IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 March 2025, accepted 19 March 2025, date of publication 25 March 2025, date of current version 7 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3554638

RESEARCH ARTICLE

Blockchain and RL-Based Secured Task Offloading Framework for Software-Defined 5G Edge Networks

N. SETHU SUBRAMANIAN[1], (Member, IEEE), PRABHAKAR KRISHNAN[1], (Senior Member, IEEE), KURUNANDAN JAIN[1], K. B. ANEESH KUMAR[2], TULIKA PANDEY[3], AND RAJKUMAR BUYYA[4], (Fellow, IEEE)

[1]Center for Cybersecurity Systems and Networks, Amrita Vishwa Vidyapeetham, Amritapuri Campus, Kollam 690525, India
[2]Cyber Security Group, Center for Development of Advanced Computing (CDAC), Thiruvananthapuram, Kerala 695033, India
[3]Research and Development in Cybersecurity Group, Ministry of Electronics and Information Technology (MeitY), Government of India, New Delhi 110003, India
[4]Cloud Computing and Distributed Systems Laboratory, School of Computing and Information Systems, The University of Melbourne, Melbourne, VIC 3010, Australia

Corresponding author: N. Sethu Subramanian (sethus@am.amrita.edu)

ABSTRACT 5G (Fifth Generation) technology represents a significant advancement in telecommunications, facilitating industry innovation and advancement for applications across various sectors. It enhances high-speed, low-latency communication, making it ideal for task offloading in resource-constrained mobile devices. By leveraging task offloading, 5G networks maximize the efficiency of both computational and network resources, ensuring faster, more reliable data delivery and enabling high-performance requirements of modern applications. However, dynamic and secure task offloading remains a challenge due to fluctuating network conditions and trust concerns. This paper proposes SAGE (Secured Adaptive Generalized Edge), a reinforcement learning-based task offloading framework that leverages contextual bandits for adaptive decision-making in Multi-Access Edge Computing (MEC) environments. By integrating blockchain smart contracts for security and SDN-based orchestration for dynamic resource management, SAGE ensures robust, low-latency offloading. Experimental evaluations demonstrate that SAGE reduces task offloading delays by 36% and task durations by 30% compared to baseline methods under varying load and energy constraints.

INDEX TERMS 5G mobile, software-defined-networking (SDN), mobile edge-computing (MEC), blockchain-assisted offloading, fog/edge computing, security, trust modeling, dynamic edge server offload (DESO), secured adaptive generalized edge (SAGE).

I. INTRODUCTION

Advancements in cloud computing and wireless communications have increased mobile network connectivity for applications such as wearable devices, VR streaming, self-driving automobiles, wearable social media, and vehicle systems [1]. These edge applications require resources with continuous data processing capabilities where communication networks need to undertake computational offloading from devices in support of the future [2]. Current "smart" devices lack the processing powers needed to execute these activities. Ultra-dense networks (UDN), which include small and macro cell Base stations (BSs), are key technologies in 5G that can handle these environments. Computations dependent on networks have significant delays where the deployment of edge clouds or fog nodes on networks have been probable solutions. Edge cloud computing has two distinct advantages over local computing environments [3]:

1) They are alternatives for overcoming the restricted computational capacities of mobiles, and
2) They substantially minimize latencies while offloading computations to Blockchain and Learning-Based

The associate editor coordinating the review of this manuscript and approving it for publication was Bilal Khawaja.

Secure task offloading framework for software-defined 5G edge networks to distant clouds.

MEC, uses network edges to move tasks to the nearest data centers, is a better workaround for time-sensitive and computationally demanding tasks [4]. MEC may help mobile devices execute resource-intensive apps. However, establishing trust amongst several stakeholders in MEC is difficult because these parties have conflict of interests with one another. Integrating MEC with blockchains is another area with several advantages in computations, flexibility, data integrity, and security with minimized latencies. Edge computing employs computational offloads to disperse tasks to idle and computation-free edge nodes that share workloads. These offloads can be to local or remote devices. They ensure continuous data flows and avoid needless computational latencies and service delays. Effective offloading in MEC results in several Quality of Service (QoS) benefits [5]. Task offloads involve shifting tasks from low-powered devices at edges to higher-powered devices or clouds. Content offloads encompass caching heavy data contents like multimedia maintained at BSs or neighboring edge gateways. Mobile devices can upload tasks to ESs using migration mechanisms. Future communication systems can integrate UDNs with edge clouds for offloading processing to users [6], though most current developments in MEC and UDNs are done separately [7]. The integration of these two need to handle:

- Management of dispersed computing resources.
- Consideration of offloading computing tasks.
- Security and privacy challenges over wireless channels.

SDNs are cutting-edge network designs that can address these issues with their controller-based resource management [8] and can assist cloud service providers with centralized controllers to adjust resource sharing and offload computations dynamically to satisfy end-user demands depending on their dynamic service selections. Blockchains are also new paradigms considered in this work's architectural design as they can better control edge services and devices, making them an impertinent part of MEC settings. Distributed digital ledger systems also offer security for network data, privacy, integrity checks, and resource access [9].

A formal instance of edge computing offloads supported by blockchain is shown in Figure 1. Task offloading frameworks are susceptible to mismanagement and misuse by bad actors in the network. Most implementations do not consider security and are not executed in trusted environments. This study examines integrated optimizations of communications and computing resource allocations and partial offloading ratios while considering offloading periods and security requirements to improve total secrecy offloading data (TSOD). The proposed system architecture is for 5G device trust-based work offloads [10]. Deep learning and reinforcement learning are two examples of artificial intelligence technologies that have developed as effective ways to handle compute offloading difficulties in recent years. In complicated and dynamic situations, reinforcement learning (RL) can be effective. In unpredictable environments, Reinforcement Learning (RL) agents learn how to maximize their rewards across several decisions. RL agents are used to learn offloading procedures that may be dynamically changed to real-time conditions, enabling intelligent task allocation. RL helps the system learn optimal policies over time, tackling task-offloading decision-making issues such as edge node trustworthiness-offloading ESs and dynamic network conditions [11]. This allows efficient and reliable task distribution while preserving privacy, fairness, and security during unloading. This architecture targets computational offloads with low device resources. Cryptographic 5G blockchain devices synchronize edge nodes securely using a networked clock mechanism. Discussed the benefits of the proposed architecture for safe task offloading in 5G MEC and future applications, there are certain challenges and limitations pertaining to interoperability, scalability, security, and cost which is briefly explained towards the end of this research work. The research's significant contributions are:

- Blockchain and smart contracts in multi-user/server MEC-enabled 5G and wireless networks to manage and secure Edge-to-Cloud real time offloading for privacy, and security.
- SAGE, a RL based SDN framework, jointly addresses security and workload optimization by incorporating threat detection, secure decision-making, and adaptive offloading policies. It ensures safe task execution while minimizing delay and resource waste in dynamic environments.
- DESO (Dynamic Edge Server Offload) algorithm in the proposed architecture dynamically identifies and assigns tasks to the most reliable and capable MEC servers. It detects harmful or overloaded edge servers to reduce the risk of task delays or failures, thereby improving offloading reliability and efficiency.

This study found 20% of work time reductions and 30% of energy cost savings over random and even job offloads. These findings demonstrate that the integrated RL and DESO methodology optimizes work offloading in 5G-enabled MEC systems.

II. RELATED WORK

This section details work related to 5G technology, including offloads, security, and clouds. Numerous studies have examined how computational offloads may reduce network energy utilization and service response times. Table 1 depicts the comparison of few of the research approaches, its advantages and the limitations. Applications like Clonecloud [12], [13] offloads heavy or complex computations from mobile devices to clouds mainly due to reduced execution times and sharing of findings. Significant delays can be caused while shifting tasks from cellular networks to clouds where MEC was used to offload tasks to the edges of multiple users in [14]. Though multi-server MECs were deemed in [15], UDNs were disregarded. Hence, this work proposes SDN-based task

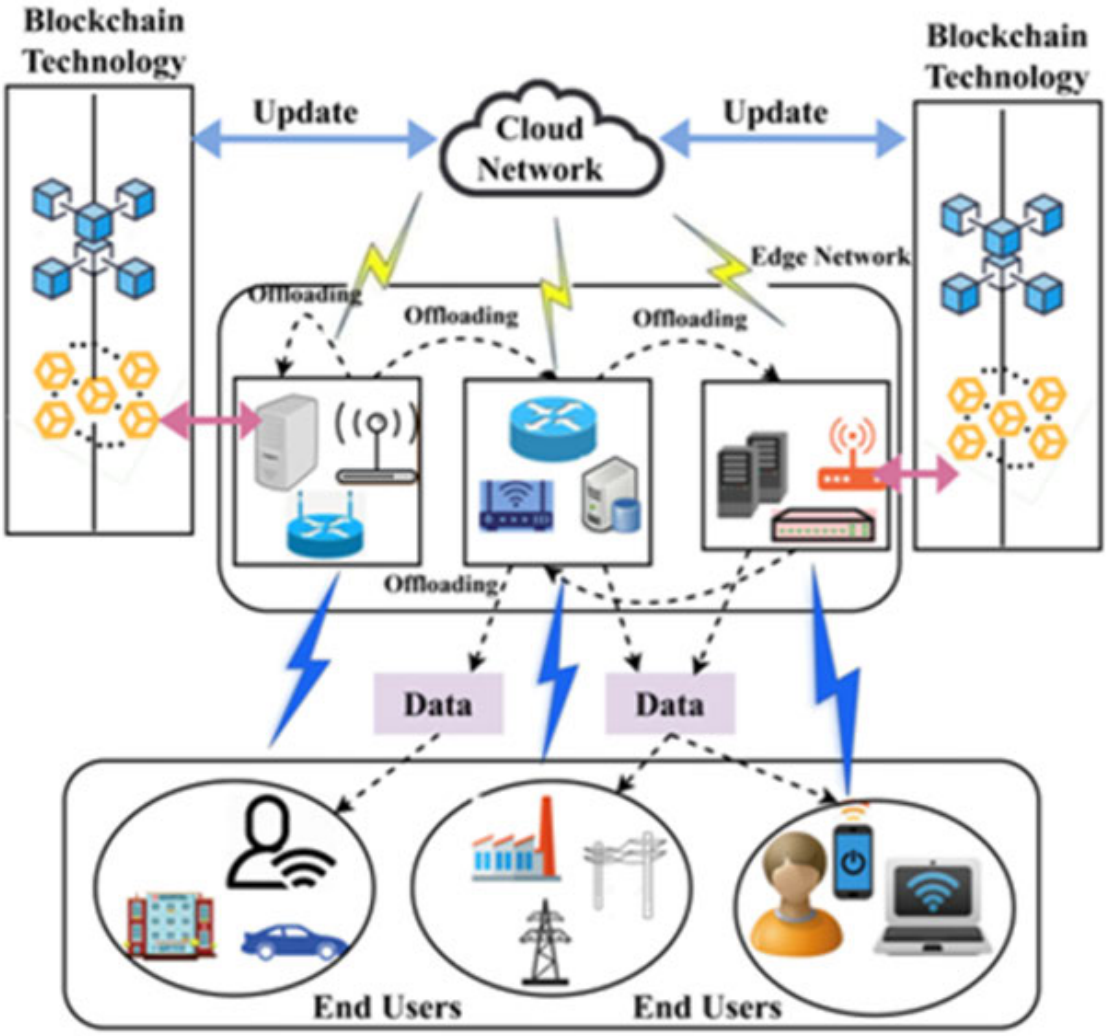


FIGURE 1. Block chain-based edge computing and offload framework.

