Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 19 No. 18 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijim.v19i18.57233

PAPER

Intelligent Task Prediction and Partial Computation Offloading in Mobile Edge Cloud Computing

Suresh Kumar Jha[1], Bharthala Hema Kumari[2], Ketan Anand[3](✉), Jyoti Kanjalkar[4], Ravindra Babu Gaddam[5], Sumit Kumar[6]

[1]Manipal University Jaipur, Jaipur, Rajasthan, India

[2]Department of Information Technology, Sreenidhi Institute of Science and Technology, Hyderabad, Telangana, India

[3]Sharda School of Computing Science and Engineering, Sharda University, Greater Noida, Uttar Pradesh, India

[4]Vishwakarma Institute of Technology, Pune, Maharashtra, India

[5]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Bowrampet, Hyderabad, Telangana, India

[6]Department of CSE, Shivalik College of Engineering, Dehradun, Uttarakhand

ketan.anand@sharda.ac.in

ABSTRACT

The risk of fraudulent software or apps undermining user privacy is rising for users of cell phones and other portable electronics. Because malicious apps need less permission to run, they are more intrusive than necessary. Due to its open-source nature, support for third-party app stores, and stringent app assessment, the Android platform is more susceptible to assaults. Thus, the Android platform has also led to an increase in the use of portable computing apps. Using edge computing and cloud services, Mobile Edge Cloud Computing (MECC), showing promise in the fractional computation offload approach, has opened up fresh opportunities for mobile apps that are delay-sensitive and computationally demanding. We thoroughly examine the unpredictability method, calculating the arrivals of requests, assistance latency, and variable processing resources to solve this problem. High traffic volumes are produced by many devices that the MEC architecture can manage. First, we give a comprehensive introduction to MCC/MEC technology in this paper, covering the history and development of remote computation techniques. This paper's main body then examines current research regarding the ideas of computing offloading, offloading granularities, and offloading procedures techniques. Furthermore, we go over optimization techniques as well as both static and dynamic offloading mechanisms. Environments. We also go over the difficulties and possible paths for MEC research in the future.

KEYWORDS

mobile edge cloud computing (MECC), computation techniques, Android platform, software

1 INTRODUCTION

The elements influencing the task of transferring computations in mobile edge computing (MEC) and MECC regarding a specific computational activity using components that are separate from one another have been the subject of extensive research in recent years. Many studies [1] have created multiple objective improvement techniques for delegating choices or tactics, taking into account various metrics like task service time, energy intake, collaboration choices, power distribution,

Jha, S. K., Kumari, B. H., Anand, K., Kanjalkar, J., Gaddam, R. B., Kumar, S. (2025). Intelligent Task Prediction and Partial Computation Offloading in Mobile Edge Cloud Computing. *International Journal of Interactive Mobile Technologies (iJIM)*, 19(18), pp. 106–117. https://doi.org/10.3991/ijim.v19i18.57233

Article submitted 2025-05-14. Revision uploaded 2025-06-25. Final acceptance 2025-07-03.

network flow scheduling, and so forth, to provide quality of service (QoS) guarantees in computation offloading. Nevertheless, these approaches typically overlook the interdependencies among several modules in a calculation process, which could significantly affect the execution outcomes. For example, the execution time may not be equal to the total of the individual module execution durations in a computing task where modules must be run in a particular order because of data dependencies. Rather, a delay in one module may have a cascading effect on other modules, increasing the total completion time. The computing challenge of offloading is expressed as a mixed-infinity linear programming issue in contemporary research, with the average execution time serving as the primary basis for the partial offloading choices [2]. To cut down on execution time, this could, however, result in excessive usage of computer resources.

Computational systems must share limited resources, such as memory, bandwidth, and processing power, between several devices or tasks. Thus, effective resource management and allocation are essential to fulfilling the system's requirements. We noted that the Internet of Things (IoT) application is a mission-critical, delay-sensitive signaling application in the context of our work.

