Journal of Theoretical and Applied Information Technology
15th August 2023. Vol.101. No 15
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENHANCED COLLABORATIVE TASK EXECUTION IN EDGE COMPUTING BASED ON BELIEF RULE GEO CLUSTERING AND PARTICLE SWARM JOINT OPTIMIZATION

UDAYAKUMAR K[1], RAMAMOORTHY S[2*]
[1]Research Scholar, Department of Computing Technologies
[2]Associate Professor, Department of Computing Technologies
[1,2] SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu-603203, India
Email:[1]uk7447@srmist.edu.in, [2*]ramamoos@srmist.edu.in

ABSTRACT

Multi-access Edge Computing (MEC) has recently been suggested as an addition to cloud computing. MEC servers are placed close to the network's edge to reduce latency and relieve demand on cloud data centres. The cloud is more resourceful than the MEC server, which has less resource. Each MEC server cannot satisfy all computational and large data needs from user devices when operating independently. Collaborative edge computing (CEC) is a popular new concept in which devices on the edge work together by sharing data and computer resources. CEC must determine when and where each task is executed, making task offloading an important problem to address. However, it is challenging to solve task offloading in CEC because tasks can be offloaded to neighbouring devices, resulting in bandwidth contention among network flows. Most existing works do not jointly consider network flow scheduling, which can result in network congestion and inefficient completion time performance. This paper suggests a unique approach for planning, executing, and clustering MEC servers in a resource allocation model. Belief rule Geo clustering method used to cluster the MEC server based on real-time data of intensive tasks. For the simple sake, real-time monitoring application based on video surveillance is taken. The unsupervised cluster algorithm used to distribute the software components optimally among mobile devices and reduce cluster traffic in relation to the data center. The distributed MEC channel task allocation model is used to share tasks when the MEC server is busy receiving data, and a particle swarm-based joint optimization architecture is used to optimize clustered data. Extensive experiments have been performed on three datasets concerning energy utilization, delay, computational burden, throughput, and network Quality of service (QoS). The proposed technique achieved better results than benchmark solutions, which do not make a joint decision.

Keywords: *Mobile Edge Computing, Collaborative Tasks Scheduling, Resource Allocation, Geo Clustering, Particle Swarm Optimization*

1. INTRODUCTION

Traditional specialized systems are transforming due to the Industrial Internet of Things (IIoT). The IIoT is the way to go today due to the proliferation of high-tech devices (adaptable machines, sensors) and the diverse range of required uses. By 2025, it is anticipated that 31 billion IoT devices with varying capacities would have been placed to carry out operations that need varying QoS standards. The IIoT consists of many industrial devices linked together through various communication channels [1]. It helps firms to predict the occurrence of problems, improve the amount of automation used, and speed up manufacturing. However, because these applications usually consume a large amount of data and processing, they will rapidly consume the limited computational capacity and energy of wireless IIoT devices. New business models are being developed that place emphasis on both MEC as well as Wireless power transfer methods in order to meet new needs as well as make the most of the IIoT's potential [2].

Additionally, it is anticipated that mobile data traffic will continue to double annually. Network operators must put in a lot of effort to improve the user experience while maintaining a healthy revenue growth rate to meet these growing demands. The two new paradigms have been proposed as a means of overcoming the drawbacks of the present RANs: i) Cloud Radio Access Network (C-RAN), which proposes to empower network edge through virtualization, and ii) MEC, which aims to centralize Base Station (BS) functions. Even though two methods propose to move computing capabilities in two distinct directions cloud and edge they complement one another, each playing a distinct role in the 5G ecosystem [3]. These kinds of applications typically use a lot of computation, are sensitive to delays, and use a lot of energy. Nonetheless, because of the restricted battery duration and computational limit of a mobile device (MD), it is frequently and truly challenging for MD to meet necessities as well as the Nature of Involvement of these portable applications.

Applications like real-time video processing, autonomous automobiles, AR/VR etc., have been developed as a result of the growth of IoT devices with increased computing, communication, and storage capacity. These applications have essential needs that centralized cloud computing cannot satisfy, such as real-time processing, mobility, context awareness, etc. By bringing computing into the network adjacent to data sources, MEC tries to address the demands of these applications. Each MEC server cannot satisfy all computational and large data needs from user devices when operating independently. Collaborative edge computing (CEC) is a popular new concept in which devices on the edge work together by sharing data and computer resources. To enable the coexistence of many applications and stakeholders, it is necessary to promote collaboration across various types of devices as applications develop. The idea of collaborative edge computing has lately been presented to help applications with these needs. CEC must determine when and where each task is executed, making task offloading an important problem to address. Making decisions about job sharing is one of the most important parts of CEC. However, it is difficult to solve task offloading in CEC because tasks can be offloaded to neighboring devices, resulting in bandwidth contention among network flows.

MD and MEC servers' limited computation resources and a large number of users will also have an impact on the task execution time. Optimization of either the offloading decision or computation resources have partially solved the issue [4]. Each task can be divided into local and offload tasks in those schemes, with local tasks being processed at the MD and offload tasks being carried out on the MEC servers. In most MEC networks, there are only a limited number of resources for computation and communication. A game-based distributed joint offloading and resource allocation scheme was proposed to help MDs pay less for communication and computation resources by balancing the overhead associated with those resources and the monetary cost of MDs. MIMO, OFDMA, and heterogeneous networks are just a few of the physical layer techniques that have been used to meet the offloading latency while also reducing MD energy and computation resource consumption [5]. Most current methods for offloading computations assume that wireless communications allow for directly transferring computations to MEC servers. However, a mobile device may be unable to directly offload computing tasks to MEC servers because of poor or intermittent connectivity. Signal loss may affect mobile devices' computation offloading performance if computing tasks are forced to offload to MEC servers directly. With the assistance of nearby nodes, a mobile device must offload computing tasks to MEC servers [6].