offloads for MEC in UDNs with efficient task offloads. Most works on blockchains and learning-based work offloads are conducted independently. Historically, the bulk of approaches were user-centric. For example, [17] investigated trustful task offloads of end users in MECs using blockchains and reinforcement learning, while [18] employed DNNs (Deep Neural Networks) for offload decisions to ESs in mobile blockchain networks. The study in [17] examined end-user-centered offloading strategies that included blockchains, machine learning, and game theories, while ESs began to host typical end-user perspectives [19]. None of these, however, used blockchain technology to address security and privacy concerns. A blockchain-based method called Task Offloading and Resource Allocation (TO-RA) was created from smart contracts for resource allocation. MEC for wireless blockchain networks was proposed in [20], where MEC servers mimicked edge nodes and block's cryptographic hashes blocks. User groups were formed by offloading tasks in networks or adjacent access points. Distributions were handled in [6] for huge data using the resource-efficient blockchain called the Alternating Direction Method of multipliers (ADMM). According to the study in [21], SDN-UDNs for MEC may be used to offload tasks, reduce delays, and extend battery life. Systems that use Trust and Reputation Systems (TRSs) for task offloading must also be secure. Several investigations into assaults against TRSs include attacks like (i) speaking poorly about someone, speaking well about someone, colluding, acting selectively, and self-promotion. Most attack evaluations concentrate on network-level protocols routing packets in networks. System resource constraints make them vulnerable to several attacks that apply to conventional wireless systems (such as eavesdropping, jamming, replay, and others). Investigations into the security of communication layers like DTLS and OSCORE have been prompted by systems that depend on the security these levels provide. Threat modeling

research is highlighted in this part for suggestive trust-based job offloading system architectures that carry executions from constrained resource domains to resource-rich device domains. This implies the need to identify attacks and threats to systems, a deeper understanding of system components, interactions, and how assaults can be integrated to help adversaries. DFD employs De Marco notation to help with threat modeling for this system [22]. Data flows from context databases to entity borders demonstrate the necessity of OSCORE security contexts in communications. Systems may confront three prominent threat actors: (i) malicious resourceful edges, (ii) malicious devices with constrained resources, and (iii) external threat actors. Threats can be categorized as Tactics, Techniques, and Procedures (TTPs), where techniques are precise explanations of conduct in contexts of methods, tactics are high-level explanations of behaviors, and procedures are at lower levels where specific methodologies are described [23]. Attackers can easily jam wireless transmissions using DoS attacks, making communications impossible as these attacks involve flooding systems with requests. This increases the costs of conducting cryptographic operations. In contrast to previous techniques, ESs have devices with limited resources and must employ suitable offloads [24], [25]. With 5G, several ESs may be deployed to deliver computing services to customers [26], [27]. Given the complexity of 5G applications, solitary actions are usually subdivided into smaller tasks to allow for multi-threaded processing and increased task execution efficacy [28]. Despite the efforts in these papers, challenges in offloading persist. Reinforcement Learning (RL) is an optimistic method that lets a learning agent experiment and learn from mistakes to find the best solution without prior knowledge of the environment [29]. It is strongly believed that complex offloading problems involving multiple-use IoT devices result in a high-dimensional state and action space. This, in turn, causes RL-based solutions to be inefficient [30]. Thankfully, researchers have developed Deep Reinforcement Learning (DRL) methods [31], [32], such as deep Q-network (DQN) to solve high dimensional offloading task problems. The authors in [33] focused on a strategy for shifting blockchain mining duties to edge clouds, to improve the quality of service (QoS) and reduce the workload on mobile miners. The study described in [34] introduced a Mobile Edge Computing (MEC) that incorporates blockchain technology to enhance future wireless networks. Spectrum distribution, block size, and the number of successive blocks are all factors that the model takes into account to maximize computation offloading and resource allocation. This optimization is achieved using a double-dueling deep Q network. [35] examined a collaborative computation offloading framework for IoT networks that utilizes blockchain technology. The Multi-Agent Deep Reinforcement Learning (MA-DRL) algorithm lets IoT devices collaborate to examine offloading scenarios to reduce long-term expenses associated with offloading. Study in [45] describes offloading decision in MEC, however, privacy and security were not considered. The study presented in reference [31] examines the security and computation offloading challenges in a multi-user system with blockchains using DRL with Q-network. It is observed that combining contextual bandits with DRL techniques produces better outcomes.

III. PROPOSED SDN-ENABLED ARCHITECTURE

The proposed architecture primarily focus on 5G devices on the 5G network, which provides ultra-low latency, high bandwidth, extensive device connectivity essential for large-scale, real-time, and secure job offloading. The dynamic characteristics of task migration, coupled with the necessity for trust-aware judgments, necessitate a network infrastructure proficient in managing high-speed and low-latency communications — a capability that 4G and previous generations cannot consistently assure. Furthermore, MEC is most efficacious when seamlessly integrated with 5G infrastructures, as it was conceived with 5G architecture to reduce round-trip latency and improve computational offloading performance. Legacy devices and networks generally do not possess inherent support for MEC and SDN, which are essential to our approach. The foundational features of the proposed architecture — such as safe offloading via blockchain and adaptive learning can potentially be adapted to non-5G contexts; however, modifications would be necessary to address constraints in latency, throughput, and edge computing integration. These extensions are beyond the scope of the current study but are recognized as potential avenues for future investigation. The 5G MEC paradigm presents difficulties in developing effective tactics that offload applications to fog or cloud layers while maintaining the best possible service response times. Though response times are dominated by execution times of traditional computing offloading policies, variables such as application features and contextual considerations can affect reaction times. Most of the available literature on the computation offloading problem offers effective solutions but considers only a few factors like computing capacities and network bandwidths, ignoring other important factors. This section presents SAGE, the proposed secure edge computing/offloading framework. Figure 2 shows the proposed TCO Architecture, the structure for MEC job offloads in SD-UDNs includes three planes, namely control, data, and user planes. User planes encompass users who offload tasks. SD-UDN controllers integrated into macro cell BSs implement control planes. Wireless links connect users to small or macro cell BSs where the former is implicitly connected to centralized macro cell BSSs by high-speed front haul networks. Macro-cell BSs provide control coverage, while small-cell BSs provide data coverage. The SD-UDN controllers oversee all critical control functions, including allocations of resources and schedules, and acquire mobile user information through small cell BSs. Job offloading mechanisms in SD-UDNs are based on control, data plane separations, and aggregations in macro cell BSs. The centralized controllers separate computational and control features from data from small cell BSs. These

TABLE 1. Summary of related work section.

Sl No	Author(s)	Approach	Advantage(s)	Limitation(s)
1	A. Ksentini, et al [13]	Approach that decides whether a service should be migrated to a data center when the relevant user equipment (UE) is involved	The proposed solution always gains maximum rewards by deciding whether to migrate a service or not, as the migration itself is an expensive operation.	Potential delay in the service migration
2	T. X. Tran et al [15]	An integrated approach to job offloading and resource allocation in a multi-server MEC-assisted network and maximize user experience.	Average system offloading utility has been significantly improved compared to the other traditional approaches.	With more users, the system utility decreases.
3	M. Liu, et al [16]	Optimized scheduling, resource allocation, and block size adaptation using a blockchain-based architecture	Provided a tight lower bound and the solution outperformed the baseline schemes when compared to other schemes.	The communication and computing capabilities were not considered as part of the design solution.
4	L. Xiao, et al [17]	A blockchain-based MEC trust mechanism that includes improved mobile device computing performance, safeguarding against selfish edge attacks, and mitigating deception in service record fraud.	Deep RL based CPU utilization algorithm is used to improve the edge utility by reducing the response latency and the energy consumption.	Security and privacy were not considered in the approach.
5	N. Luong, et al [18]	An analytical approach for mobile blockchain networks' edge resource allocation underpins a multi-layer deep learning neural network architecture.	Increased revenue of Edge Computing Service Provider while guaranteeing Dominant-Strategy Incentive Compatibility (DSIC) and Individual Rationality (IR).	The solution has not considered multiple edge computing resources.
6	L. Chen, et al [19]	Online PEer OffloadiNg, or OPEN, is a new architecture that allows networks of MEC-enabled Small Base Stations to offload stochastic computations to one another.	Facilitates centralized and independent decision-making, accompanied by performance assurances.	Solution is not certain in the event of inaccurate assessment of task arrival rates
7	M. Liu, et al [20]	A wireless blockchain architecture that is enabled by MEC was presented in the research. As part of its optimization process, the suggested solution examined compute offloading and content caching.	Solution shows better performance with deterministic constraints.	QoS constraints have not been considered while arriving at the results.
8	C. Xu et al [6]	Utilizes blockchain to provide reliable large data sharing in edge collaboration.	The suggested approach minimizes the waste of computing resources.	The restricted computing, network, and storage capabilities of edge devices provide challenges to solution design.
9	M. Chen and Y. Hao [21]	The technique improves task offloading by breaking it down into two parts: a. assigning tasks and b. allocating resources. The two sub problems provide a method for offloading that is efficient.	Task offloading is performed with SD-UDN. More efficient than random and uniform computation offloading techniques	Solution has not been verified with user mobility.