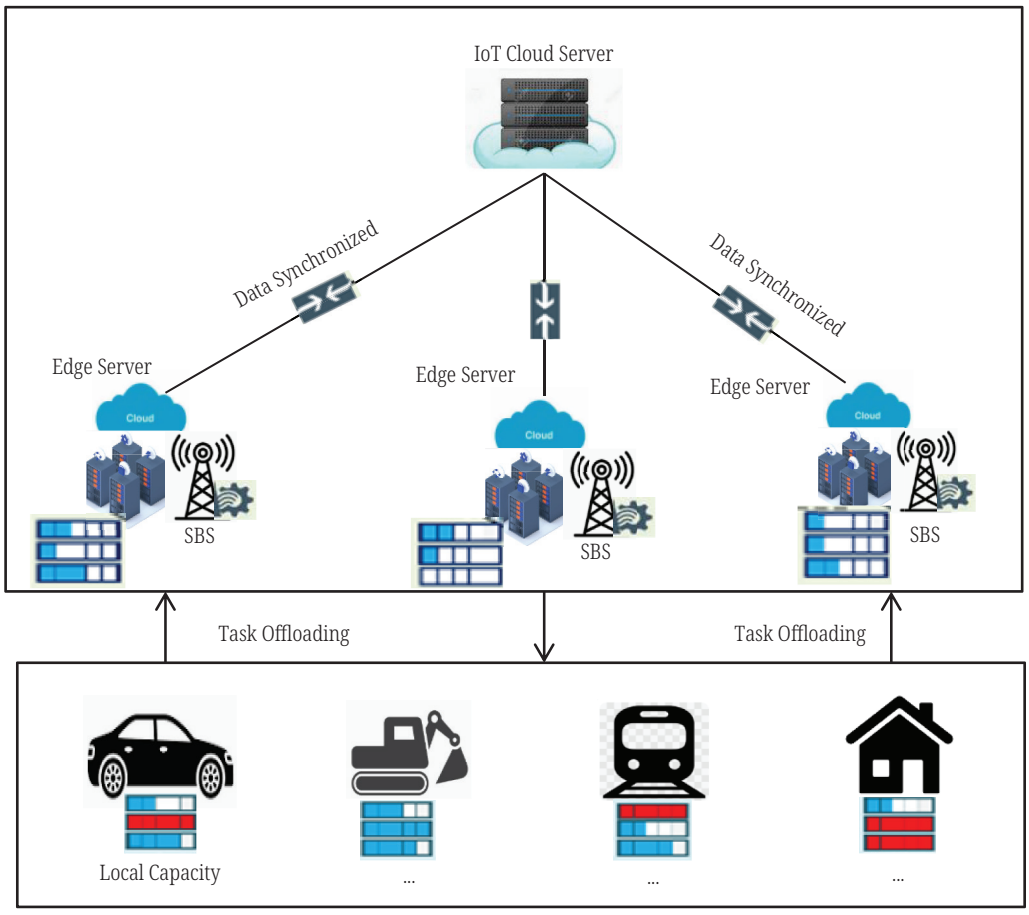

Fig. 1. Model of computation offloading for edge computing on mobile devices

Effective processing requires considering the communication expense related to the task offloading [12]. Minimizing energy and delay usage throughout the communication and processing process is the main problem. Making educated decisions on

task offloading requires accurate power, delay, and expense estimation models that consider the price of energy of interaction by time limitations of IoT applications. In a similar vein, precise latency estimates models aid in choosing the best offloading plan while taking costs, delay, and energy into account [13]. If the energy consumption and overall latency are lower than the time limitations of IoT applications, the objective is to move jobs to the edge server. Our algorithms will decide to process jobs locally if they can do so within the time limitations without taking user device power use into account, provided that the total latency is greater than the time restrictions. This is accomplished by developing an energy, latency, and cost model that allows user devices to precisely determine the amount of energy, latency, and computing expenses incurred when delegating tasks. The goal of this method is to create an offloading technique that considers both energy and latency. To prevent devices used by users from overheating during task computing, our suggested network infrastructure model in Figure 1 shows many user gadgets and several edge servers. Multiple jobs or gadgets must share computational systems' limited resources, which include memory, bandwidth, and processing power. This indicates that satisfying the system's requirements depends on effective allocation of resources and administration. We stated that the IoT application is a mission-critical, delay-sensitive signaling tool in the context of our work. This kind of application includes signaling for important operations when time is of the essence and delays must be kept to a minimum, such as Mission-Critical Push-to-Talk signaling (MC-PTT) or MC video signaling. Through resource optimization for each activity, the suggested method incorporates MEC into the 5G network, which can improve network performance.

The following structure describes how this document is organized. Section 2 explores the body of existing literature that is pertinent to the topic. In Section 3, the problem is defined and the system model's design is described. The model training methodology, the simulation procedures used, and the outcomes analyses are covered in Section 4. Section 5 concludes the report with an outline of the findings.

2 RELATED WORKS

The VM movement technique in a surrogacy cloud server was suggested by the majority of the pioneering publications in mobile cloud computing [3, 4]. And close computing boxes for offloading mobile data for in-depth processing are the cloudlets [5]. A cloudlet is regarded as being within a mobile device's one-hop communication range and has a strong internet connection to the central cloud. To facilitate mobility, the neighboring cloudlets can converse and move virtual machines (VMs) among themselves thanks to mesh networking. Through VM placement, each cloudlet is linked to a central cloud computing on mobile devices to retrieve, store, and analyze the required data. Although the actual hosts' services, mobility cloud, and droplets are situated in firmly geographic places, the states of VMs fluctuate due to random user demands and resource requirements.

