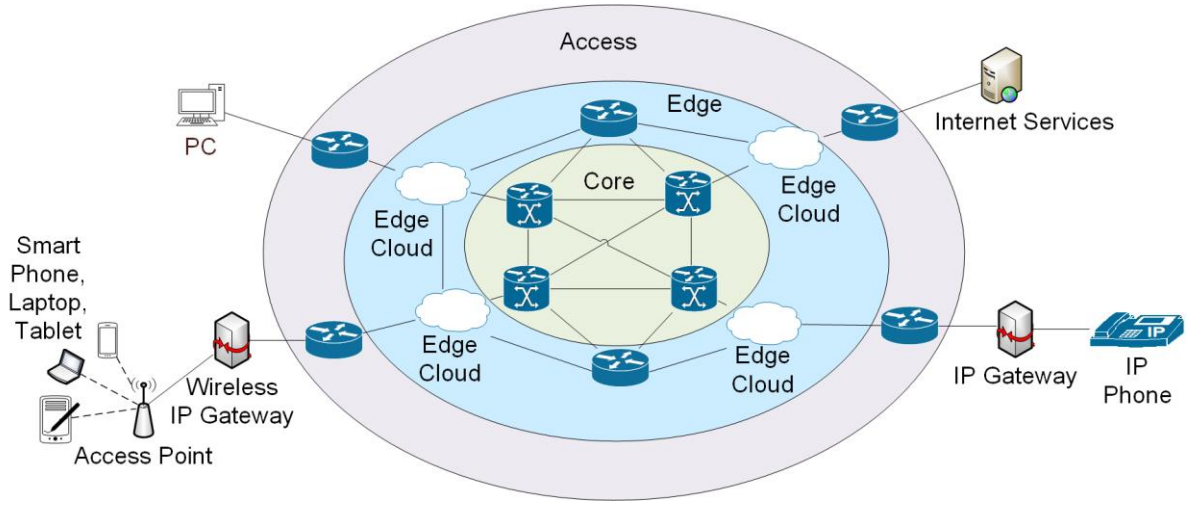


Figure 1. Edge cloud architecture.

Multi-objective optimization is challenging in general. Such problems are generally formulated as Mixed Integer Programming (MIP) problems due to the existence of binary offloading variables. To solve the MIP problems, branch-and-bound algorithms [16] and dynamic programming [17] have been adopted. However, both alternatives tend to converge very slowly to a satisfying local optimum.

Reinforcement Learning (RL) [18] has been used as an effective solution for multi-objective optimization problems [19]. The model-free approach of RL is capable of learning optimal offloading policies based solely on the features included in the environment. By utilizing reward feedback from the time-variant environment, the RL agent can adjust its policy to achieve the optimization of resources. To solve the selection of the device to perform processing tasks, the offloading problem is modeled using the Multi-armed Bandit (MAB) problem, which is a variant of RL with a single state.

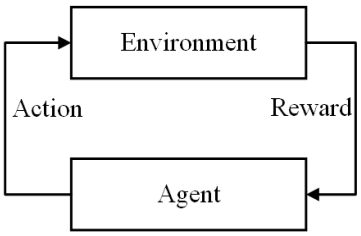


Figure 2. Multi-arm bandit problem - agent's action and environment's reply.

Multi-Arm Bandit is a classic RL problem whose main elements include agent, reward, and environment. The agent decides what action to take, and the environment will then provide feedback to the agent in the form of a reward. Based on the experience of interacting with the

environment, the agent will try to learn and create a set of rules on how to behave in this environment to get maximum reward. Fig. 2 shows the block diagram of the Multi-arm bandit problem.

RL utilizes purely evaluative feedback to indicate how good the action taken was. For that reason, it requires maintaining a balance between 'exploration', actively trying new ways, and 'exploitation', the process of applying things that have worked best in the past. The following subsections will explain this strategy and reward calculation for the proposed algorithm.