TABLE 1. *(Continued.)* **Summary of related work section.**

10	L. Chen, et al [26]	Computing offloading techniques that integrate device, edge, and distant cloud cooperation are part of the solution.	Solution outperforms by 23% - 46% in terms of application completion.	Solution has not considered the cost factor involved. Also there exists the challenge in offloading tasks.
11	M. Wang, et al [27]	Solution for task scheduling strategy in MEC for IoT systems.	The proposed scheduling strategy reduces both the overall time required to finish jobs and the average execution time for tasks by 50%.	Consideration of dependencies between tasks and find a proper task execution order for efficient performance of the solution.
12	Y. Liu, et al [28]	An effective task scheduling system that prioritizes applications and their respective tasks to ensure adherence to completion time restrictions	The solution has significantly decreased the average completion time of several applications within the stipulated time restrictions.	Solution has not considered the task communication delay
13	X. Qiu, et al [31]	Offers a way for blockchain-enabled MEC to handle online computations using deep reinforcement learning..	To maximize long-term rewards and enhance offloading performance, the DRGO method employs model-free deep reinforcement learning.	The technique is inadequate for addressing the convergence issues in deep reinforcement learning.
14	M. Li, et al [32]	An optimized framework for combined caching, computation, and security in M2M communications based on dueling deep Q-networks (DQNs) is proposed, which makes use of blockchain technology and edge computing	Reduces network expenses, improves data security, and ensures faster data delivery.	Have ignored smart city integration and energy-efficient M2M communications concerns.
15	S. Guo, et al [33]	Offload Blockchain mining duties to edge clouds.	When miners offload to both the Collaborative Mining Network (CMO) and the Edge Cloud Operator (ECO), they often earn more than when they unload to ECO alone.	With differentiated pricing, the average demand for resources diminishes beyond a certain threshold due to elevated costs resulting from an increase in the number of mining equipment
16	Z. Li, et al [35]	Suggested a multi-agent DRL architecture for cooperative compute offloading to attain long-term performance	With adjustments to the system's parameters, the suggested algorithm's performance becomes more stable and robust, and the amount of training time is dropped by about 60%	Solution has not been verified with the untrusted relays, which could eventually lead to information leakage.

controllers can distinguish and gather data from cloud edges and mobile devices, and mobiles can execute tasks locally or send them to ESs based on user preferences. SD-UDN controllers are responsible for updating task, BS, and mobile device information tables. This information table contains data such as radio access loads and edge cloud computing loads. Tables of task information provide listings of job kinds, data volumes, and task computation amounts. Regularly, users provide measurement data to serving BSs in the area, which integrate it with edge cloud data from many users and send it to the SD-UDN. SD-UDN controllers update all information tables, give task offloading policies for mobile devices, and determine how to assign resources to the edge cloud depending on task latency and energy usage.

A. SAGE FRAMEWORK

BSs offer computation and communication services to their coverage nodes. Edge server computing allows nodes to outsource their jobs to adjacent ESs with extra resources when BSs are overloaded. Thus, overloads are handled by shifting node workloads to ESs. The blockchain-based edge server computing infrastructure comprising of SDN Controller (SDNC) and three main components in data planes, BSs, nodes, and ESs are detailed below (See Figure 3).

1) BSs: Blockchains are maintained by authorized BSs acting as consensus nodes which encompass:
   - Register Authority Components (RAC): RAC controls identity management and registrations. It refers to mobile operators 'Certificate Authorities (CA) and unique digital certificates for guaranteeing legitimacy.
   - Computing Components (CCs): CCs are responsible for designing and executing smart contracts. They are a part of blockchain mining for financial gains.
   - Storage Components (SC): SCs ensure block validations and task offloading and maintain blockchain ledgers.
2) Nodes: These are network endpoints for redistributions or transmissions of data. They recognize, process, and route communications to other nodes automatically when programmed.
3) Edge Servers: These are powerful computers existing at networks' edges. Their physical locations are nearer to systems or applications generating data or used by ESs.

Blockchain-based edge server implementations are detailed as follows: All entities must first get hold of safe wallets containing many digital currencies. Like Bitcoin [36], a digital coin is a form of virtual money hosted on Ethereum [37] used to settle task-offloading transactions. The entities generate public and private keys separately. The public and private keys of the devices are used for data encryption and decryption. Digital signatures are created and verified using BSs' public and private keys, where BSs share the same key pairs. Nodes register with RACs for receiving certificates. Secondly, nodes choose one of the available ESs and notify BSs of their need for job offloads and required latencies. The BSs then carry out task unloading of smart contracts. The chosen ESs receive nodes offloaded encrypted data. ESs process the received data and send results to BSs and nodes. Thirdly, BSs examine offloads to detect malicious activities. Honest nodes are rewarded with virtual cash in line by smart contracts, while dishonorable nodes are fined. BSs create transaction blocks and upload them to SDUDN through blockchains while remaining BSs tie amongst themselves to find real PoW. BSs broadcast transactional blocks to other BSs for verifications, and on authorization by most BSs, the blocks are put at the ends of blockchains.

B. THREAT MODEL

The following assumptions are made: (i) Blockchain nodes are safe, (ii) 5G devices are not disposed to exchange private keys with one another. Despite securing nodes, certain nodes may engage in harmful behaviors posing security risks. These dangers are categorized and explained below:

- Nodes Engaging in Malicious Behaviors: ''Double claim'' attacks refer to malicious node attempts to claim rewards several times for increasing profits. A rogue node may undertake repudiations to prevent ESs from receiving rewards even after the successful completion of tasks. They can cheat BSs and deny inputs from ESs.
- ESs with harmful Behaviors: Malicious ESs or freeride attacks attempt to gain incentives without executing any computations. They may allocate minimal or no computing resources, resulting in taskoffloading delays and failures. This threat model assumes that ESs are honest and cannot produce false results.
- BSs that Display Malicious Behaviors: Attackers may change the CCs of BSs using remote hijacks. The hacked CCs may not be able to settle transactions fairly i.e. they may refuse rewards to trustworthy nodes, and repudiation attacks may ensue
- Privacy Disclosures: Despite the anonymity of nodes during task offloads, attackers may deduce personal information by intercepting task data.

C. NETWORK MODEL

Assuming SD-UDNs include densely deployed BSs with edge clouds where B $= \{b_1, b_2, \ldots, b_n\}$ stands for Mobile Edge Clouds while U $= \{u_1, u_2, \ldots, u_m\}$ represents users of SD-UDNs. Mobiles have computational workloads that can be executed locally or offloaded. These tasks Q_i can be detailed using task models [38] i.e., $Q_i = (i, s_i)$, where s_i denotes sizes of computational tasks Q_i i.e., data inputs and associated processing codes sent to edge clouds, and i denotes computational parts of tasks Q_i, or total CPU cycle counts for task completions. Figure 4 illustrates the different phases of SAGE pipelines and their corresponding delays modeled in this work's solution. This study aims to execute mobile tasks locally while offloading computations to ES_s to minimize task duration.

D. DELAY MODEL

Apps on mobile face queueing delays and lengthy queuing times, resulting in bad user experiences. Little's Law states that queue lengths determine queuing times. Hence, this work aims at minimizing queuing backlogs and MEC congestions. The proposed DESO (Dynamic Edge Server Offloading) algorithm splits and distributes executions amongst resources for optimizations.

E. MATHEMATICAL BASIS

The computational processes can be delegated to edge clouds or performed locally, and these concepts of MEC, local

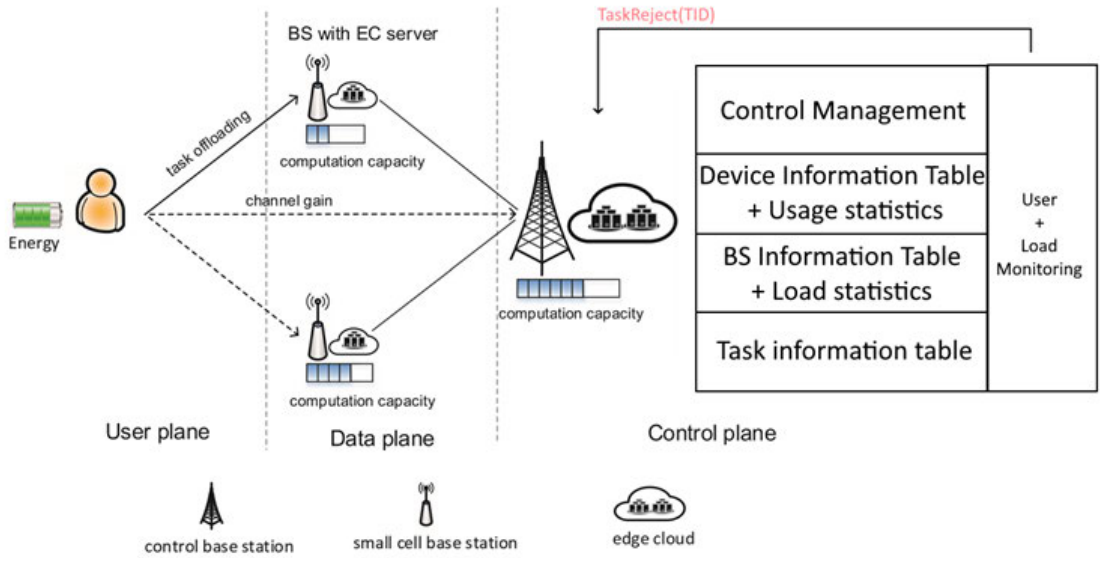


FIGURE 2. Proposed TCO architecture.

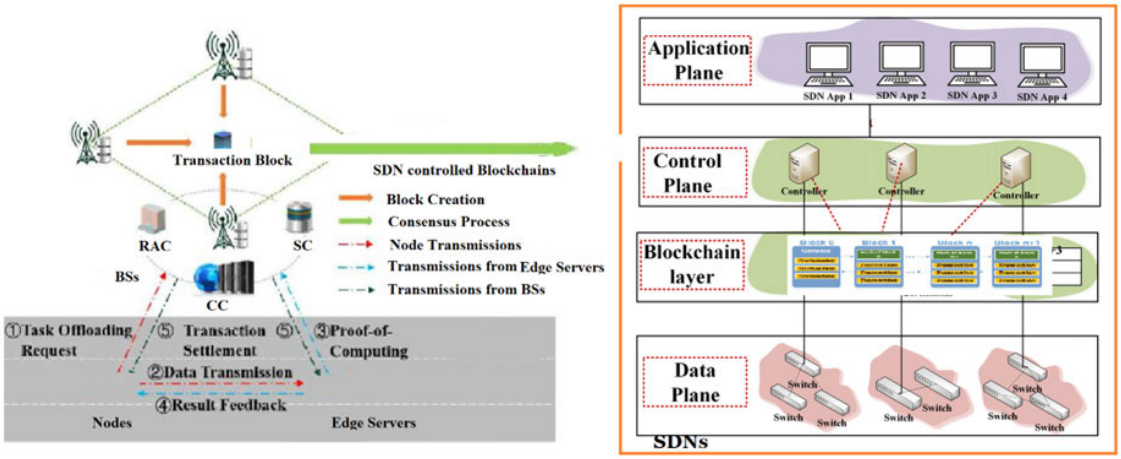


FIGURE 3. Blockchain-based edge service proposed computing architecture.