ECC offloading strategies have been researched, and a few ECC designs have been suggested for distant carrying out of the assignment [6]. Special offloading and cloud distributing resources policies are needed for ECC. First, when there are several individuals using the ECC system, the ECN resource could not be enough. Second, because of their flexibility, ECNs and SMDs might only be able to stay in the WLAN's coverage area for a short period of time. Lastly, since each ECN has a distinct transmission rate and processing control, an SMD must select the right

ECN to offload. Nevertheless, the impact of mobility and restricted computational resources on offloading technique is not well considered by the current ECC architectures and offloading methods. Our work on this study is motivated by this. Even though latency-sensitive applications could benefit from ECN services, creating an affordable ECC system is still difficult. In this study, we concentrate on the optimal multi-user dumping problem in an ECC system to make possible the potential benefits of ECC in terms of energy savings and performance enhancement for SMDs.

These services have extremely low latency offered via computing on the edge and can be utilized by a control aircraft equipped with multiple controller designs at the outermost layer in large-scale networks to promptly handle network requests. Nevertheless, dynamic network traffic demands cannot be met by a fixed control plane. Although the increased mobility of mobile devices results in constant changes to the network architecture, this issue can be effectively resolved by dynamically modifying controller deployment based on the network state. Controller state synchronization overhead can be decreased by implementing the control plane as an ordered framework [7]. The control plane's performance has a significant impact on an SDN's performance. Network burdens in a cloud–edge cooperation system are not uniformly distributed because network traffic fluctuates over time. Deploying a robust, load-balanced, and low-latency controller plane is therefore crucial for SDN-enabled cloud-edge collaboration platforms.

Edge computing was suggested as an answer to this problem [14]. A computer paradigm known as "edge computing" places system, storing, as well as computational capacity near to the user's position. Cloudlets, fog analysis, and multi-access edge computing, originally known as mobile edge computing, are some of the instances of edge computing that have surfaced in recent years [8]. MEC, as established by the Multi-access Edge Computing Industrial Specification Group (MEC ISG) of the European Telecommunications Standards Institute (ETSI) in September 2014, has emerged as the standard for edge computing in 5G, specifically in the context of mobile communication networks [15, 16]. By deploying services that manage compute-intensive operations at a network's edge, MEC makes it easier for UEs to optimize their resources. The primary goal is to provide IT services and resources close to the UE in the Radio Having access Network.

There are delays between endpoint gadgets and the central cloud because of the amount of information produced and handled by IoT gadgets. Fog is a new method for reducing delays [9] and making appropriate use of IoT devices. In 2014, Cisco launched fog computing, which processes data in real time and executes it at the closest server. Fog computing is now being used by educational organizations to process data. Instead of sending all of this information to the cloud, the institutions process it in conjunction with major cloud multinationals and store it in their local caches. These caches are thought of as a fog that makes communication between cloud central and learning edge devices easier. Applications for mobile learning that use a fog-cloud architecture run on fog nodes, which are learners' smartphones. These kinds of architectures are important for carrying out and adaptive educational apps in accordance with learners' execution requirements. These architectures are also effective for embedded applications based on AI, reactive mobile learning, and providing actors with services like low latency, context awareness, reliability, and quality of service. They can also be integrated with 5G.

Information has changed from basic words and images to more intricate music and video as multimedia technology continues to advance [10]. Digital video is a key component of information processing and is used extensively in communication.

The scene can be effectively restored using audio and video files. Eighty percent of the information-gathering duty is completed by case vision, which can also directly reflect environmental changes. The quick development of information technology since the start of the digital era has led to the widespread adoption of digital video technology. There has been widespread use of digital video transmission and retention.

Numerous of these methods concentrate on optimizing with many objectives, making selections about unloading by taking into account variables like time spent on activities, energy usage, allotment of power, cooperation selection, and other important measures for achievement. Although these techniques are good at preserving QoS, their inflexible frameworks and the elevated computing complexities required to solve the optimization issues frequently make them difficult to adapt and scale in dynamic contexts [11]. Digital video plays at a specific frame rate and is made up of a series of frames. From one application to another, there are significant differences in video characteristics, including frame rate, image quality, and pixel depth. As a result, it has enormous importance and practical value for the creation, use, and dissemination of video data, and it will eventually become a trend in growth. Although video information is ubiquitous, precise, successful, and easy to understand, there is too much of it.

3 METHODS AND MATERIALS

The allocation of resources and offloading of computation have drawn the interest of numerous scholars in recent years. Two criteria are typically used to evaluate the efficiency of offloading: 1) execution latency and 2) resource assumption.