1.1 Research Motivation

We concentrate on the IIoT environment due to the increasing demand for IoT in industrial sectors. The IIoT network consists of numerous end devices (i.e., IIoT devices) and an edge server. Therefore, the data is collected from the fields or end devices and processed by servers close to the periphery. Due to the heterogeneous IIoT devices in the network, multiple computational operations can be performed concurrently with a limited quantity of computing power and energy. Therefore, offloading their workload to peripheral servers or neighbour nodes may increase task processing speed, decreased latency, and enhanced energy consumption. In addition, the Industry 4.0 revolution necessitates increased process flexibility, enhanced manufacturing quality, and increased revenue. Industry 4.0 includes advanced robotics, the Internet of Things, machine learning, augmented and virtual reality, big data analytics, and cyber security. Industry 4.0 applications requiring high computational capacity and low latency (time-critical) performance include security applications, augmented/virtual reality technology, real-time cyber-physical systems, and autonomous vehicles. Due to the delay in processing, the industry suffers significant monetary and human losses.

For instance, the mining industry is classified as critical according to its economic impact, time, and dangers. Due to infrastructure limitations in communication, data administration, storage, and information exchange, the mining industry has been slow to adopt IIoT. The IIoT focuses on ventilation monitoring, accident analysis, fleet and personnel management, tailing dam monitoring, Autonomous mining equipment, and pre-alarm systems within the mining industry. However, the heterogeneous IIoT devices in the network enable them to perform multiple computational tasks concurrently while consuming limited energy and processing resources locally. It is inadequate to meet the current demand, so edge computing was implemented. The edge-computing paradigm employs a high-pressure burden in the primary network while executing the virtual source with periphery communication between data terminals.

Devices on edge cooperate by sharing data and processing resources under the emerging idea of collaborative edge computing. Promoting collaboration across various types of devices is crucial as apps expand to allow the coexistence of multiple applications and stakeholders. The concept of collaborative edge computing has recently been put out to assist applications with these objectives. Work offloading is a significant issue since CEC must decide when and where each work is carried out. One of the most crucial aspects of CEC is the decision-making process for work sharing. However, task offloading in CEC is a problem since tasks are transferred to nearby devices, causing competition for capacity across network flows.

An intelligent camera-based object detection and human motion (ODTHM) system is one of these IIoT use cases that require high computing resources and low latency for processing data. Object detection and motion monitoring of individuals traveling through the mining field makes it easier to anticipate where an incident may occur. IIoT in the mining industry can go beyond merely monitoring their environments by adopting a proactive stance. Thus, the constant flow of employees on the field, industrial apparatus, products, and machines will operate in an environment that allows unrestricted movement while minimizing unnecessary risks. It is also possible to identify missing safety masks or other security equipment, such as hard helmets, and to send out alerts to prevent accidents. MEC is becoming the principal computing and storage platform for most of today's applications. Edge systems receive numerous daily activities that must be concurrently and effectively mapped to edge resources, including IIoT and Industry 4.0 workloads, big data analytics, and decision-making responsibilities. It prompted us to contemplate the collaborative task execution scenario for resolving intensive and delay-sensitive task processing with multiple objectives. We considered a real-time monitoring application based on video surveillance, a typical IoT application for offloading time-sensitive and intensive tasks. The observed outcomes based on the application provide additional support for the need for the research problem and provide insight into problem formulation.

2. RELATED WORKS

The MEC architecture that incorporates security service assignment, cooperative task offloading, and caching is proposed [7]. For IoT networks with multiple cells, architecture is designed to achieve energy savings and strict security protection. Cited works [8] investigate the issue of dynamic caching, taking into account the tasks' varying popularity over time. However, these works only consider the MEC server's storage capacity and consider that server has sufficient computational capacity to support all offloaded tasks. MEC servers' limited storage, as well as computational capacities, make this assumption unworkable. In Refs, constraints on computation and storage are considered [9]. However, they only offer offline solutions and do not take into account time-varying system dynamics. The authors of [10] combined 3C to develop a novel information-centric heterogeneous network method that makes it possible to compute and cache content in MEC. They thought about virtualized resources, which allow users of various virtual service providers to share computing, communication, and cache resources [11]. In addition, in [12], the authors proposed a framework that uses less energy and takes into account joint networking, caching, and computing to meet the needs of the next generation of green wireless networks. In addition, the fundamental trade-offs between caching, computing, and communication for VR/AR applications were investigated by authors in [13] for MEC applications. Work [14] suggests scheduling policies for single-user tasks that are delay-optimal. However, obtaining the actual transition probability matrix is highly challenging. Author [15] proposes a deep reinforcement learning strategy for the total offloading scheme. However, due to the unsupervised nature of reinforcement learning, global minima may not be guaranteed. A machine learning-based computational offloading strategy is proposed in the work [16]. They demonstrate that their instance-based online offloading scheduler chooses the best scheduling decision. Similarly, in [17], they suggest an online-training ML-based mobile offloading scheduler called MALMOS. Previous research in this area has assumed infinite energy reserves for the mobile user and edge server [18]. Performance of conventional shallow learning techniques [19] and is now utilized in almost every sphere of life [20]. Decision-making applications can gain a lot from deep learning for communication networks, which has yielded outstanding results for runtime scheduling [21]. This is because the network is already taught. Because deep learning methods can learn complex decision boundaries [22] as well as complex data patterns, we are developing one for smart offloading decision-making procedures.

The proposed contribution of this research is as follows:

- To propose novel MEC server clustering and collaborative tasks scheduling and execution in a resource allocation model.
- The MEC server has been clustered using belief rule Geo clustering, where an unsupervised cluster algorithm is used to distribute the software components optimally among mobile devices and reduce the traffic inside the cluster concerning the data center.
- When the MEC server is busy receiving data, the task sharing is carried out using distributed MEC channel task allocation model. The network is optimized using particle swarm-based joint optimization architecture.

3. PROPOSED MODEL

This part examines an original strategy for portable edge processing server clustering and collaborative task scheduling and execution in IIoT application asset distribution model for ongoing video reconnaissance information. The video camera data was used to cluster the MEC server using the belief rule Geo clustering method. The unsupervised cluster algorithm was utilized to optimize the distribution of the software components among mobile devices and minimize cluster traffic in relation to the data center. When the MEC server is busy receiving data, the distributed MEC channel task allocation model is used to share tasks, and a particle swarm-based joint optimization architecture is used to optimize clustered data. This process is given as the proposed distributed MEC architecture is shown in Figure 1.