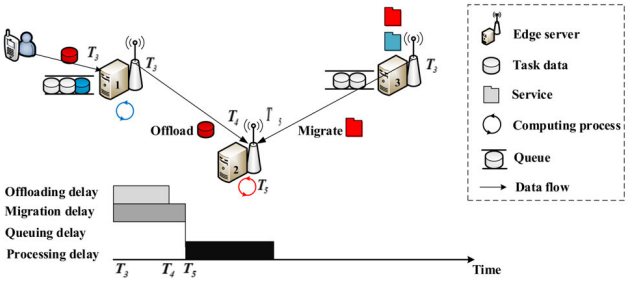


FIGURE 4. SAGE pipeline delay.

computing, models, and parameters used in the offload scheme are detailed below:

- User planes encompass users who need computing work.
- Data planes encompass small cell BSs and edge clouds.
- The control plane includes an SDN controller installed in macro cell BSs.
- VMs in ES are instantiated as instances with heterogeneous computing capacities
- Active VM instances measure resource utilizations in VM pools.

We also assume that VMm refers to counts of VM instances in the m-th ES_s e_m.

The set of mobile edge clouds is represented as $\beta = \{b_1, b_2, \ldots, b_n\}$. Users connected to networks are represented as $\mu = \{\mu_1, \mu_2, \ldots, \mu_n\}$. The set of BSs is represented as:$\alpha_{u_i} = \{\alpha_1, \alpha_2, \ldots \alpha_n\}$, where BSs b_j are elements of ($b_2 \epsilon \alpha_{u_i}$). Since interference and other parameters are needed for computing communications gains, channel gains (h_{ij}) between users (μ_i) and and BSs (b_j) are considered, assuming task offloads happen in low mobility.

Data rates,

$$r_{ij} = B\, log_2 \left(1 + \frac{p_i h_{ij}}{I_{ij} + \sigma^2}\right),$$

where σ^2 is the noise power of mobile devices,

B is channel bandwidth,

I_{ij} is inter-cell interference power,

p_i is the transmission power of user μ_i.

When nodes receive results outside specific delays, it implies task offload failures. Considering transmissions of tasks from mobiles/devices to ESs, we get:

Transmission Delays

$$t_{ij}^T = \frac{S_i}{r_{ij}},$$

where t_{ij}^T are transmission delays in offloading tasks from users μ_i to servers b_j. S_i implies data sizes of tasks needing offloads. Transmission Energy Consumptions $E_{ij} = \frac{p_i S_i}{r_{ij}}$ where assuming two scenarios

1) The tasks are executed locally; their execution time becomes $t_i = \frac{w_i}{f_i}$.
2) Tasks are executed remotely by offloading them to edge cloud servers' execution times becomes t = $\frac{w_i}{K_i^{Ci} f_i} + \frac{S_i}{r_{ic_i}}$

and E_{ij} represents energies consumed by mobiles in offloading tasks from users u_i to servers b_j and since the processing is done remotely, and energy consumption of mobile device/user equipment becomes $E_i = p_i w_i$

F. DESIGN OVERVIEW

It is typical for 5G users to offload delay-sensitive jobs to ESs for quicker task executions. This work's algorithmic offload selections are ideal for multi-tasking environments as their outcomes are best offloading decisions in minimal completion times. The strategies use multiple ESs in design where servers quickly decide on migrations. They also consider signal interferences caused by user migrations/ relocations in their decisions. The EFP approach is also enhanced by minimizing work lengths to handle edge computing offloads in multi-user contexts and limitations of battery energies. The offloading strategy has two stages, namely

1) deciding on task execution types local (based on remaining mobile battery power) or ESs (selecting available ESs) and
2) managing allocations of resources, which are further divided into two minor problems and 0-1 assignment of tasks

Different optimizations are used for the placement of tasks and resource allocations in task offloads based on [39]. Security in task offloads can be realized by ensuring the elements detailed below:

- **Privacy**: Recommended mechanisms should safeguard the privacy of nodes, preventing attackers from determining their genuine identities.
- **Fairness**: Mechanisms should force fairness, like rewarding successful task offloads and ESs that return findings within specific periods.
- **Defense** : against "double-claim" attacks: Malicious nodes need to be identified and punished using smart contracts, which also need to penalize malicious ESs engaged in "free-ride" assaults.
- **Protection against repudiation attacks**: Guarding BSs against false claims by nodes, as BSs cannot refuse reimbursements to honest nodes on successful offload executions. The job offloads can be dynamically optimized based on observed queue times, handover costs, and dependability of ESs with three parameters listed below:

1) QUEUING-DELAY AWARENESS

Significant weights, F (t), are added to queuing delays to encourage nodes to offload more data. This factor is obtained when queuing delays deviate significantly from their corresponding needs.

2) HANDOVER-COST AWARENESS

Nodes are compelled to minimize handover intervals between ESs and are granted significant weights, Z(t). This factor is obtained when handover costs deviate significantly from corresponding requirements.

3) TRUSTFULNESS AWARENESS

This factor is possible by nodes' propensity to choose highly trustful ESs. The recommended strategy should allow BSs to learn and select optimum task-offloading approaches with minimal delays while handling information. The offload costs are formulated by combining computational cost with queue lengths, a complex issue while unloading in dynamic contexts. The unloading costs and queue lengths should be balanced to reduce MEC offload costs, as it may result in overworked or unstable MEC. Moreover, as device counts increase, offloading complexity also rises. This work proposes a dynamic offloading method, DESO, for ESs divides optimization issues into sub-issues where offload decisions of time slots solve the issues efficiently. DESO algorithm is an offloading method (Figure 5) in which each sub-problem of the job to be offloaded is delivered to one MEC server linked to BSs within the users' communication range. The task to be offloaded is modeled as a dynamic programming problem and divided into sub-problems. By maximizing work size and job complexity, DESO offers a practical response to the issue of task dumping to the edge and allocation over a sustained duration rather than an instantaneous burst of time. This is achieved by also utilizing reinforcement learning, as detailed in the following steps. Figure 6 shows the RL algorithm. The RL algorithm is integrated with the SAGE framework, utilizing blockchain-based smart contracts and SDN-based dynamic computation offloading and migration, as specified in the original text. This integrated framework leverages the RL-based contextual bandits' approach and the DESO

algorithm to optimize task offloading in 5G environments. It aims to reduce task offloading delays improve energy efficiency, and other performance objectives specified in the reward function. Task offloading to ESs in 5G is our goal. Device load, network circumstances, and available ESs should be considered to minimize task offloading delays and energy consumption. Considering factors like CPU, RAM, Disc space etc., [44] while choosing an optimal and efficient ES is one of the key essentials. First, we use the Q-learning update rule to teach the agent the value of a certain action in a state. To update the Q-value for a state-action combination, it combines the current Q-value, observed reward, learning rate (α), and discount factor (γ).

$$Q(s,a) = (1-\alpha).Q(s,a) + \alpha.\Big(r + \gamma.\max_{a}\big(Q\left(s',a\right)\big)\Big),$$

where Q(s, a) is the Q - value for state s and action a

α is the learning rate parameter: we can set $\alpha = 0.1$,

r is the observed reward for taking action a in state s

γ is the discount factor parameter: we can set $\gamma = 0.9$

s' is the next state after taking action

$\max_{a}\big(Q\left(s',a\right)\big)$ represents the maximum Q-value for the next state.

The agent chooses actions using the epsilon-greedy policy after this. It balances exploration (random activities) and exploitation (highest Q-value actions).

$$\pi(a|s|) = \frac{\epsilon}{|A|} + 1 - \epsilon, \; if \; a^* = arg\max_{a}\big(Q\left(s',a\right)\big),$$
$$\pi(a|s|) = \frac{\epsilon}{|A|} \; otherwise,$$

where

$\pi(a|s|)$: Probability of selected action a in state s

a^*: Action with the highest Q-value in state s

ϵ: Exploration factor, probability of exploration (e.g., $\epsilon = 0.1$)

$|A|$: Total number of possible actions Finally, the DESO task offloading algorithm will then determine the most suitable MEC (Multi-Access Edge Computing) server for the offloading task. The optimizations are divided into slots denoted by T = $\{1, \ldots, t, \ldots, T\}$. Task data generated by nodes A(t) initially enter local buffers and subsequently offloaded to N ESs within nodes' communication ranges indicated by S = $\{s_1, \ldots, s_n, \ldots, s_N\}$. Task offloading approaches are represented by binary indications $x_{n,t}$, where $x_{n,t} = 1$ indicates that ESs n are selected in nth slots and $x_{n,t} = 0$ otherwise.

G. IMPLEMENTATION OF SAGE DYNAMIC SECURE AND INTELLIGENT OFFLOADING FRAMEWORK

The implementation involves system configurations, smart contract developments, and blockchain creations. Figure 7 is a diagrammatic overview of the major operations of the SAGE pipeline.

Algorithm Dynamic Edge Server Offloading (DESO)Method

Definitions

In time slot t, the number of tasks for the ith application needing to be offloaded is denoted by $A_i(t)$

For any slot t and any $i \in$ I, it holds that $\sum_{j\in J} a_{ij}(t) = A_i(t)$.

queue length is determined by $Q_{ij}(t)$.

Input

t	// timeslot
I	// application
Q	// Queue length
MEC	// MEC Edge Server

Output

Edge Server Offloading decision ESn for a task t

// Run this algorithm periodically on the Edge Controller

for all applications i **do**

Determine the variable $e_i(t)$, by solving the integer programming problem for optimal offloading.

$$\min_{e_i(t)} \; Q_{ij}\text{-}(t)e_i(t) + W[\varphi_i(e_i(t),t) + \phi_i(A_i(t) - e_i(t),t)]$$

for all the server j ∈ {1,2,⋯ ,n} **do**

Get the index j^* with the minimum value of $Q_{ij}(t)$,

i.e., $j^* \in argmin_{j\in\{1,2,\cdots,n\}} Q_{ij}(t)$.