A. Exploitation and Exploration

Upper Confidence Bound (UCB-1) [20] is a widely used solution method for the multi-armed bandit problem that keeps the balance between 'exploitation' and 'exploration'. UCB-1 makes the selection of an arm or device based on uncertainty. In other words, the greater the uncertainty about an arm the more important it is to explore that arm. The confidence interval of each device is calculated using equation (1).

$$\text{Confidence interval } \Delta_i(n) = c\sqrt{\frac{\ln(n)}{N_i}} \quad (1)$$

It shows that each time device i is selected, the uncertainty is presumably reduced: N_i increments as it appears in the denominator and the uncertainty term decreases. However, each time a device other than i is selected n increases as it appears in the denominator, but N_i does not. Therefore, the uncertainty estimate increases. UCB-1 pseudo-code is presented in Algorithm 1.

Algorithm 1 Upper Confidence Bound (UCB-1)

1: **for** n in range (number of device): **do**
2: The average reward of device i up to iteration n:
3: $\overline{reward_i(n)} = \frac{reward_i(N)}{N_i(n)}$
where N is number of time device i is selected over n iteration steps
4: $\text{Confidence interval } \Delta_i(n) = c\sqrt{\frac{\ln(n)}{N_i}}$
where c is degree of exploration
5: $\text{Device selection} = \max(\overline{reward_i(n)} + \Delta_i(n))$
6: **end for**

© 2022 J. Adv. Inf. Technol.

However, UCB-1 policies are not appropriate for environments where the evolution of rewards undergo abrupt changes [21], i.e., non-stationary. Discounted UCB (D-UCB) [22] solves this problem, where average rewards are computed based on past values with a discount factor giving more weight to recent observations. D-UCB pseudocode is presented in Algorithm 2. Algorithm 3 provides the pseudocode for offload decision making based on D-UCB.

Algorithm 2 Discounted UCB (D-UCB)

1: **for** n in range (number of device): **do**
2: The average reward of device i up to iteration n:
3: $\overline{reward_i(n)} = \frac{\sum_{s=1}^{n} \gamma^{n-s} * reward_i(n)}{\sum_{s=1}^{n} \gamma^{n-s}}$ where, N is number of time device i is selected over n iteration steps.
4: $\text{Confidence interval } \Delta_i(n) = c\sqrt{\frac{\ln(n)}{N_i}}$ where c is degree of exploration
5: Device selection = max ($\overline{reward_i(n)} + \Delta_i(n)$)
6: **end for**

Algorithm 3 Offload decision making

reward memory: to store [choice, reward]
memory: to store [device, number of face, time, energy, cost]
number of device = 3
function MAIN:
  for n in range (iteration): **do**
    choice = 0 max upper bound = 0
#calculate the UCB value for each device, find out which one is maximum, that will be the selected for offload
    for i = 1 to 3 **do**
      if *numbers of selections[i]* > 0 **then**
average-reward = Calculate discounted empirical average reward for i-th device from *reward memory[choice, reward]*
end if
counts[i] = Calculate number of times i-th device is selected from *reward memory[choice, reward]*
      delta_i = 2 / 3 × $\sqrt{\ln(n)/counts[i]}$
      upper_bound = average_reward + delta_i
      else upper bound = infinite
      end if
      if *upper bound* > *max upper bound* **then**
        max upper bound = upper bound
        choice = i
      end if
  end for
# Offload processing task to the selected device and collect corresponding 'time', 'energy', 'cost' information
 time, energy, cost = OFFLOAD(choice, number of face)
 memory UPDATE
 # Calculate reward based on selected device for processing task Offload
 choice reward = REWARD(choice, number of face, memory)
 reward memory UPDATE
 end for
end function
function OFFLOAD(choice, number_of_face):
 if choice == 0 **then**
  process in the smartphone and measure $T_{smartphone}$, $E_{smartphone}$, $C_{smartphone}$
 else if choice == 1 **then**
  process in the Edge Cloud and measure T_{edge}, E_{edge}, C_{edge}
 else if choice == 2 **then**
 process in the Core Cloud and measure T_{core}, E_{core}, C_{core}
 end if
end function