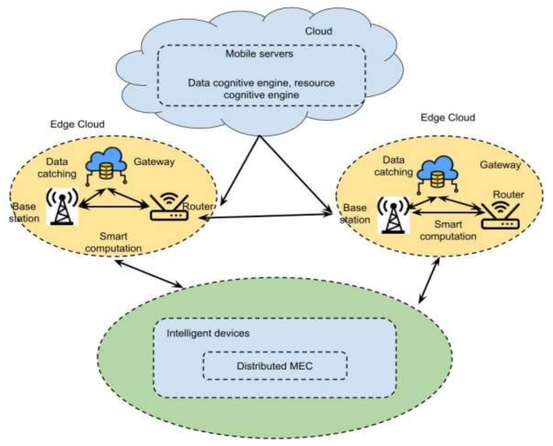


Figure 1. Proposed Distributed MEC Architecture with Edge Servers

We attach at least one request di, which represents amount of data in request, with each vertex (MEC server). As a result, the proposed distributed edge server represented as weighted graph $W_g = (S, A, C)$, where S denotes edge servers, A represents network link between edge servers, and C denotes weight that is aggregate capacity of connecting edges. Thus, the total number of edges defined in the specified range is determined using equation (1).

$$i - 1 \le a \le \frac{i(i-1)}{2} \quad (1)$$

Therefore, the capacity C exists at each edge, resulting in the equation (2).

$$C_a \ge 0, \text{ with } a \in A \quad (2)$$

We correlate a traffic demand, designated di, with each vertex i∈ X. Let ϕ information from vertex as data pass through an edge with k (1 to n) is what we refer to as ϕ in our definition. As a result, eq. (3) is used to characterize the overall flow of data traveling across edge ϕ a:

$$\varphi_a = \sum_{k=1}^{n} \varphi_a^k \le C_a \quad (3)$$

Then, if by eq(4), we consider no waiting in the backlog: current workflow

$$\varphi_a \le C_a \quad (4)$$

Let τ_a represent edge network communication cost:

$$\tau_a = \frac{d_a}{C_a}, \text{ where } d_a \text{ is present data flow in edge } a.$$

3.1 Belief Rule Geo Clustering (BRGC)

Belief rule base is an expanded form of the conventional IF then rule that incorporates belief structure and aids in handling all forms of ambiguity. A Belief Rule is represented by Eq. 5. The BRB is used to generate results, and ER is employed as the inference approach.

$$R_k: \begin{cases} \text{IF } (A_1 \text{ is } V_1^k) \text{ AND / OR } (A_2 \text{ is } V_2^k) \text{ AND / OR} \\ \dots \text{ AND / OR } (A_{T_k} \text{ is } V_{T_k}^k) \\ \text{THEN } (C_1, \beta_{1k}), (C_2, \beta_{2k}), \dots, (C_N, \beta_{Nk}) \end{cases} \quad (5)$$

Where $\beta_{jk} \geq 0, \sum_{j=1}^{N} \beta_{jk} \leq 1$ with rule weight θ_k, and attribute weights $\delta_{k1}, \delta_{k2}, \dots \delta_{kTk}, k \in 1, \dots, L$ where $A_1, A_2, \dots, A_{T_k}$ are antecedent attributes of k^{th} rule. $V_i^k (i = 1, \dots, T_k, k = 1, \dots, L)$ is the referential value of i^{th} antecedent attribute. C_j is j^{th} referential value of the consequent attribute. We take into account a MEC deployment shown in Figure 2.

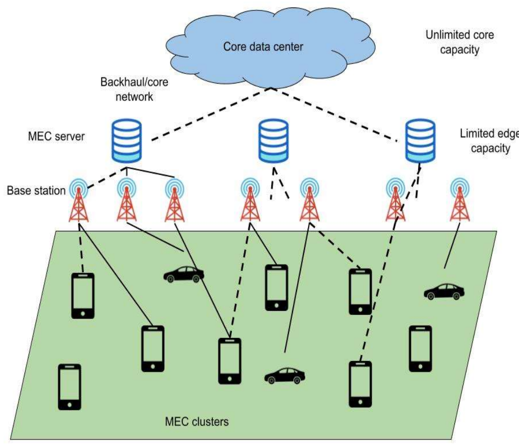


Figure 2. MEC Deployment

The computation identifies MEC groups that will typically increase the traffic dealt with inside the groups and hence reduce the traffic that flows up to the central server farm, given the extreme MEC server limit. Iteratively repeated in two separate stages. Anticipate that we begin with two schematics with similar cluster arrangements. These hubs contrast with the region's discretization, where MEC correspondence requests are distributed in groups. It is marked C. The major graphic $G_a = (C, E_a)$ discusses the relationships between the local clusters. For instance, a node (a matrix cell) can have up to 8 consecutive nodes in a square lattice. The associations between the nodes are addressed in the second diagram, $G_{int} = (C, E_{int})$. It is crucial to remember that a node can connect with itself to form a self-loop, as this symbolizes communications within the corresponding MEC cluster. As a result, there are as many MEC clusters in this initial split as there are nodes.

Algorithm of BRGC

```
Input: Graph of cluster adjacencies and interactio
$G_{int}$(C,$E_{int}$) : undirected
$E_{int}, i,j \in C$ with weight $w_{i,j} \in R$
Edge-weighted graph : $E_a \subseteq E_{int}$ and $e_{i,j} \in$
High cluster capacity: $M$ is $w_{i,j} \leq M$
Assure: $G_a(C, E_a)$ and $G_{int}(C, E_{int})$
1: redo
2: Choose 2 adjacent clusters are high interactio
n
 communication weight:
$i,j \in C$ is:
$\max_{\{e_{i,j} \in E_a\}} \mid w_{w_{i,i}+w_{i,j}+w_{j,j} \leq M} w_{i,j}$
3: Merge $j$ with $i$ in $G_a(C, E_a)$
4: Combine $j$ with $i$ in $G_{int}(C, E_{int})$: $w_{i,i}$
              $\leftarrow w_{i,i} + w_{i,j} + w_{j,j}$
 {Modify $C$ and $E_{int}$ }
 5: There are no merged adjacent clusters: $\forall e_{i,j}$
$\in E_a, w_{i,i} + w_{i,j} + w_{j,j} \geq M$
```

3.2 Distributed MEC Channel Task Allocation Model in Task Sharing

In our method, appropriated MEC network comprises interconnected edge nodes that communicate with one another in a specially appointed way to offer computational types of assistance to MDs. The proposed MEC-based channel task allocation is shown in figure 3. We expect that edge nodes to be associated with an essential power matrix so that we can focus on the energy utilization of MDs. Additionally, since edge nodes are connected to a backend network, we anticipate delays in the organization's ability to deliver assistance requests inside the MEC network would be irrelevant. Administrative demands are directed between edge nodes relying on their asset accessibility, load on ongoing MEC server, and an absolute heap of MEC organization.