Set $a_{ij}(t)$ according to

$$a_{ij}(t) = \begin{cases} e_i(t), & j = j^* \\ 0, & otherwise \end{cases}$$

end for

Set $a_{i(n+1)}(t) = A_i(t) - e_i(t)$

end for

FIGURE 5. Dynamic task offloading decision.

1) SYSTEM AUTHENTICATIONS

These authentications are based on keys and certificates. BSs first produce their public and private keys (Kp, Ks) used in encryptions. Nodes also create their respective key pairs (Kp, Ks) and (Knp, Kns), where Kp is sent to BSs, which encrypts nodes' profiles using Ks resulting in unique signatures for f (Sig Node necessary for node communications to BSs). RAC then combines nodes' public keys with digital signatures for generating certificates. SCs store these certifications, and anonymous blockchain network nodes obtain legal identities.

2) REQUESTS FOR TASK OFFLOADS

Node task offloads to CCs with max delays. On receiving offloading requests, smart contracts assure fairness and security in offloads [40]. CCs send node signatures to RAC, which transmits them to nodes for authenticity. Migration requests may originate from nodes or contract controllers, which dynamically offload tasks to ESs.

3) TASK OFFLOAD DECISIONS

DESO dynamically offloads computing jobs and decides on ESs considered by controllers at BSs. DESO uses

```
Algorithm Reinforcement Learning

Initialization
  Initialize_parameters()
  Initialize_q_table()
Input
    Q(s, a) is the Q-value for state s and action a.
    α Is the learning rate parameter: we can set α=0.1
    r Is the observed reward for taking actionα in state s
    γ Is the discount factor parameter: γ=0.9
    s^' is the next state after taking action
Output
    Offload or Non-offload

Reset the environment for a new episode
        For episode in range(num_episodes):
        Reset_environment()
While not termination condition:
        Step 1: Get the Current State
        Current_state= get_state()

        Step 2: Select Action
        Chosen_action= epsilon_greedy_policy(current_state)
        if chosen_action == 'offload':

        Step 3: Task Offloading using the DESO Algorithm
        Selected_mec_server_deso_offload(current_task)
        offload_task(current_task, selected_mec_server)
        action_result = (chosen_action, selected_mec_server)
        Else:
        Action_result=chosen_action
end while
```

FIGURE 6. Reinforcement learning algorithm.

parameter W to balance offload costs and performances and respond to changing external environments. Reducing offload costs occurs while limiting queue lengths.

4) DATA TRANSMISSIONS

Nodes encrypt data and send them using their certificate's public keys. The task offloads are verified by constructing Merkle hash trees as leaf nodes.

5) DATA COMPUTATIONS

Encrypted task data are decrypted using Kn and begin computations. Merkle hash root values (m) are computed using task data and computational results comparable to [41].

6) FEEDBACK ON RESULTS

Task data size ratios are considered constant inside slots, like [40] by t. Selected ESs Sn, must transfer data back to nodes after computations, and hence, results are transmitted to nodes through BSs where τ represents resulting delays. Nodes communicate Root (m2) to CCs, which compares Root (m1) obtained from nodes to Root (m2) obtained from nodes. When these two are similar, it implies ESs are misleading CCs in data computations by directly leveraging offloaded data to construct Root (m2). If nodes fail to send Root (m2) to CCs before the specified time, smart contracts instantly conclude transactions and directly conduct Case 2 of transaction settlements otherwise, they assume nodes have completed computations, and findings are encrypted and sent back to nodes using their certificate's public keys.

7) BLOCKCHAIN CONSTRUCTIONS

SDN Controllers record transactions as blocks and upload them to blockchains whose heads have difficulty, timestamps, the last block's hash value, and the Block's Merkle hash root value. Authorized BSs deduce their proof-of-work by computing block hash values with random numbers and information, including dates. This discovered proofs-of-work is broadcast to other BSs for verifications and earn rewards. Blocks added become permanent when most BSs accept proof-of-work.

8) EDGE SERVER TRUSTFULNESS ASSESSMENTS

The Trustworthiness of ESs is assessed using a subjective logic framework [42] based on beliefs and connection probabilities. Trustworthiness of nodes, represented as opinion vectors $\pi_{n,t} = \{\mu_{n,t}, \mu^{\iota}_{n,t} \cdots uc_{n,t}\}$, here $\mu_{n,t}$, $\mu^{\iota}_{n,t}$ and $uc_{n,t}$ denote beliefs, disbeliefs, and uncertainties where $\mu_{n,t} + \mu^{\iota}_{n,t} + uc_{n,t} = 1$. The uncertainties $c_{n,t}$ are connection probabilities. The data saved in local buffers of nodes are established in initialization phases when data queues backlog. Selection indicators and queue backlogs are set to 0. Available ESs during decisions are selected at least once. The suggested architecture for safe and intelligent task offloads is not restrictive and can be extended to cellular-based vehicular networks or Narrow-band IoT in unique signatures for f (Sig Node necessary for node communications to BSs). RAC then combines nodes' public keys with digital signatures for generating certificates. SCs store these certifications, and anonymous blockchain network nodes obtain legal identities.

IV. PERFORMANCE EVALUATION

We performed several simulations to gain an in-depth understanding of the components that make up a complete 5G network and build a testing environment for applications utilizing 5G networks. The experiments of this work mimic network environments with large numbers of concurrently available ESs and user devices distributed randomly around these Es, which can handle multiple sub-tasks based on their processing capabilities as user devices split tasks into sub-tasks.

Hardware Configuration:

- CPU: Intel® Core™ i7-6700HQ
- RAM: 16GB
- GPU: NVidia GeForce 960M
- OS: Ubuntu 16.04.6 LTS 64-bit

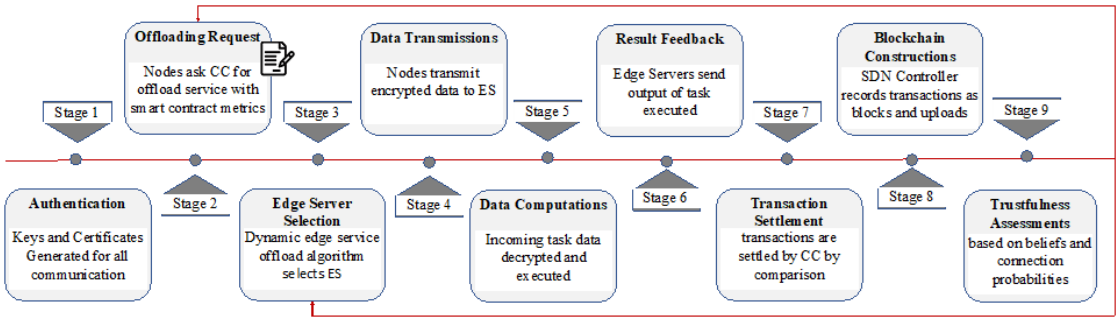


FIGURE 7. SAGE framework's operational flow.

Software Used:

- ns-3: Discrete-event simulation generator for Internet-related protocols
- CloudSim: This is a framework for modeling and simulating cloud infrastructures
- openLEON: Emulator that builds on Mininet
- srsLTE to build MEC-compliant simulations

A. PERFORMANCE ANALYSIS OF THE LEARNING ALGORITHM

This section analyzes the suggested RL algorithm's performance in choosing the offloading server in the dynamic task offloading scheme (DESO) using simulation.

1) CONVERGENCE ANALYSIS

Convergence performance in machine learning is a critical metric that measures the rate at which an algorithm approaches a stable or optimal solution during the iterative refinement process. It signifies the point at which further iterations yield diminishing returns, indicating that the solution has nearly stabilized. Achieving convergence is fundamental in training models effectively, as it ensures that the algorithm converges within a reasonable time frame. In dynamic and diverse environments, prioritizing convergence over specific learning rates and batch size values proves to be a prudent approach. The algorithm gains adaptability across various datasets and problem landscapes by emphasizing convergence. This flexibility allows the model to effectively navigate different scenarios without being tethered to rigid hyper parameter values, a crucial quality in real-world applications. Moreover, a convergence-centric approach bolsters the robustness of the system, particularly in dynamic settings like 5G-enabled MEC systems. Prioritizing convergence also mitigates the risk of overfitting a specific dataset's hyper parameters like learning rate and batch size. Instead, it encourages the development of a more generalized and versatile model that can perform well across diverse scenarios. Convergence-oriented strategies strike a balance between speed and accuracy. While specific values of learning rate and batch size are important considerations, they should be chosen in the context of achieving convergence.

TABLE 2. Binary decision counts vs actual value counts.

Actual Values	Predicted Values	
	0	1
0	T0	F1
1	F0	T1

This ensures that progress is neither overly slow nor erratic, striking an equilibrium that allows for both efficiency and effectiveness in model training. Figure 8 illustrates the rewards of the system in the proposed blockchain-enabled M2M communications networks at various learning rates. In the context of Deep Reinforcement Learning (DRL), the learning rate corresponds to the scale at which the network parameters are tweaked based on the gradient of the loss function. To explain simply, a higher learning rate corresponds to a wider range of parameter updates. The system incentivizes consistent performance by utilizing a reduced learning rate, as it possesses the ability to accurately determine the optimal value's exact position. Furthermore, a higher learning rate facilitates superior convergence

2) ALGORITHM ACCURACY

Accuracy testing is a crucial metric in machine learning, assessing the proportion of right predictions made by a model. It is particularly relevant in classification tasks, quantifying the model's correctness in its classification decisions. For a binary classification task, the confusion matrix includes:

- True Positive (TP): Correctly predicted positives (e.g., correctly offloaded tasks)
- True Negative (TN): Correctly predicted negatives (e.g., correctly retained tasks locally)
- False Positive (FP): Incorrectly predicted positives (e.g., tasks incorrectly offloaded)
- False Negative (FN): Incorrectly predicted negatives (e.g., tasks incorrectly retained locally).