B. Reward

The proposed algorithm, once a device is selected and computation task processing is performed, collects the corresponding processing time, energy consumption and monetary cost information. For example, when the smartphone is selected for task processing then processing time $T_{smartphone}$, energy consumption $E_{smartphone}$ and monetary cost $C_{smartphone}$ of the smartphone are collected for the current iteration θ. Next, a matrix is constructed that includes three choices (end-device, Edge and Core) and their corresponding three criteria values (shown in Table I). The criteria value of other choices (Edge Cloud and Core Cloud) which are not selected in this iteration are calculated by taking the average value from the previous $\theta - 1$ iteration (shown in Equation (2)). The reward calculation pseudocode is presented in Algorithm 4.

Algorithm 4 Reward calculation

Calculate normalized score of $T_{smartphoneN}$, T_{edgeN}, T_{coreN}
Calculate normalized score of $E_{smartphoneN}$, E_{edgeN}, E_{coreN}
Calculate normalized score of $C_{smartphoneN}$, C_{edgeN}, C_{coreN}
# Calculate preference score
if choice == 0 **then**
  smartphone_performance = TsmartphoneN + EsmartphoneN + CsmartphoneN
  return smartphone_performance
if choice == 1 **then**
  edge_performance = TedgeN + EedgeN + CedgeN
  return edge_performance
if choice == 2 **then**
  core_performance = TcoreN + EcoreN + CcoreN
  return core_performance
end if
end function

© 2022 J. Adv. Inf. Technol.

TABLE I. CRITERIA VALUE OF CHOICES SHOWN IN THE MATRIX AT Θ ITERATION

	Criteria		
Choice	Time (s)	Energy (mAh)	Cost ($)
Smartphone processing	x_{11}	x_{12}	x_{13}
Edge Cloud processing	$x_{21}avg,\ \theta$	$x_{22}avg,\ \theta$	$x_{23}avg,\ \theta$
Core Cloud processing	$x_{31}avg,\ \theta$	$x_{32}avg,\ \theta$	$x_{33}avg,\ \theta$

$$x_{ij}avg, \theta = \frac{\sum_{m=1}^{\theta-1} x_{ij}, m}{p} \quad (2)$$

where x_{ij} is the *j-th* criterion value of the *i-th* choice, m is the iteration index where choice *i* is selected, p is total number of times the *i-th* choice is selected in θ-1 iteration.

Then, the normalized matrix for criteria (shown in Equation (3)) is calculated using linear scale transformation (max method) (shown in Equation 4 and 5) [23].

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (3)$$

$$\text{For benefit criteria } r_{ij} = \frac{x_{ij}}{x_j^{Max}} \quad (4)$$

$$\text{For cost criteria } r_{ij} = 1 - \frac{x_{ij}}{x_j^{Max}} \quad (5)$$

where, x_j^{Max} is the maximum value among choices for criteria j = 1, 2, 3.

Table I shows the value of choices in the matrix at θ iteration. These values are normalized using the Equation (4) and (5). After normalization, Equation (3) shows the normalized value in the matrix format. Finally, Equation 6 shows how the reward value of choice is calculated by summing the normalized score of each criterion.

$$R_i = \sum_{j=1}^{n} r_{ij} \quad (6)$$

Finally, the reward value for local processing in the smartphone is calculated. If smartphone processing minimizes processing time, energy consumption, monetary cost then it will receive a higher reward value, which will be added to the corresponding cumulative reward value of the smartphone. The location for task processing will be selected based on the average reward, which is calculated from the cumulative reward value. Therefore, if a device minimizes processing time, energy consumption, and monetary cost compared to other devices, it will be selected for task processing.