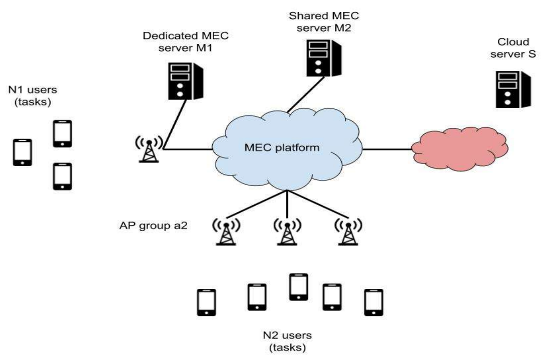


Figure 3. MEC Based Channel Task Allocation

Finally, handled solicitation withdraws from the framework, and the outcome can be put away to the client or cloud. The event that remains balanced at every edge server usage $\rho_j = \lambda_j/\mu_j < 1$ by eq. (6) and eq. (7)

$$Q(n_1, n_2 \dots, n_M) = \prod_{j=1}^{M} Q_j(n_j) \quad (6)$$

$$\lambda_j^e = \Lambda_j^e + \sum_{k=1}^{M} p_{kj}\lambda_k \quad (7)$$

The diagram depicts a load-sharing distributed network with four edge nodes $\{e_1, e_2, e_3, e_4\}$ that create an open queuing network. The edge node receives $\{\Lambda_1^e, \Lambda_2^e, \Lambda_3^e, \Lambda_4^e\}$ Service requests from MDs, and it processes those requests at a pace of $\{\mu_1^e, \mu_2^e, \mu_3^e, \mu_4^e\}$. In the picture, $\{p_{12}, p_{21}, p_{23}, p_{32}, p_{34}, p_{43}\}$ are the related routing probabilities between edge nodes. As an illustration, e_2 passes its traffic load to e_1 and e_3 with the corresponding routing probabilities of p_{21} and p_{23}. Hence, p_{21} and p_{23} represent traffic loads that e_2 forwards to e_2 to e_1 and e_3 is $p_{21}\lambda_2^e$ and $p_{23}\lambda_2^e$ represents the local processing of the remaining services. The notation $(1 - p_{21} - p_{23})\lambda_2^e$ represents the entire amount of incoming traffic on e_2 As a result, we represent the edge nodes utilizing M/M/1 queue. As a result, (8) can be used to express the typical number of jobs W_j^e the jth edge node as:

$$W_j^e = \frac{\lambda_j^c}{\mu_j^e - \lambda_j^e} \quad (8)$$

The service-processing rate $\mu_T^e$of the M edge nodes can be added to get the entire processing capability of the MEC network, as illustrated in eq. (9):

$$\mu_T^e = \sum_{j=1}^{M} \mu_j^e \quad (9)$$

Observe that the MEC network handles all service requests if the total load of forwarded task requests. λ_T^m is equal to or less than the network's all-processing capacity μ_T^e. In such a scenario, eq. (10) illustrates that the proportion of requests is equal to 1:

$$\phi^e = 1, \text{if } \mu_T^e \geq \lambda_T^m \quad (10)$$

In line with this, eq. (11) gives the real processing load on the MEC network, denoted as λ_T^c:

$$\lambda_T^c = \phi^e \lambda_T^m = \lambda_T^m, \text{if } \mu_T^e \geq \lambda_T^m \quad (11)$$

$$T^e(\lambda_T^c) = \frac{\sum_{j=1}^{M} W_j^e}{\lambda_T^e} \quad (12)$$

We observe that the user equipment's decision regarding offloading and channel share are merged. In the event that too many MD repeatedly use the same remote channel to forward computation tasks to the MEC server, each MD belonging to this channel will experience intense obstruction, resulting in a lower uplink correspondence rate. In this case, no more advantageous to perform the task locally. Additionally, wasted are the MEC server's registering resources. Therefore, efficient task offloading and channel asset assignment are required to ensure job completion with little delay and energy consumption. The sheer number of small cells and MDs makes identifying the ideal job offloading option and channel asset allocation strategy challenging.

3.3 Particle Swarm Based Joint Optimization Architecture

Particle swarm optimization is a developmental registering innovation started from an investigation of bird predation conduct. The fundamental thought of molecule swarm advancement calculation is to find an ideal arrangement through coordinated effort and data dividing among people in the gathering. By designing a massless molecule, the molecule swarm computation simulates birds in a herd of birds. Only two attributes apply to molecules: velocity and position. Every molecule looks for ideal arrangement exclusively in hunt space, records it as ongoing individual outrageous worth, and offers singular outrageous worth with different particles in a whole molecule swarm. Which expects to find ideal individual outrageous worth as ongoing worldwide ideal arrangement of whole molecule swarm, all particles in molecule swarm change their speed as well as the position as per ongoing individual outrageous worth they find and ongoing worldwide ideal arrangement shared by whole molecule swarm. The MEC server's CPU clock frequency represented as $\{C_{u,1}, C_{u,2}, C_{u,3} \dots C_{u,M}\}$. the channel advance matrix is described as eq. (13):

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,N} \\ A_{2,1} & A_{2,2} & \dots & A_{2,N} \\ \dots & \dots & \dots & \dots \\ A_{M,1} & A_{M,2} & \dots & A_{M,N} \end{bmatrix} \quad (13)$$

A molecular element is comparable to a certain number of job sets. As mentioned in the previous section, the use of the molecule position vector P = $\{p_1, p_2, .., p_M\}$ to show that all tasks are offloaded to the corresponding MEC server, whose aspect is equal to the number of jobs to be offloaded and whose esteem is added arbitrarily, shows that all tasks are offloaded. We should accept the model's current tasks t_1 to t_5. The Molecule Speed Vector, denoted by V = $\{v_1, v_2, \dots, v_M\}$, is used to address a variety of offloading jobs to various servers. Give all particles a varying speed as a matter of some significance, and update the altered speed value before estimating during the emphasis interaction. The number of jobs to be offloaded equals one element of the molecular speed vector.