The accuracy is expressed mathematically as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The accuracy score is computed using the confusion matrix. This matrix is very advantageous when addressing

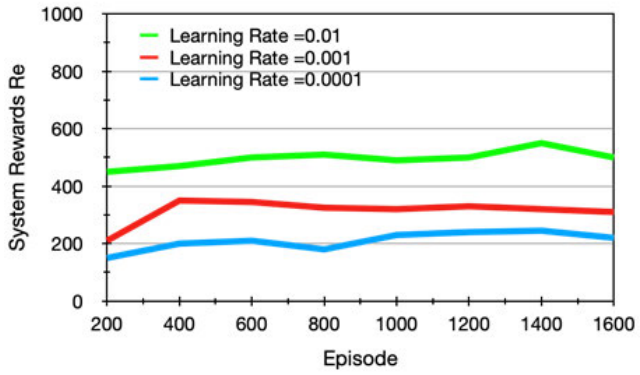


FIGURE 8. System rewards under different learning rates.

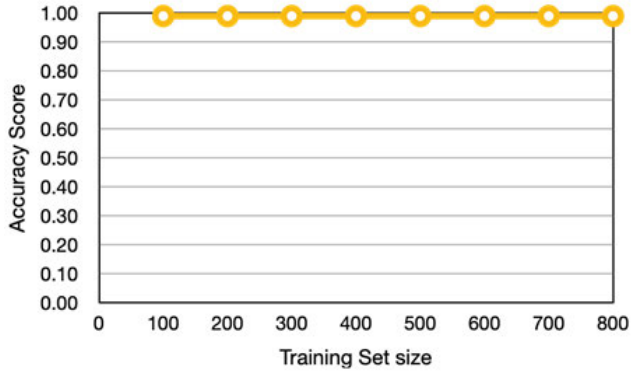


FIGURE 9. Cross-validation values of RL algorithm.

categorization challenges. For our research problem focused on task offloading in 5G-enabled MEC systems with integrated RL and DESO algorithms, accuracy is a pivotal metric for several reasons. Accuracy directly evaluates the precision of the offloading decision-making process. It measures the system's proficiency in correctly allocating tasks to the optimal destination, a critical aspect for efficient and responsive offloading strategies.

Generalization and Validation: A model's accuracy reveals how effectively it can generalize to new data. Evaluating the model's performance on a validation or testing dataset ensures it can make accurate offloading decisions in real-world scenarios. Comparative Analysis and Optimization: Using accuracy facilitates meaningful comparisons between different iterations or versions of the offloading model. This aids in identifying the most effective configuration for accurate task allocation.

Feedback Loop for Continuous Learning: In the integrated RL component, accuracy is a crucial feedback signal for training the agent. It allows the reinforcement learning algorithm to adjust its policy based on the accuracy of its offloading decisions, enabling continuous learning and improvement. In the RL-based offloading decision-making process, the agent uses accuracy as a feedback signal for policy updates. Assuming that at each step (t), the agent selects an action at (task offloading decision) and receives a reward rt based on the accuracy of the decision:

$$r_t = f\ (\text{Accuracy}\ (a_t))$$

where f(.) is a function that maps the accuracy to a reward value. The RL agent's objective is to gain the most anticipated cumulative reward over time:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

where π is the policy, and γ is the discount factor. Balancing Trade-offs for Efficiency: While accuracy is primary, it can be balanced with critical considerations like latency, energy consumption, and fairness in task allocation must also be considered. A multi objective optimization problem can be formulated as:

$$\begin{aligned}\text{Minimize: } L = {} & w_1 \cdot \text{Accuracy} + w_2 \cdot \text{Latency} \\ & + w_3 \cdot \text{Energy Consumption} + w_4 \cdot \text{Fairness}\end{aligned}$$

where (w_1, w_2, w_3) and w_4 are weights representing the importance of each metric. Other metrics that can be derived from the confusion matrix to further evaluate the performance include:

- Precision (Positive Predictive Value):

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall (True Positive Rate or Sensitivity):

$$\text{Recall} = \frac{TP}{TP + FN}$$

- F1-Score (Harmonic mean of Precision and Recall):

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Beyond accuracy, these metrics provide a whole picture of the model's performance, which is especially useful in cases when there is a potential for class imbalance. These equations help quantify the effectiveness of task offloading decisions, allowing for an analysis of the performance across various dimensions, including accuracy, latency, energy consumption, and fairness. This ensures that the offloading system meets the specific requirements and constraints of the dynamic 5G-enabled MEC environment. Table 2 displays the number of binary decisions and the corresponding counts of real values. There exist four distinct categories of output:

- T0 denotes the value of true as 0
- T1 denotes the value of true as 1
- F0 denotes the value of false as 0
- F1 denotes the value of false as 1

The values 0 and 1 correspond to the choices of performing the task locally or offloading the task, respectively.

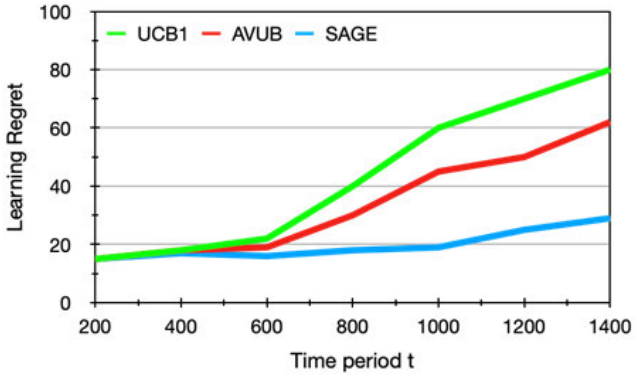


FIGURE 10. Comparative regret values.

TABLE 3. Comparative accuracies of RL algorithms for both divisions of the dataset.

Algorithm	Train-test Split	Cross Validation
ET	0.971	0.966
AB	0.90	0.908
GB	0.964	0.965
XGB	0.964	0.972
LGB	0.972	0.979
Proposed Algorithm	0.976	0.985

TABLE 4. F1 score.

Algorithm	F1 Score
ET	0.82
AB	0.84
GB	0.85
XGB	0.87
LGB	0.88
Proposed Algorithm	0.92

Figure 9 depicts the results achieved by the application of the cross-validation technique on the RL algorithm. By repeatedly dividing the dataset into training and validation sets, cross-validation assesses the model's performance. The dataset is partitioned into k subsets, or folds, in k-fold cross validation. The model is trained k times, with each training iteration utilizing $k - 1$ fold and the remaining fold for validation. The overall accuracy can be computed as the mean of the accuracies across all folds:

$$Accuracy_{CV} = \frac{1}{k}\sum_{i=1}^{k} Accuracy_i,$$

where $Accuracy_i$ is the accuracy obtained in the i-th folder. We showcase the efficacy of our supervised learning model by evaluating its performance using both train-test split and cross-validation procedures on a generated dataset. The data is inputted into several classification algorithms. A comparison of the RL algorithms' accuracy scores for both divisions of the dataset is depicted in Table 3. Table 4 shows the F1 score for the various algorithms included the proposed one.

3) LEARNING RATE

Learning regret, a crucial metric in reinforcement learning, quantifies the sub-optimality of an agent's decisions compared to an oracle with perfect knowledge of the environment. It represents the difference between the agent's actual cumulative rewards and what could have been achieved with ideal decision-making. Learning regret proves invaluable in the context of our task offloading challenge within 5G-enabled MEC systems. It quantitatively assesses how well the reinforcement learning agent performs compared to an idealized oracle. This metric is instrumental in fine-tuning the agent's offloading policies for more effective decision-making. Figure 10 shows the learning regret values of the proposed SAGE algorithm with the already available UCB1 and AVUB algorithms under diverse occurrence times of SeVs and identical loads.

4) COMPUTATIONAL COMPLEXITY ANALYSIS

The DESO algorithm employs a dynamic programming-based task offloading approach integrated with reinforcement learning (RL) to optimize the assignment of tasks to mobile edge computing (MEC) servers. The computational complexity of DESO depends on the offloading decision-making process, learning updates, and queue management mechanisms. The DESO framework solves the offloading decision problem using dynamic programming, which typically has a complexity of $O(N \times T)$ where:

- N is the number of edge servers,
- T is the number of time slots considered

This makes DESO computationally feasible for real-time offloading as long as N and T are moderate. The RL-based decision-making in DESO employs a Q-learning process. The update rule for the Q-table follows $O(S \times A)$ where:

- S is the number of system states (e.g., available resources, network conditions),
- A is the action space (offloading decisions)

For large-scale 5G edge computing, the state-action space grows exponentially, requiring function approximation techniques (e.g., DRL) for scalability. DESO integrates queuing delay awareness into the decision-making process. Based on Little's Law, queue-related computations follow $O(N \log N)$ complexity for sorting and scheduling tasks among multiple servers. Compared to DRL - based offloading, DESO has a lower computational overhead as it avoids deep neural networks' training complexities. Compared to Federated Learning, Federated Learning requires multiple iterations of local training and global aggregation, making it $O(K \times M \times D)$ where:

- K is the number of communication rounds,
- M is the number of participating devices,
- D is the dataset size per device.

To summarize, in the best case scenario, If offloading decisions are computed efficiently, DESO runs in $O(N \log N)$ time. In the worst-case scenario, with a large state space,

TABLE 5. DESO vs. drl (deep reinforcement learning-based offloading).

Feature	DESO (Proposed algorithm)	DRL-Based Offloading
Computational Complexity	O(N log N) (fast)	O(S × A × T) (higher complexity)
Training Overhead	Low (No deep learning required)	High (Neural network training)
Real-time Feasibility	High (DP and Q-learning scalable)	Lower (Deep models require training)
Energy Consumption	Lower (fewer computations)	Higher (NN training overhead)
Performance Trade-off	Good for low-latency offloading	Better for long-term optimization

TABLE 6. DESO vs. federated learning (FL) for task offloading.

Feature	DESO (Proposed algorithm)	Federated Learning (FL)
Is Training Required?	No (Uses reinforcement learning)	Yes (Distributed training per node)
Communication Overhead	Low (Only offloading decisions transmitted)	High (FL requires iterative model updates)
Privacy Considerations	Moderate (Tasks are offloaded to edge servers)	High (No raw data sharing)
Computation Latency	Low (Task decisions made in real-time)	Higher (Training rounds cause delay)
Energy Efficiency	Higher (Less processing on devices)	Lower (Continuous model training overhead)

the Q-learning update process may require O(S × A × T), impacting real-time deployment where:

- S is the number of system states (e.g., available resources, network conditions),
- A is the action space (offloading decisions),
- T is the number of time slots considered

Table 5 describes the comparative analysis with the state of the art approaches between DESO (proposed) and Deep Reinforcement learning-based offloading. Likewise, Table 6 shows the comparison between DESO and FL for task offloading.