IV. SIMULATION RESULT

For face recognition, the Inception v3 [24] model of the Tensorflow [25] is used in a smartphone, Edge Cloud and Core Cloud. The Inception v3 is a deep convolutional architecture designed for classification tasks on ImageNet [26] dataset. It consists of 1.2 million RGB images from 1000 classes. Table II lists the details of the simulation parameters, most of which reflect real measurements.

In the first experiment, the behavior of UCB-1 and D-UCB are tested by investigating their performance in a dynamic environment. In this case, the computation time $T_{edgeserver}$ is varied. Fig. 3 shows the variation of Edge Cloud server computation time $T_{edgeserver}$ for face recognition in the remote Cloud. Fig. 4 shows that the selection of Edge Cloud as a computation device changes over time due to the variation of Edge server computation time $T_{edgeserver}$. However, the response of UCB-1 (Fig. 4a) is slow compared to D-UCB (Fig. 4b) to detect the breakpoints (iteration = 300 and 600). UCB-1 and D-UCB respond to the first breakpoint at iteration = 430 and 400 respectively by selecting Core Cloud as an optimal device for offloading. It takes a long time after the second breakpoint by the UCB-1 to select Edge Cloud (iteration = 1800) compared with D-UCB (iteration = 690).

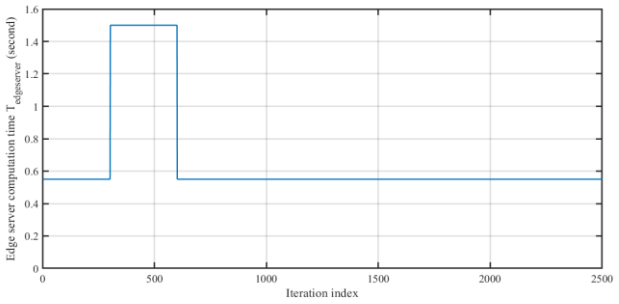
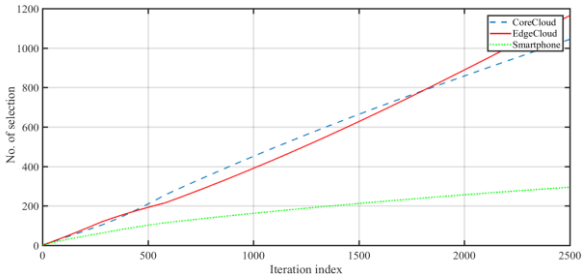
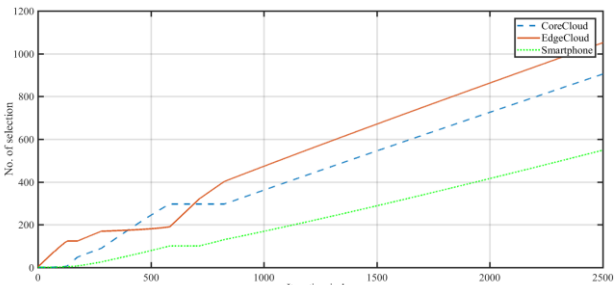


Figure 3. Variation of computation time $T_{edgeserver}$ for face recognition in the remote cloud.

Next, regret is used to compare the performance of UCB-1, random selection, and D-UCB. Regret analysis is defined as the expectation of the difference between the total reward obtained by selecting the optimal device and the total reward obtained by selecting the device using the algorithm. Fig. 5 shows the regret analysis between USB-1, random selection, and D-USB. The cumulative regret of D-USB rises slowly compared with USB-1 and random selection.



a. UCB-1



b. D-UCB

Figure 4. Selection of edge cloud as computation device changes over time due to the variation of edge server computation time $T_{edgeserver}$.