4. RESULTS AND DISCUSSION

The main goal of the real-time monitoring tool, which is based on video surveillance, is to spot anomalies and issue alerts. It is obvious that a system prototype like this requires extensive processing (like abnormality detection) and enormous data transport (like recorded video). Additionally, there is a deadline that the jobs must meet. If not, it might negatively impact system performance or have dire repercussions. As a result, this system prototype accurately depicts the essential elements of the application scenario in question, making it appropriate for use as a case study. In this section, using a benchmark dataset with various correspondences, we evaluate our MEC clustering calculation by comparing technique on minor problem occurrences that considers different day types and times of the day. After, we assess it based on significant issue occurrences. Finally, we analyze its results over time.

Dataset: A large part of the advancement in reconnaissance has been conceivable and attributable to the accessibility of public datasets, for example, the Weizmann, VIRAT, and TRECVID datasets. Notwithstanding, the present status of of-the-workmanship reconnaissance frameworks have been immersed by these current datasets, where activities are in compelled scenes, and some unscripted observation film will generally be redundant, often overwhelmed by scenes of individuals strolling. There is a requirement for another video reconnaissance dataset to invigorate progress.

VIRAT: The goal of flying recordings is at 640x480 with a 30Hz framerate. The camera is on a gimbal on a monitored airplane where the regular pixel level of individuals in assortments is around 20 pixels tall. From a sum of 25 hours of unique recordings recorded at this site, a subset of the dataset, which show generally smooth camera movement and great weather patterns (no serious cloud) was physically chosen and remembered for this dataset with the aggregate sum of 4 hours of recordings. Down-sampled renditions were not considered on the grounds that caught-moving articles are as of now, at a genuinely low goal.

TRECVID: At Gatwick Airport in London, UK, where there are severe obstacles and persistent cooperation, five region's data sets were captured. The dataset included five surveillance systems on ten different days, each of which was filmed for around 2 hours. Each report's area and the offices that collected the information were comparable. The enhancement set comprises 100 hours of video appropriated as MPEG-2 organization, de-joined, buddy design, and 720 x 576 resolution at 25 edges per second.

In this part, we explore the exhibition of proposed engineering by concentrating on the impact of various boundaries on offloading goals and playing out near examination with current offloading plans. For trial assessment, we consider a multi-server mobile edge network with enormous smart devices conveyed haphazardly in a given region. A nearby help rate exists for every SD in the scope of [3.5, 5] MIPS. The size of a help demand is haphazardly created for every SD by consolidating MATLAB rand capability that consistently produces solicitation size between [300, 1000] Kb. Neighborhood user has a computer processor recurrence of up to 1 GHZ. The quantity of diverts in full-scale cells is 20.Table 1 shows, Settings of principal reenactment boundaries in this research.

Table 1. Simulation Parameter

Simulation Parameter	Value
System total bandwidth	10MHz
Total number of channels K	20
Number of small cell N	4,6,8,10,12
Number of MUE U_M	15
Number of SUE in a small cell U_n	2,6,10
Transmission power of MUE p_M	27dBm
Transmission power of SUE p_n	23dBm
CPU frequency of UE $f_{u_c}^l$	[0.1,1]GHz
CPU frequency of MEC f_C^C	4GHz
Size of the computation task $s_{u_c}^l$	[2,10]MB
CPU cycles required by the task $\omega_{u_c}^l$	[0.5,2.5]GHZ

User tolerates maximum delay $T_{u_c}^{max}$	[1,4]s

4.1 Proposed Analysis

The Proposed analysis evaluates various parameters based on the number of SD (smart devices). Here the parameters analyzed are QoS, energy consumption, delay, computational load, and throughput.

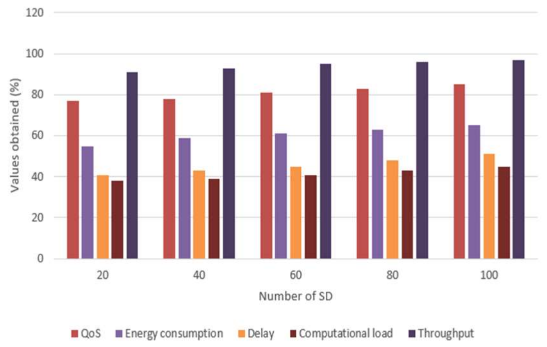


Figure 4. Proposed Technique-Based Parametric Analysis for Number of SD

From figure 4 above, the parametric analysis has been carried out for the proposed technique concerning the number of smart devices used. Proposed technique attained QoS of 77%, the energy consumption of 55%, delay of 41%, a computational load of 38%, a throughput of 91% for 20 SD. QoS of 78%, an energy consumption of 59%, delay of 43%, computational load of 39%, throughput of 93% for 40 SD. QoS of 81%, energy consumption of 61%, delay of 45%, computational load of 41%, throughput of 95% for 60 SD. QoS of 83%, energy consumption of 63%, delay of 48%, computational load of 43%, throughput of 96% for 80 SD. QoS of 85%, energy consumption of 65%, delay of 51%, computational load of 45%, throughput of 97% for 100 SD.

Figure 5 (a)- (e) The parametric analysis for the proposed technique concerning the number of iterations has been carried out. Proposed technique attained QoS of 58%, energy consumption of 45%, delay of 35%, computational load of 51%, throughput of 81% for 10 iteration. QoS of 63%, energy consumption of 51%, delay of 41%, computational load of 55%, throughput of 85% for 30 iteration. QoS of 68%, energy consumption of 55%, delay of 43%, computational load of 58%, throughput of 91% for 50 iteration. QoS of 73%, energy consumption of 58%, delay of 48%, computational load of 63%, throughput of 95% for 70 iteration. QoS of 75%, energy consumption of 61%, delay of 51%, computational load of 65%, throughput of 96% for 80 iteration. QoS of 77%, energy consumption of 63%, delay of 53%, computational load of 65%, throughput of 97% for 90 iteration.