A) **Full Offloading:** An approach to one-dimensional searching that takes the buffers for applications into account, standing state, and available processor power is proposed by the authors in order to reduce the implementation delay. Energy harvesting techniques, along with dynamic voltage and frequency scaling, are used to maximize data transfer for offloading computations; it can furthermore lessen the likelihood that unloading apps would fail. To reduce the average reaction time for offloading, the authors build three-layer traffic architecture. The edge nodes based on moving vehicles are modeled using queuing theory. To improve Quality of Experience (QoE), the authors collaboratively optimize scheduling and offloading. The energy usage can be significantly decreased by using the median execution delay constraint. But the majority of recent research only looks at the scenario with a single user.

   Two groups are created from each UE based on how much data they must discharge. While the UEs in the second group are limited to local task processing, the UEs in the first grouping have access to the MEC server. The distribution of computational and communication resources can be used to calculate the ideal transmission power. The authors concentrate on using cooperative computing at the edge in wireless mesh networks to decrease the saturated backhaul capacity.

B) **Partial Offloading:** The authors use linear programming to solve the issue of incomplete unloading, which they frame as a non-linear restriction problem. Organizing the delay of application restrictions in a Time Division Multiple Access (TDMA) integrated framework results in an efficient resource allocation approach. The authors offer a partial offloading strategy that maximizes energy

savings based on the maximum application latency limit. To determine the ideal transmission rate in uplink, an iterative approach is suggested. By considering using a buffer, the authors formulate the challenge of minimizing energy use. The best CPU frequency, transmission job, and bandwidth allocation are determined by an online method based on Lyapunov optimization. The challenge of resource block scheduling in narrowband IoT is formulated. Power regulation and relay choice are both taken into account by the heuristic algorithm.

C) **Collaboration of MCC and MEC:** Compared to one individual problem, issues with multi-user offloaded computation are more prevalent and intricate. A greedy heuristic algorithm solves the calculation by multiple users' issue with partition, which is phrased as a MILP problem. This work's flaw is that it doesn't account for user interference or collaboration between MCC and MEC. Users' interference is taken into account by the authors. Nevertheless, it ignores MCC and MEC cooperation in favor of full offloading.

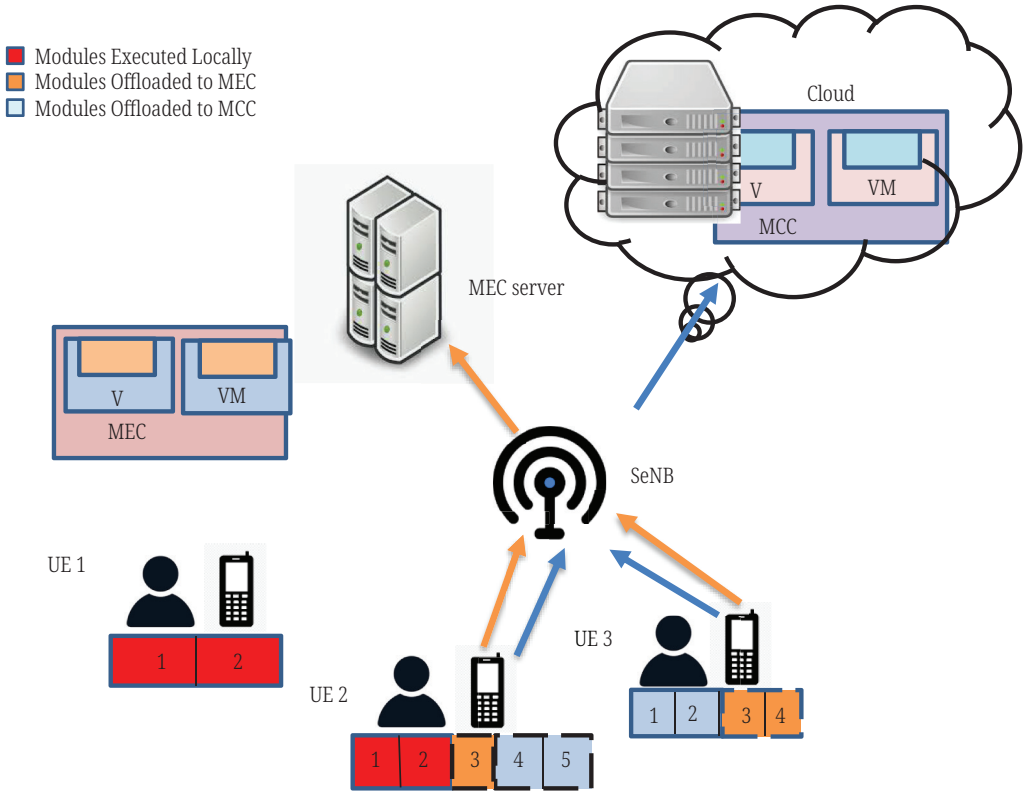

Fig. 2. A partial computation offloading example

In contrast to the aforementioned studies, this work fully examines the collaboration between MCC and MEC, whereby IoT-based UEs can choose to offload their duties via MEC or MCC based on completion delay.