TABLE II. LISTS OF SIMULATION PARAMETERS

Target device	Parameters
Smartphone 1.4 GHz Quad-core Cortex-A53 RAM: 2 GB	Processing time = 0 to 2.8 second Energy consumption of the smartphone for local processing = 8.736 mAh per face image Monetary cost = 0 $ per second
Edge Cloud (UIU datacenter) Intel Xeon Processor E5-2630 V3 4 core @ 2.40 GHz, RAM: 8 GB	Processing time = 0 to 0.6 second Energy consumption of the smartphone for remote processing = 1 mAh per face image Monetary cost = 0.000046 $ per second
CoreCloud (t2.xlarge instance) Intel Xeon Processor E5-2686 V4 4 core @ 2.30 GHz, RAM: 16 GB	Processing time = 0 to 0.55 second Energy consumption of the smartphone for remote processing = 1 mAh per face image Monetary cost = 0.0000555$ per second
Network	
Smartphone to Edge Cloud	Bandwidth = 12 mbps Edge RTT = 0.0157 millisecond Hop count = 13
Smartphone to Core Cloud	Bandwidth = 10 mbps Core RTT = 0.087 millisecond Hop count = 27

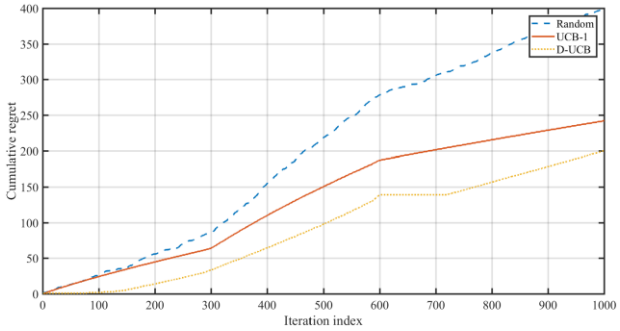
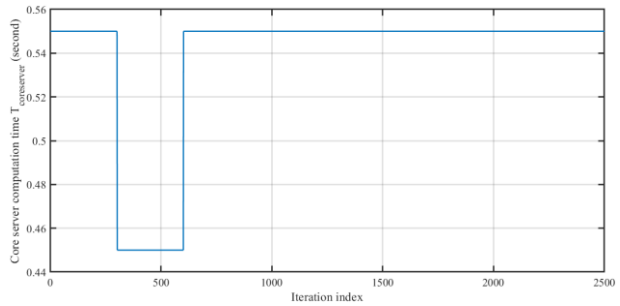
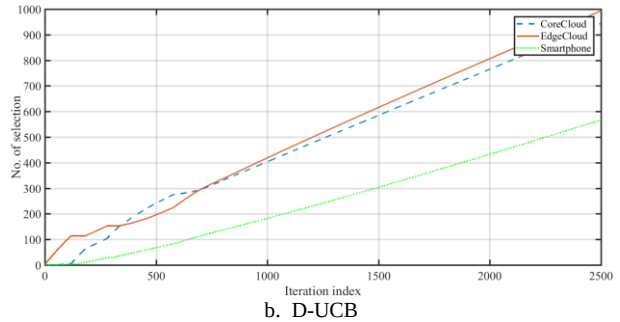


Figure 5. Regret analysis between UCB-1, random selection and D-UCB.



a. Commutation time $T_{Coreserver}$ variation for face recognition in the remote cloud



b. D-UCB

Figure 6. Selection of core cloud as the processing device changes over variation of core server computation time $T_{coreserver}$.

In the second experiment, the behavior of D-UCB is tested by investigating its performance when the computation time of the Core Cloud server $T_{coreserver}$ varies (Fig. 6). Fig. 6a shows that Core Cloud is selected as a computation device at iteration = 340 due to the variation of Core server computation time at the first breakpoint (iteration = 300) (Fig. 6b). The selection changes back to Edge Cloud at iteration = 690 due to the second breakpoint (iteration = 600) of Core server computation time. In the third experiment, the behavior of D-UCB is tested by investigating its performance by varying smartphone processing time, the energy consumption of the smartphone for local processing, smartphone to Edge Cloud network bandwidth, the monetary cost for Edge Cloud processing, smartphone to Core Cloud network bandwidth, the monetary cost for Core Cloud processing. Fig. 7 shows that the smartphone as a commutation processing device is selected at iteration = 390 due to the variation of Core server computation time (Table III) at the first breakpoint (iteration = 300). The selection changes back to Edge Cloud at iteration = 910 due to the second breakpoint (iteration = 600).