4.2 Comparative Analysis

The benchmark dataset analyzed is Weizmann, VIRAT, and TRECVID in terms of QoS, energy consumption, delay, computational load, and throughput with the comparison of proposed and existing MEC, OFDMA, and C-RAN techniques.

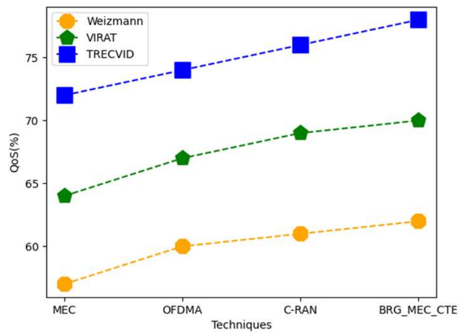
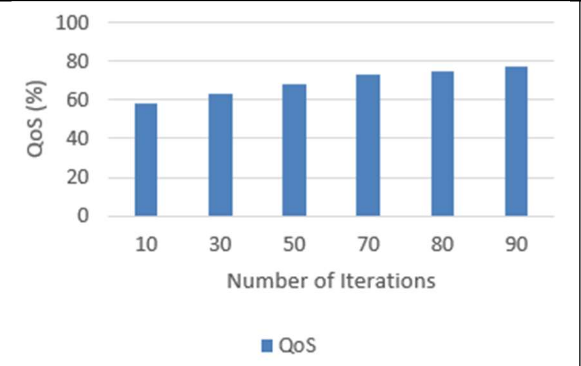
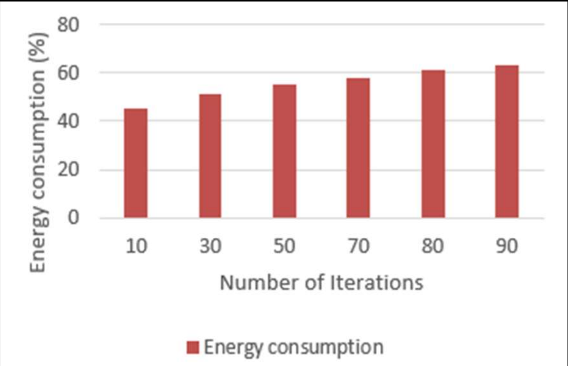


Figure 6. Comparison of QoS

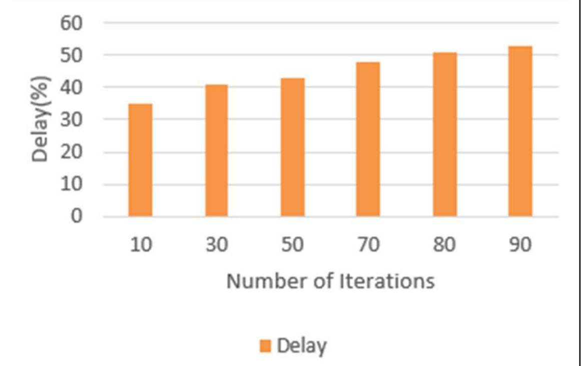
The above figure 6 shows an analysis of Weizmann, VIRAT, and TRECVID datasets for QoS. The proposed technique attained QoS of 62%, existing MEC attained 57%, OFDMA attained 58%, C-RAN attained 61% for Weizmann dataset. For VIRAT dataset, the proposed technique attained QoS of 70%, existing MEC attained 63%, OFDMA attained 67%, C-RAN attained 69%. The proposed technique attained QoS of 76%, existing MEC attained 72%, OFDMA attained 71%, C-RAN attained 77% for TRECVID dataset.



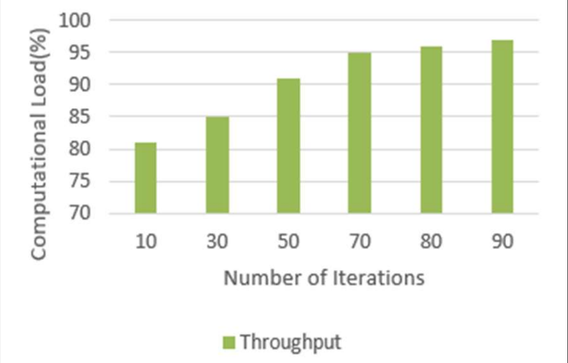
(a) QoS



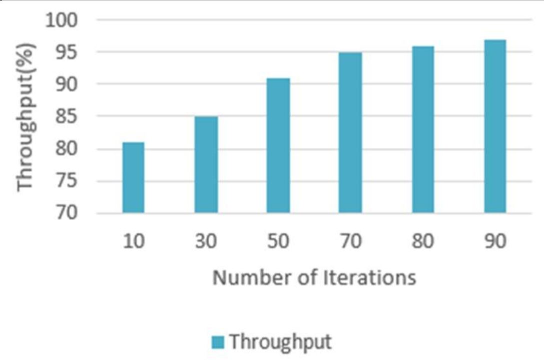
(b) energy consumption



(c) delay



(d) computational load



(e) throughput

Figure 5. (a) - (e) Proposed technique parametric analysis based on the number of iterations in terms of (a) QoS, (b) Energy consumption, (c) delay, (d) computational load, (e) throughput.