3.1 Challenges for computation offloading in edge computing

Edge computing faces that are offloaded by computation numerous obstacles, including task allocation, job execution, and application segmentation. We look into the study that has been done on these issues in this part. Specifically, we will

highlight the distinct challenges in edge computing as opposed to MCC for each problem. Figure 2 provides an overview of these difficulties.

A) **Application Partitioning:** The initial phase of compute offloading is division, which separates the program enter the code multiple sections that will run on various systems, such as cloudlets, mobile devices, or the cloud, or the cloud. It has been thoroughly examined in MCC. Nonetheless, it displays a few novel edge computing features.

B) **Partitioning in MCC:** Applications can be partitioned using two general methods: 1) programmer-specified partitioning and 2) automatic program analysis. In the first, the movement of data in the program code is examined, and possible off-loadable components are examined, using dynamic or static program analysis techniques. In actuality, the most popular model for representing the connections between tasks, objects, procedures, or components in an application analysis is a graph-based model. The aforementioned entities are represented by the vertices of such an interaction graph, while the masses of the borders provide the expenses of interactions (such as data volumes, transmission durations, and bandwidth). Dividing a graph technique is then used to trim the diagram by accomplishing certain objectives, such as optimizing throughput, lowering network traffic, saving energy, or shortening execution time.

This technique was created in 1999 by Coign, one of the first offloading mechanisms. Coign creates a graph model of the application's inter-component interaction based on program behavior profiling. Coign then chooses a partition to reduce communication time using a graph-cutting algorithm called lift-to-front minimum-cut. The purpose of Coign, the initial system designed to validate the idea of autonomous program analysis, is to reduce the effort required of programmers. Figure 3 represents the challenges in computation offloading.

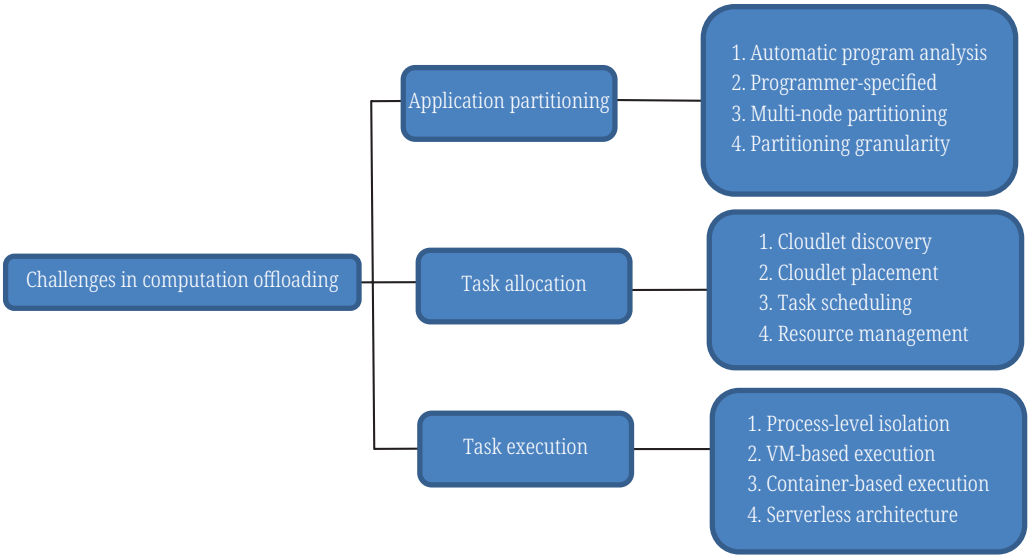

Fig. 3. Immediate of challenges for computation offloading toward edge computing

C) **Partitioning in Edge Computing:** Considering that the partitioning methods listed above are readily available, what is novel about edge computing-based

compute offloading? Partitioning for multi-node offloading is a hurdle. Runtime decisions define the precise locations tasks in MCC should be placed on, either local (mobile devices) or remote servers (the cloud). But in edge computing, work can be divided among servers throughout the internet link that connects handheld gadgets to the cloud, including several cloudlets. What partitioning scheme is appropriate in this situation?