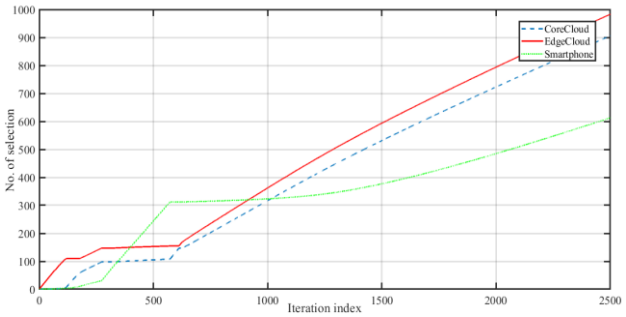


Figure 7. Selection of smartphone as processing device changes over time due to the variation of Table III parameters.

TABLE III. CHANGE OF SIMULATION PARAMETERS FROM ITERATION 300 TO 600

Parameter
Smartphone processing time = 2.8 second
Smartphone energy consumption = 9 second
Smartphone to Edge Cloud network bandwidth = 12000000 mbps
Monetary cost for Edge Cloud processing = 0.000046$
Smartphone to Core Cloud network bandwidth = 10000000 mbps
Monetary cost for Core Cloud processing = 0.000055$

© 2022 J. Adv. Inf. Technol.

V. Conclusion

In this paper, we have proposed an offloading framework exploiting trade-offs between processing time and energy saving, as well as considering economic factors. This framework can adapt its decision in dynamic network and cloud server load environments with reduced computation time. In future work, we will expand the capability of the proposed framework for video streaming with transcoding, with extensions towards Virtual Reality/Augmented Reality (VR/AR) applications.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Nasif Muslim conducted the research, collected data and wrote the paper. Salekul Islam and Jean-Charles Grégoire supervised the work and approved the final version.

Acknowledgment

This work was supported by a grant, UIU-RG-161011 from the Institute for Advanced Research (IAR), United International University (UIU).

References

[1] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51-56, 2010.

[2] N. Muslim, S. Islam, and J. C. Gregoire, "Offloading framework for computation service in the edge cloud and core cloud: A case study for face recognition," *International Journal of Network Management*, p. e2146, 2020.

[3] W. Mei and W. Deng, "Deep face recognition: A survey," arXiv preprint arXiv: 1804.06655, 2018.

[4] X. Xia, C. Xu, and B. Nan, "Inception-v3 for flower classification," in *Proc. 2nd International Conference on Image, Vision and Computing*, 2017, pp. 783-787.

[5] T. Y. H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proc. the 13th ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 155-168.

[6] T. Ahonen, A. Hadid, and M. Pietikainen, "Face recognition with local binary patterns," in *Proc. European Conference on Computer Vision*, 2004, pp. 469-481.

[7] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[8] Q. K. Gill and K. Kaur, "A review on energy efficient computation offloading frameworks for mobile cloud computing," *International Journal of Innovations & Advancement in Computer Science*, vol. 5, p. 1, 2016.

[9] M. H. Chen, B. Liang, and M. Dong, "A semidefinite relaxation approach to mobile cloud offloading with computing access point," in *Proc. IEEE 16th International Workshop on Signal Processing Advances in Wireless Communication*, 2015, pp. 186-190.

[10] S. Islam and J. C. Gregoire, "Network edge intelligence for the emerging next-generation internet," *Future Internet*, vol. 2, no. 4, pp. 603-623, 2010.