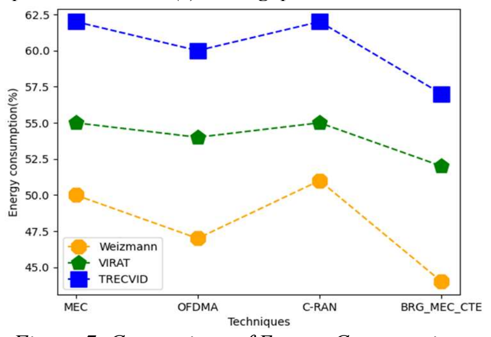


Figure 7. Comparison of Energy Consumption

In figure 7, the analysis in terms of Energy consumption is presented. The proposed technique attained an Energy consumption of 44%, existing MEC attained 50%, OFDMA attained 46%, C-RAN attained 51% for Weizmann dataset. For VIRAT dataset, the proposed technique attained an Energy consumption of 52%, existing MEC attained 55%, OFDMA attained 54%, C-RAN attained 54%. Proposed technique attained Energy consumption of 57%, existing MEC attained 62%, OFDMA attained 62%, C-RAN attained 60% for TRECVID dataset.

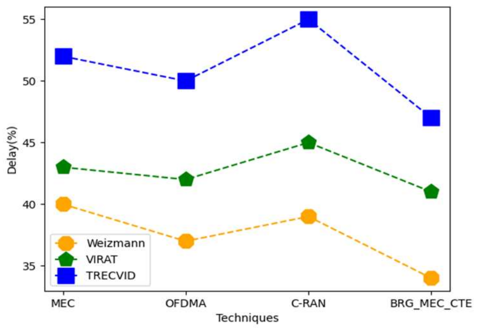


Figure 8. Comparison of Delay

The above figure 8 shows an analysis of Weizmann, VIRAT, and TRECVID dataset for delay. The proposed technique attained delay of 34%, existing MEC attained 40%, OFDMA attained 36%, C-RAN attained 35% for Weizmann dataset. For VIRAT dataset, the proposed technique attained delay of 41%, existing MEC attained 43%, OFDMA attained 42%, C-RAN attained 44%; the proposed technique attained delay of 47%, existing MEC attained 52%, OFDMA attained 50%, C-RAN attained 53% for TRECVID dataset.

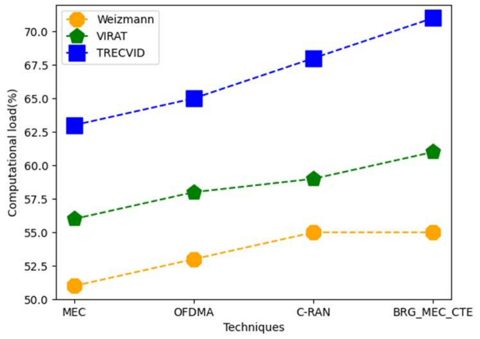


Figure 9. Comparison of Computational Load

In figure 9, the analysis in terms of Computational load is presented. The proposed technique attained a Computational load of 54%, existing MEC attained 51%, OFDMA attained 53%, C-RAN attained 54% for Weizmann dataset. For VIRAT dataset, the proposed technique attained computational load of 60%, existing MEC attained 56%, OFDMA attained 57%, C-RAN attained 58%. The proposed technique attained Computational load of 71%, existing MEC attained 63%, OFDMA attained 65%, C-RAN attained 67% for TRECVID dataset.

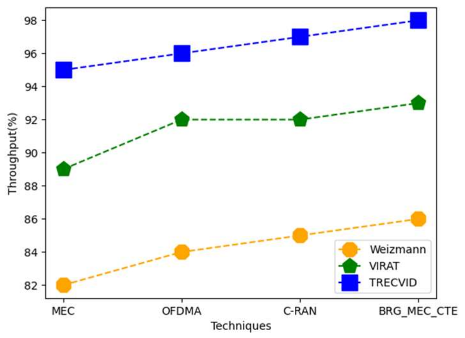


Figure 10. Comparison of Throughput

The above figure 10 shows an analysis of Weizmann, VIRAT, and TRECVID dataset for throughput. The proposed technique attained throughput of 86%, existing MEC attained 82%, OFDMA attained 84%, CRAN attained 85% for Weizmann dataset; for VIRAT dataset, proposed technique attained throughput of 93%, existing MEC attained 89%, OFDMA attained 92%, CRAN attained 92%. The proposed technique attained throughput of 97%, existing MEC attained 95%, OFDMA attained 96%, CRAN attained 96.5% for TRECVID dataset.

4.3 Industrial Significance of Proposed Model

Industrial IoT uses edge computing at the network's edge. Industrial IoT devices are wirelessly connected to edge servers for Industry 4.0 duties. Industry 4.0 requires low latency and high computational resources for security applications, AR/VR gadgets, real-time cyber-physical systems, and autonomous cars. Critical application delays cause industry delinquency. More IIoT devices and their heterogeneity create increased pressure on computing resources. We presented collaborative task execution at distributed edge Based on belief rule geo clustering and particle swarm joint optimization with the multi-constraint objective model for time-critical and resource-intensive IIoT tasks. MEC resource allocation reduces processing time, energy cost, and load. Critical industrial use cases with running applications like AR/VR for field workers, real-time predictive maintenance, object

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

recognition and tracking of human movements, process monitoring, robotic control, etc., would benefit from the suggested model.

5. CONCLUSION

In collaborative edge computing, where heterogeneous independent tasks are produced at various heterogeneous devices at various release times, this work examines the multi-task collaborative execution problem with a multi-constrained objective. The novel strategy for MEC server clustering, collaborative task scheduling, and execution in the resource allocation model is investigated. Belief rule Geo clustering method used to cluster the MEC server based on real-time data of intensive tasks. For the simple sake, real-time monitoring application based on video surveillance is taken. When the MEC server is busy receiving data, the distributed MEC channel task allocation model is utilized to distribute tasks, and a particle swarm-based joint optimization architecture is used to optimize clustered data. We concentrate on the proposed framework by reenactment to grasp its adequacy under different boundaries and exhibit its viability over accessible MEC offloading plans. We have figured out the issue as a joint improvement issue that plans to limit a direct mix of data transmission consumed and network QoS. We mean to incorporate boundaries similar to yield transmission postponement and organization delays inside the MEC organization. We may likewise expand our method for multi-class traffic and incorporate steering likelihood improvement for better resource usage. Proposed technique attained QoS of 77%, energy consumption of 63%, delay of 53%, computational load of 65%, throughput of 97%.