A partitioning technique was created by Sinha and Kulkarni to carry out multi-node offloading, which distributes an application's tasks over several distant servers. Applications that focus on data are the driving force behind this study. For instance, Flick and Facebook are common applications that carry out image matching and photo recognition across several websites. The program's behavior is represented by an object interaction graph, and communication expenses are reduced via a heuristic graph-cutting technique.

The dividing granularity presents another difficulty. Many MCC works carry out compute offloading at the class, thread, and method levels. At these levels, task execution necessitates consistent runtime settings, which entails both context dependency and data dependence. Such a need is costly, though, given the heterogeneous infrastructures found in edge computing. As a result, letting go of the task or level of components would increase productivity. Programmers are able to create several approaches to task/component execution depending on the platform.

This technique was created in 1999 by Coign, one of the earliest offloading systems. Using software behavior profiling, Coign builds a graph model of the application's inter-component interactions. Coign then chooses a partition to reduce communication time using a graph-cutting algorithm called lift-to-front minimum-cut. The purpose of Coign, the initial system designed to validate the idea of autonomous program analysis, is to reduce the effort required of programmers.

4 IMPLEMENTATION AND EXPERIMENTAL RESULTS

The following is the configuration of the algorithm parameters: The set of data used for training of the LSTM module consists of 1500 computation logs being offloaded on edge cloud nodes, whereas the testing set of data consists of 250 logs of offloaded computations. There are 1000 iterations and four layers that are hidden. There is a 50-batch size and a 0.025 convergence loss. Data size V for the computing job and data size φ for its subtasks will vary linearly as experimental variables, allowing the task to test the overall duty that delays processes following the implementation of different algorithms.

For comparative tests, two methodologies are chosen to evaluate the calculation offloading methodology based on task prediction: 1) The calculation task will be carried out immediately on mobile devices. There is no data transfer delay in this mode; instead, the task's overall latency is mostly caused by computation delay. 2) Every computing work must be transferred from mobile devices to edge computing nodes that are connected for execution. Transmission delay, computing delay, queuing delay, and other delays are all included in the task's overall delay in this manner.

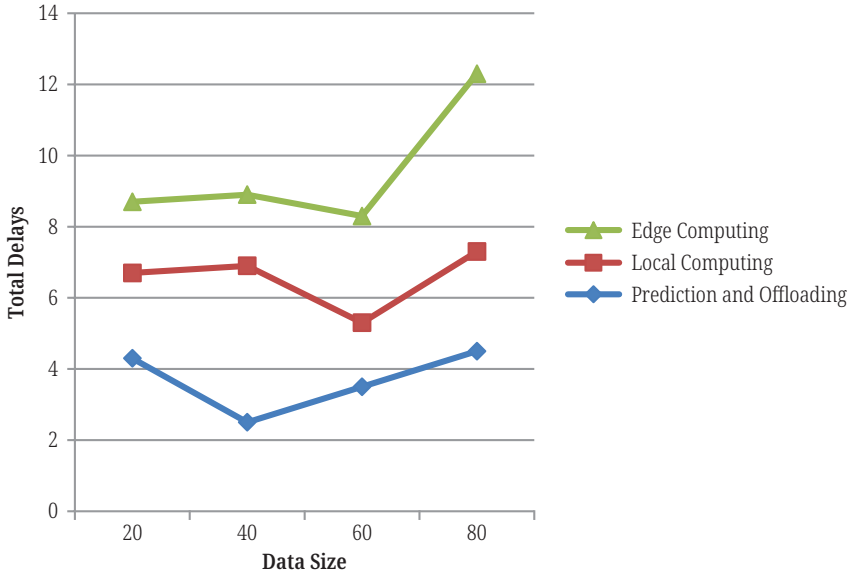

Fig. 4. Total computing job delays for varying data sizes

Figure 4 shows how the overall job latency changes as the volume of data increases gradually to fewer than three different methods for unloading. The volume of data of task V falls between the {40, 50, ..., 120}. It is evident that, considering the condition of our local technology, mobile devices' processing capability is inadequate to manage duties including massive quantities of information. Consequently, local processing is faster when the data amount is lower. Local processing power will grow in a nonlinear manner as data size increases; it makes the services difficult that depend on delays. The total delay optimization is hampered by tiny data sizes due to the delay in transmission in the edge computing offloading mode. However, the benefit of edge computing nodes' increased computing capacity becomes apparent as data sizes increase. In our technique, local computing, edge computing, and subtask migration can be incorporated to take into account the subtask forms of compute tasks. To maximize the overall task delay, it is possible to identify an ideal compute offloading approach for jobs with varying data volumes.