[11] D. Chemodanov, C. Qu, O. Opeoluwa, S. Wang, and P. Calyam, "Policy-based function-centric computation offloading for real-time drone video analytics," in *Proc. IEEE International Symposium on Local and Metropolitan Area Networks*, 2019, pp. 1-6.

[12] S. Chinchali, *et al.*, "Network offloading policies for cloud robotics: A learning-based approach," arXiv preprint arXiv:1902.05703, 2019.

[13] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "An autonomous computation offloading strategy in mobile edge computing: A deep learning-based hybrid approach," *Journal of Network and Computer Applications*, vol. 178, p. 102974, 2021.

[14] A. Shahidinejad and M. Ghobaei-Arani, "Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach," *Software: Practice and Experience*, vol. 50, no. 12, pp. 2212-2230, 2020.

[15] A. Shahidinejad, F. Farahbakhsh, M. Ghobaei-Arani, M. H. Malik, and T. Anwar, "Context-aware multi-user offloading in mobile edge computing: A federated learning-based approach," *Journal of Grid Computing*, vol. 19, no. 2, pp. 1-23, 2021.

[16] S. Xiao and X. l. Wang, "The method based on q-learning path planning in migrating workflow," in *Proc. International Conference on Mechatronic Sciences, Electric Engineering and Computer*, 2013, pp. 2204-2208.

[17] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access*, vol. 5, pp. 11255-11268, 2017.

[18] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[19] A. Moustafa and M. Zhang, "Multi-objective service composition using reinforcement learning," in *Proc. International Conference on Service-Oriented Computing*, 2013, pp. 298-312.

[20] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235-256, 2002.

[21] C. Hartland, S. Gelly, N. Baskiotis, O. Teytaud, and M. Sebag. (2016). Multi-armed bandit, dynamic environments and meta-bandits. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00113668

[22] L. Kocsis and C. Szepesvari, "Discounted UCB," in *Proc. 2nd PASCAL Challenges Workshop*, 2006.

[23] S. Chakraborty and C. H. Yeh, "A simulation based comparative study of normalization procedures in multiattribute decision making," in *Proc. the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases*, 2007, pp. 102-109.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818-2826.

[25] M. Abadi, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," arXiv preprint arXiv:1603.04467, 2016.

[26] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248-255.

Copyright © 2022 by the authors. This is an open access article distributed under the Creative Commons Attribution License (CC BY-NC-ND 4.0), which permits use, distribution and reproduction in any medium, provided that the article is properly cited, the use is non-commercial and no modifications or adaptations are made.

Nasif Muslim received the B.Sc. degree in Computer Engineering from American International University-Bangladesh (AIUB), Dhaka, Bangladesh, in 2007, and the M.Sc. degree in Communication Technology from University of Ulm, Ulm, Baden Württemberg, Germany, in 2012. He has also completed M.Sc. degree in Computer Science and Engineering from United International University (UIU), Dhaka, Bangladesh in 2019. His current research interest includes Edge Cloud computing, Software Defined Networking, and future Internet architecture.

Salekul Islam is a Professor and Head of the Computer Science and Engineering (CSE) Department of United International University (UIU), Bangladesh. Earlier, from 2008 to 2011, he worked as an FRQNT Postdoctoral Fellow at the Énergie, Matériaux et Télécommunications (EMT) center of Institut national de la recherche scientifique (INRS), Montréal. He completed his PhD in 2008 from Computer Science and Software Engineering Department of Concordia University, Canada. His present research interests are in Cloud computing, virtualization, future Internet, blockchain, SDN and analysis of security protocols. He also carried out research in IP multicast especially in the area of security issues of multicasting.

Jean-Charles Grégoire is a Professor at INRS, a constituent of the Université du Québec with a focus on research and education at the Masters and Ph.D. levels. His research interests cover all aspects of telecommunication systems engineering, including protocols, distributed systems, network design and performance analysis, and more recently, security. He also has made significant contributions in the area of formal method

© 2022 J. Adv. Inf. Technol.