REFERENCES:

[1] Hadi, M., & Ghazizadeh, R. (2022). Joint resource allocation, user clustering and 3-D location optimization in multi-UAV-enabled mobile edge computing. *Computer Networks*, 218, 109420. https://doi.org/10.1016/j.comnet.2022.109420

[2] Du, J., Sun, Y., Zhang, N., Xiong, Z., Sun, A., & Ding, Z. (2022). Cost-Effective Task Offloading in NOMA-Enabled Vehicular Mobile Edge Computing. *IEEE Systems Journal*,17(1),928-939. https://doi.org/10.1109/JSYST.2022.3167901

[3] Li, X., Lan, X., Mirzaei, A., &Bonab, M. J. A. (2022). Reliability and robust resource allocation for Cache-enabled HetNets: QoS-aware mobile edge computing. *Reliability Engineering & System Safety*, *220*, 108272. https://doi.org/10.1016/j.ress.2021.108272

[4] Asghari, A., &Sohrabi, M. K. (2022). Multiobjective Edge Server Placement in Mobile-Edge Computing Using a Combination of Multiagent Deep Q-Network and Coral Reefs Optimization. *IEEE Internet of Things Journal*, *9*(18), 17503-17512. https://doi.org/10.1109/JIOT.2022.3161950

[5] Ziaeddini, A., Mohajer, A., Yousefi, D., Mirzaei, A., &Gonglee, S. (2022). An Optimized Multi-Layer Resource Management in Mobile Edge Computing Networks: A Joint Computation Offloading and Caching Solution. *arXiv preprint arXiv:2211.15487*. https://doi.org/10.48550/arXiv.2211.15487

[6] Shakarami, A., Shakarami, H., Ghobaei-Arani, M., Nikougoftar, E., &Faraji-Mehmandar, M. (2022). Resource Provisioning in edge/fog computing: A comprehensive and systematic review. *Journal of Systems Architecture*, *122*, 102362. https://doi.org/10.1016/j.sysarc.2021.102362

[7] Youn, J., & Han, Y. H. (2022). Intelligent task dispatching and scheduling using a Deep Q-Network in a cluster edge computing system. *Sensors*, *22*(11), 4098. https://doi.org/10.3390/s22114098

[8] Lakzaei, M., Sattari-Naeini, V., SabbaghMolahosseini, A., &Javadpour, A. (2022). A joint computational and resource allocation model for fast parallel data processing in fog computing. *The Journal of Supercomputing*, *78*(10), 12662-12685. https://doi.org/10.1007/s11227-022-04374-x

[9] Mahmud, R., Pallewatta, S., Goudarzi, M., &Buyya, R. (2022). iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Journal of Systems and Software*, *190*,111351.https://doi.org/10.1016/j.jss.2022.111351

[10] Mohajer, A., Daliri, M. S., Mirzaei, A., Ziaeddini, A., Nabipour, M., &Bavaghar, M. (2022). Heterogeneous computational resource allocation for NOMA: Toward green mobile edge-computing systems. *IEEE Transactions on Services Computing*. https://doi.org/10.1109/TSC.2022.3186099

[11] Li, F., Fang, C., Liu, M., Li, N., & Sun, T. (2023). Intelligent Computation Offloading Mechanism with Content Cache in Mobile

Edge Computing. *Electronics*, *12*(5), 1254. https://doi.org/10.3390/electronics12051254

[12] Bréhon–Grataloup, L., Kacimi, R., &Beylot, A. L. (2022). Mobile edge computing for V2X architectures and applications: A survey. *Computer Networks*, *206*, 108797. https://doi.org/10.1016/j.comnet.2022.108797

[13] Jamil, B., Ijaz, H., Shojafar, M., Munir, K., &Buyya, R. (2022). Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, *54*(11s), 1-38. https://doi.org/10.1145/3513002

[14] Huang, Y. Y., & Wang, P. C. (2023). Computation Offloading and User-Clustering Game in Multi-Channel Cellular Networks for Mobile Edge Computing. *Sensors*, *23*(3),1155. https://doi.org/10.3390/s23031155

[15] Vijayasekaran, G., &Duraipandian, M. (2022). An efficient clustering and deep learning based resource scheduling for edge computing to integrate cloud-IoT. *Wireless Personal Communications*, *124*(3), 2029- 2044. https://doi.org/10.1007/s11277-021-09442-8

[16] Mousa, M. H., & Hussein, M. K. (2022). Efficient UAV-based mobile edge computing using differential evolution and ant colony optimization. *PeerJ Computer Science*, *8*, e870.https://doi.org/10.7717/peerj-cs.870

[17] Mei, M., Yao, M., Yang, Q., Qin, M., Kwak, K. S., & Rao, R. R. (2022). Delay analysis of mobile edge computing using Poisson cluster process modeling: A stochastic network calculus perspective. *IEEE Transactions on Communications*, *70*(4), 2532-2546. https://doi.org/10.1109/TCOMM.2022.3151879

[18] Bute, M. S., Fan, P., Liu, G., Abbas, F., & Ding, Z. (2022). A cluster-based cooperative computation offloading scheme for C-V2X networks. *Ad Hoc Networks*, *132*, 102862. https://doi.org/10.1016/j.adhoc.2022.102862

[19] Lu, Y., Chen, X., Zhang, Y., & Chen, Y. (2022). Cost-efficient resources scheduling for mobile edge computing in ultra-dense networks. *IEEE Transactions on Network and Service Management*, *19*(3), 3163-3173. https://doi.org/10.1109/TNSM.2022.3163297

[20] Hamadi, R., Khanfor, A., Ghazzai, H., &Massoud, Y. (2022), August). A hybrid artificial neural network for task offloading in mobile edge computing. In *2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS)* (pp. 1-4). IEEE. https://doi.org/10.1109/MWSCAS54063.2022.9859520

[21] Gao, H., & Liu, J. (2022). Intelligent collaboration under internet of things and mobile edge computing. *Mobile Networks and Applications*, *27*(4), 1421-1422. https://doi.org/10.1007/s11036-022-01998-4

[22] Li, H., Zheng, P., Wang, T., Wang, J., & Liu, T. (2022). A multi-objective task offloading based on BBO algorithm under deadline constrain in mobile edge computing. *Cluster Computing*, 1-17. https://doi.org/10.1007/s10586-022-03809-7