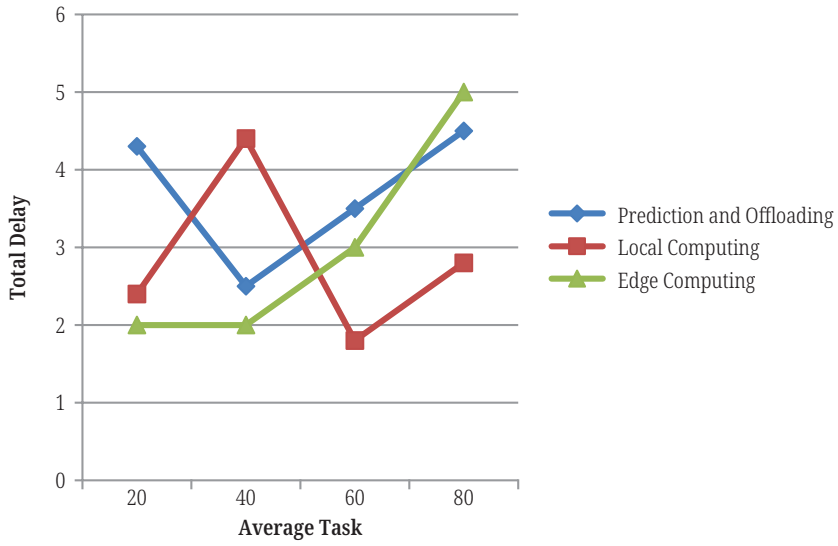

Fig. 5. Total computation task latency for varying numbers of subtasks

Figure 5 shows how the overall task delay varies as the quantity of smaller assignments for every calculation assignment increases gradually beneath various offloading techniques. Consequently, $\varphi \in \{0, 1, ..., 10\}$, the computational complexity rises with the quantity of subtasks. Mobile gadgets have a limited amount of local computational capacity. As the number of subtasks increases, the local task delay is going to be significantly greater than that of two more compute offloading schemes. Currently, a part of compute latency can be decreased by surrendering transmission delay using our subtask mitigation technique.

5 CONCLUSION

Allocating computational resources during the process of offloading compute activities might be difficult due to the dynamic character of MECC systems. In order to satisfy delay requirements without using excessive computational resources, we present DRL architecture in this paper. First, we use a two-way queue structure, which consists between two operating queues with many servers that are accessible in each, to mimic the procedure for offloading partial computations from smartphones to cloud gateways and edge devices.

This paper proposes an innovative, clever offloading of computation founded on MEC design with an amalgamation of three different styles with reference to the drawbacks of the three conventional forms of computing: local, edge, and cloud. The advantages of the current generation of MEC and its research goals are also covered. The task prediction-based computation offloading and task migration algorithm is created using the suggested architecture. The LSTM-based computation task prediction method, the task prediction-based computing offloading strategy for handheld devices, and the computing task migration for the edge cloud scheduling scheme are utilized to provide a detailed description of the ideal computation offloading approach. Performance tests are carried out using our suggested algorithm and architecture. Our technique may efficiently lower the overall assignment latency when the amount of computation information and subtasks increases, ensuring efficient processing of time-sensitive jobs when compared to desktop and single computing edge offloading strategies. The final section discusses the future areas of study from three perspectives: security and privacy, MEC administration based on smart mobility prediction, and combined optimizing of 3C depending on task prediction.

6 REFERENCES

[1] L. Yu, H. Xu, Y. Zeng, and J. Deng, "Delay-aware resource allocation for partial computation offloading in mobile edge cloud computing," *Pervasive and Mobile Computing*, vol. 105, p. 101996, 2024. https://doi.org/10.1016/j.pmcj.2024.101996

[2] C. Eang *et al.*, "Offloading decision and resource allocation in mobile edge computing for cost and latency efficiencies in real-time IoT," *Electronics*, vol. 13, no. 7, p. 1218, 2024. https://doi.org/10.3390/electronics13071218

[3] D. Uma Nandhini, S. Udhayakumar, L. Tamilselvan, and J. Silviya Nancy, "Client aware scalable cloudlet to augment edge computing with mobile cloud migration service," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 14, no. 12, pp. 165–178, 2020. https://doi.org/10.3991/ijim.v14i12.14407

[4] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[5] M. G. R. Alam, M. M. Hassan, M. Z. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for IoT applications," *Future Generation Computer Systems*, vol. 90, pp. 149–157, 2019. https://doi.org/10.1016/j.future.2018.07.050

[6] 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, 2015. https://doi.org/10.1109/TNET.2015.2487344

[7] J. Liu, S. Guo, Q. Wang, C. Pan, and L. Yang, "Optimal multi-user offloading with resources allocation in mobile edge cloud computing," *Computer Networks*, vol. 221, p. 109522, 2023. https://doi.org/10.1016/j.comnet.2022.109522