To conclude, our proposed algorithm DESO is more suitable for real-time, low-latency applications, while Deep Reinforcement learning-based offloading approaches are effective when long-term optimization and deep learning models can be trained offline. DRL provides better long-term performance but at a higher computational cost. If privacy is the primary concern, FL is preferred over DESO, but at the cost of higher latency and energy consumption. If the goal is real-time task offloading with low overhead and moderate privacy, DESO outperforms DRL and FL approaches.

B. PERFORMANCE OF THE DESO SCHEME

The benefits of the recommended SAGE algorithm are illustrated through a comparison. These closely related edge computing offloading techniques are considered (re-implemented based on the description of their algorithms in the corresponding publications) for comparison testing as follows:

- FCFS (First-come-First Service): For execution, each subtask within a task is transferred to the edge server, as opposed to being executed locally. After being carried out, the user device is subsequently notified of the task's execution structure in the order in which it was executed.
- CEFO [35] is a centrally controlled edge computing offloading technique based on SDWN. The SDWN selects the appropriate migration method for each job based on the task data provided by each user device. For any offloading system, the combined graph of user tasks assigned to the same server is considered an integrated directed acyclic graph (DAG). Which offloading approach results in the shortest waiting time is used to decide the dispatching strategy
- p-MEFP [43] is the first algorithm with the shortest time-to-completion when multiple users are present in a resource contention environment.

1) TASK QUEUEING OVERHEAD

This section contrasts the typical task delay for each strategy considering the Benchmark, CEFO, and p-MEFP simulation experiment under various resource contention circumstances. The network environment is set to three different levels: high contention (two ESs, each one can execute only one task), medium contention (five ESs, each one can execute up to two tasks), and low contention (eight ESs, each one can execute up to two tasks). Figure 11 compares the average task queuing delays for various techniques in heavy resource competition. In a case with intense competition for resources, the method's noticeable queuing delay gap becomes more pronounced as the job number grows. The average queuing delay gap reaches maximum of 100 ms for a set of 50 tasks, with the most notable difference being 60 ms for a group of 30 jobs. This demonstrates that the SAGE strategy is highly effective in reducing task waiting delays. The average task delays for the four algorithms for different task counts in a high contention environment are shown in Figure 12 In a highly competitive environment, the availability of ESs that can offer computational resources is limited, and the utilization of ESs for tasks is increasing. The Figure 12 demonstrates that the execution impacts of the four methods are equivalent when there are few jobs, with the SAGE method being just slightly superior.As the number of concurrent tasks grows, the waiting time contributes an ever-increasing proportion to the total delay, and the issue of users fighting for ESs gets ever more acute. The difference between the methods in terms of mean task delay widens as the difference in queuing

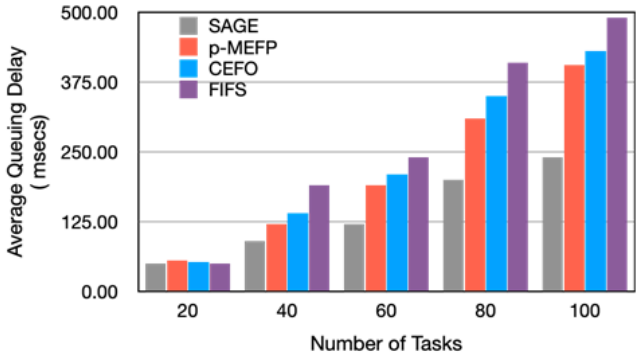


FIGURE 11. Average queuing delays.

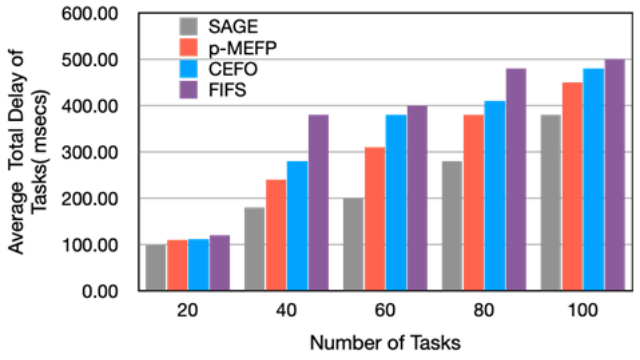


FIGURE 12. Average total delay.

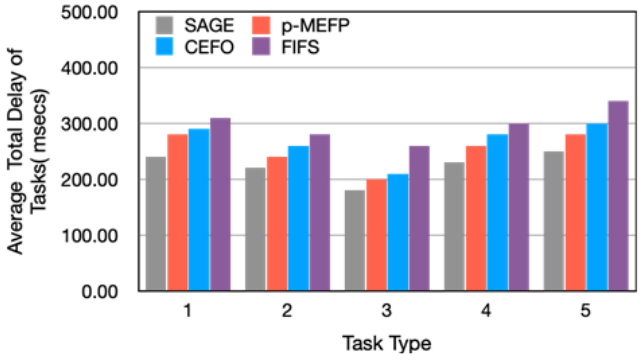


FIGURE 13. Average delays vs. offloaded task types.

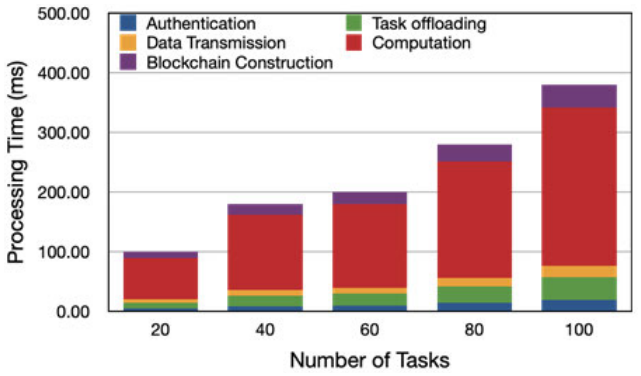


FIGURE 14. Overhead of stages in the pipeline.

delay widens. The difference between the best and worst performance, when there are 20 jobs, is only 10 milliseconds. The difference is 40 milliseconds when there are 40 jobs, but when there are 100 tasks, the average delay in the SAGE pipeline is 80 milliseconds and around 70 milliseconds lESs than CEFO. It significantly reduces execution latency and is very effective at reducing delays. Contrarily, the performance of SAGE increases with the number of users, proving its effectiveness in lowering task delay in a situation with resource competition. Nevertheless, the rate of delay escalates rapidly as the number of tasks grows. Compute offloading, data caching, and four various sensing services (such as encryption/decryption, video streaming, alert messages, temperature, and sensing for wearable devices) are the job categories the ICPs may do in the reference scenario. Each has a different execution time, required latency, packet size (i.e., for both requests and answer messages), and the interval between two consecutive requests. All four methodologies are employed to calculate and transfer a set of tasks using 20 distinct directed acyclic networks within the same network environment. The obtained average task delays are compared in Figure 13. The job of task type 1 has the most significant task execution delay, as shown in Figure 13, while task type 3 has a significantly better task execution impact than other task types. Because its five sub-tasks may only be completed in sequence, task category 1 has the longest execution delay. For task type 3, a maximum of three sub-tasks may be carried out continuously in parallel. By moving parallel processing workloads to many edge computing servers, the three subtasks can be computed simultaneously. Experimental results show that task parallelism affects optimization clarity following compute offloading and time savings.

2) PIPELINE STAGES OVERHEAD

We evaluate our proposed SAGE with additional computation workloads, N = 60, 70, and 150, respectively, and compare the overall computation overhead with those by adopting various offloading techniques, such as local computing by all mobile devices (MD), in our experiments. Additionally, this series of tests shows that the offloading method using multiple MEC servers can improve performance by an average of 60% compared to the method using a single MEC server, proving the value of using multiple MEC servers in 5G Edge/Fog networks. This pattern scales effectively as the number of computing jobs increases. Three main stages comprise a task offloading process:

- Choosing a server
- Building a blockchain or smart contract
- Transmitting task data, computing, receiving and validating the results

Figure 14 displays the system's overall end-to-end performance and the time required for each major stage.

3) SCALING WITH EDGE SERVER COUNTS

This section compares the total processing times of four algorithms—benchmark FIFS, CEFO, p-MEFP, and SAGE—across a range of edge server numbers (see Figure 15). Although SAGE provides certain advantages over the other techniques in terms of optimization, which can reduce the average task execution latency, these advantages eventually fade as the number of ESs increases. Overall, SAGE is still successful at lowering latency, and it is evident that the number of ESs significantly impacts how quickly tasks are executed. The continuous incorporation of ESs results in a decreased average task execution latency to one-third of its original value.

4) LOAD BALANCING OVERHEAD

ES clusters have limited resources, so the computational resources of ESs are represented by a collection of virtual machines (VMs). When there are more tasks sent to the ES than the VMs can handle, some tasks have to wait in a queue until the ES finishes processing the previous tasks. There is a delay in the waiting queue. Efficient load balancing is crucial for evaluating the effectiveness of task offloading. The condition of the ES cluster improves significantly with a lower value. According to the mathematical basis, the load balancing may be calculated as:

$$LB(st) = \frac{1}{UK} \cdot \sum_{w=1}^{W} \left[UV\left(st_{n,i}\right) - UC\left(st_{n,i}\right) \right]^2$$

Moreover, when the number of user applications or users grows, the ultimate load-balancing values of the three techniques are about equal. The reason for this is that each ES is in a state of equilibrium, with all of its resources fully utilized. The SAGE architecture is highly adaptable to various situations.

5) AVERAGE RESOURCE UTILIZATION

An important benchmark to use when evaluating the effectiveness of the ESs is the average resource utilization. The number of running virtual machine instances in an ES's VM pool is a proxy for resource use. The VM instances become occupied once all compute applications have been transferred to ESs utilizing the offloading strategy. The VM pool has fewer unused VM instances due to higher resource utilization. The average resource utilization is calculated as

$$ARU(st) = \frac{1}{EE} \sum_{m=1}^{M} C_m$$

where EE denotes the number of actively ''Employed Edge'' Servers, and Cm is the resource utilization for each mobile device m. Figure 18 shows the comparative findings for the average resource utilization of the four approaches discussed above for various MDs and applications. Figure 19 compares average resource utilization for these three methods at various application scales. As shown, the SAGE framework outperforms the other approaches in all the scenarios. The key reason is that for a range of application sizes, in comparison to the other methods, SAGE analyzes a variety of hyperparameters to assign applications more sensibly and enhance average resource utilization.