[8] C. Xu, C. Xu, B. Li, S. Li, and T. Li, "Load-aware dynamic controller placement based on deep reinforcement learning in SDN-enabled mobile cloud-edge computing networks," *Computer Networks*, vol. 234, p. 109900, 2023. https://doi.org/10.1016/j.comnet.2023.109900

[9] K. Zhang *et al.*, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016. https://doi.org/10.1109/ACCESS.2016.2597169

[10] M. Y. Akhlaqi and Z. B. M. Hanapi, "Task offloading paradigm in mobile edge computing-current issues, adopted approaches, and future directions," *Journal of Network and Computer Applications,* vol. 212, p. 103568, 2023. https://doi.org/10.1016/j.jnca.2022.103568

[11] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, "Efficient resource allocation for on-demand mobile-edge cloud computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8769–8780, 2018. https://doi.org/10.1109/TVT.2018.2846232

[12] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2018. https://doi.org/10.1109/JIOT.2018.2868616

[13] T. Q. Dinh, Q. D. La, T. Q. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6353–6367, 2018. https://doi.org/10.1109/TCOMM.2018.2866572

[14] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Computer Networks*, vol. 182, p. 107496, 2020. https://doi.org/10.1016/j.comnet.2020.107496

[15] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019. https://doi.org/10.1109/ACCESS.2019.2938660

[16] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective," *Software: Practice and Experience*, vol. 50, no. 9, pp. 1719–1759, 2020. https://doi.org/10.1002/spe.2839

7 AUTHORS

Dr. Suresh Kumar Jha is working as Assistant Professor (Sr. Scale) in the department of Computer and Communication Engineering at Manipal University, Jaipur. He received his B. E (CSE) from Government Engineering College, Kota, and his MTech (CSE) from RTU, Kota. He has completed his Ph.D. from MBM University,

Jodhpur, Rajasthan. His area of research is multi-agent networks, consensus theory, graph theory, and distributed algorithms. He has 9 years of academics as well as 3 years of IT industry experience as a technology trainer and technical lead. He teaches courses such as, e.g., Algorithms, Data Structures, Operating Systems, Theory of Computation, etc. (E-mail: suresh.jha@jaipur.manipal.edu).

Bharthala Hema Kumari is working as an Assistant Professor in the Department of Information Technology at Sreenidhi Institute of Science and Technology, Ghatkesar, Hyderabad, India. She received her BTech and MTech from JNTU Anantapur, India. Her area of interest includes Machine Learning, Deep Learning, Cloud Computing, Data Mining (E-mail: hemakumari.b@sreenidhi.edu.in).

Ketan Anand, A Seasoned, forward-looking academician, has a total of 10 years of experience in teaching and training the students for UG and PG in competitive programming and sophistications of AI. He did BTech (IT), MTech (Computer Science major in AI and ML) at West Bengal University of Technology and Central University in 2012 and 2015, respectively. Currently pursuing a PhD from the National Institute of Technology Jalandhar. His area of research is AI, ML, and Medical Imaging (E-mail: ketan.anand@sharda.ac.in).

Dr. Jyoti Kanjalkar is working as Assistant Professor in the Department of Computing Science and Engineering (AI & ML), Vishwakarma Institute of Technology, Pune (Maharashtra). She has 26 years of teaching experience. She holds a Ph.D. degree in Computer Science and Engineering from KLEF, Vijayawada, A.P., India. Her research area includes machine learning, deep learning, image processing, and high performance computing (E-mail: jyoti.kanjalkar@vit.edu).

Ravindra Babu Gaddam is working as an Assistant professor in the Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Bowrampet, Hyderabad Telangana, India, He Received his B.tech and M.tech from JNTU Hyderabad, India, His area of interest includes Cloud Computing, Machine Learning, Deep Learning (E-mail: ravindrababu@klh.edu.in).

Dr. Sumit Kumar is currently a Professor and Head of the Department of CSE at Shivalik College of Engineering, Dehradun. He was awarded with Doctorate Degree in the year 2019 in the field of Computer Science and Engineering Major for his research work “Prediction of student’s performance using K-Star Algorithm.” He obtained his M. Tech in CSE degree in the year 2012 from BPUT, Rourkela, Odisha. He obtained his B. Tech in CSE in the year 2006 from KIIT Deemed University, Bhubaneswar. He has more than 14 years of long teaching experience, and he taught various major subjects of Computer Science and Engineering during this tenure. His academic acumen is reflected in his prolific publications – comprising 25 research papers (SCI/SCOPUS) published in reputed international and national journals and conferences. In essence, he embodies excellence in AI & ML education, research, and institutional leadership (E-mail: headcse@sce.org.in).