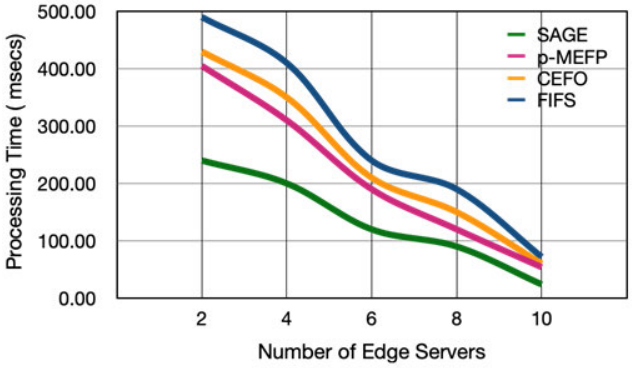


FIGURE 15. Processing time vs. edge server counts.

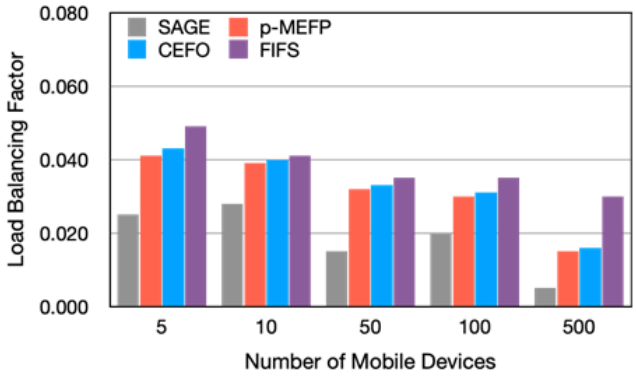


FIGURE 16. Load balancing across the ESs clusters for mobile devices.

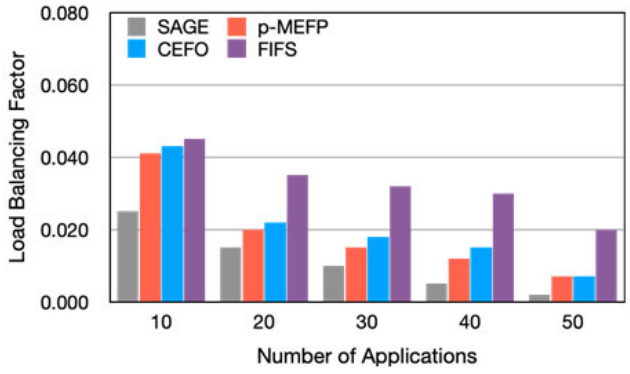


FIGURE 17. Load balancing across the ESs clusters for applications.

6) BLOCKCHAIN TRANSACTION PERFORMANCE

The unit of gas in Ethereum reflects the amount of computational work performed. Figure 20 illustrates the use of gas consumption and transactions. Transactions result in a rise in the consumption of gas. This approach resulted in a

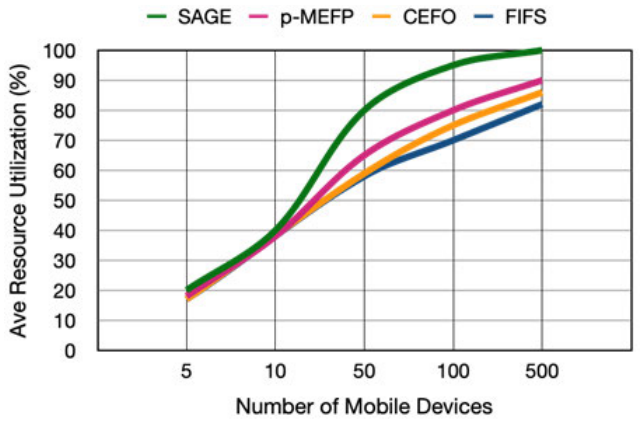


FIGURE 18. Comparative findings for average resource utilizations of approaches with mobile devices.

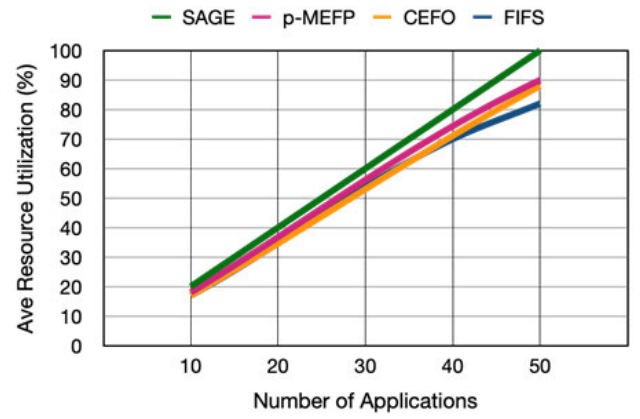


FIGURE 19. Comparative findings for average resource utilizations of approaches with applications.

30% increase in transaction throughput and an 85% reduction in transaction time. For transactions under 200, the amount of gas consumed and the processing time are similar (up to 27 seconds). With the increase in the transaction quantity, the gas consumption amount also increases linearly, however, the time to process remains unchanged. The scalability of our system is achieved by using the shorter processing time of an SDN controller compared to the gas required for a Blockchain transaction. Our proposal integrates the use of Blockchain technology to ensure both great safety and efficiency

In this benchmark, the blockchain consortium maintains a 100% success rate for permitted transactions from incoming transaction rates of 0 to 500. After that, the success rates quickly declined to almost nothing, showing that both blockchains have topped 500 TPS. However, for both test systems, the transaction latency remained consistent between 15 and 22 seconds during the evaluation time. At a rate of [100, 1000] transactions per second (TPS), 1000 transactions are being put into the blockchain.

C. SECURITY ANALYSIS

This section will analyze numerous attack scenarios and their security. We will conclude by using real-world attack

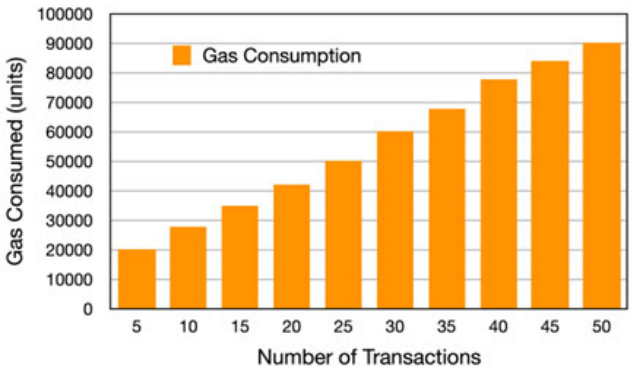


FIGURE 20. Blockchain energy consumption.

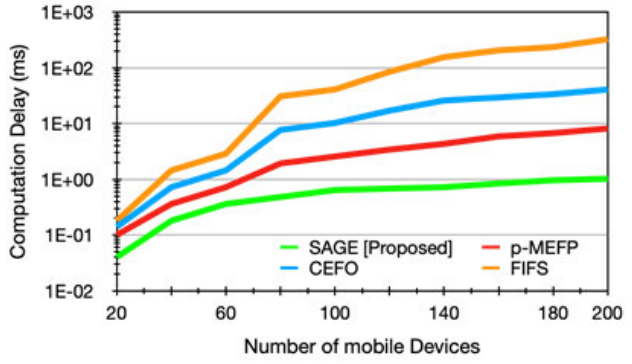


FIGURE 21. Computation delays vs. no. of devices.

scenarios to demonstrate how our proposed SAGE system protects endpoints and network infrastructure.

1) SECURITY PROPERTIES AND ATTACK STUDIES

This section provides the findings of a practical assessment of the security capabilities of SAGE. We presented case examples that demonstrate such functionalities in operation. To demonstrate the vulnerability of a device that is either in the process of being supplied or has already been provisioned, we conducted real-time assault scenarios. These case examples illustrate how an adversary might quickly compromise the device. Pre-condition and post-condition constraints, however, shield our SAGE system from these attacks.

Case 1—"Device sending a Single Malicious Packet": By using stolen login credentials, a person can access a device without authorization and attack its gateways. To lessen this attack, an identity and verification system is required. Case 2—"Device sending several Malicious Packets": In this scenario, fake data is used to take control of the IoT gateway via a gadget. This attack evaluates authentication mechanisms' legitimacy.

Case 3—"Compromised Device Injecting Malware": Access to the smart network is restricted to devices that have been pre-authenticated and provided beforehand. For example, malware such as Mirai infiltrates devices. We will

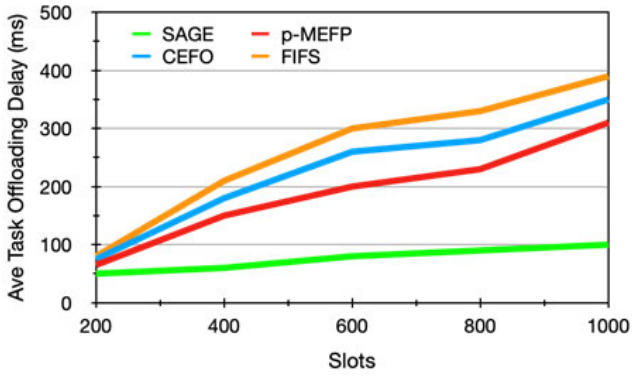


FIGURE 22. Offloading delays based on trust.

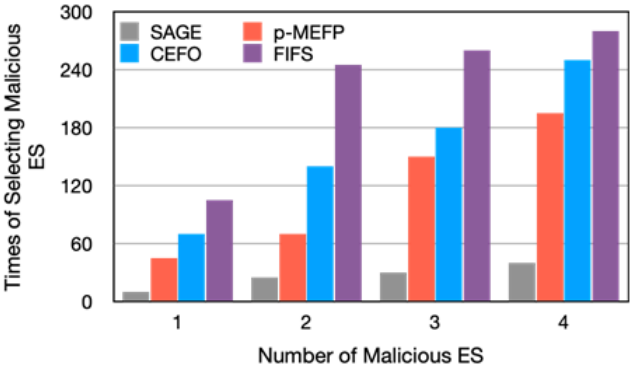


FIGURE 23. Malicious device counts vs. selection probabilities.

orchestrate such attacks to thoroughly test the effectiveness of our permission policies.

Case 4—"State Change of a Device Due to External Manipulation": The SAGE analyses events and situations to monitor the device. This test case is designed to verify the real-time effectiveness of SDN Security Monitoring. Here, we will carry out attacks on the network's connected devices, such as mobile phones, to cause changes in the state of those devices. Authorization policies face challenges in detecting these state shifts. The event-driven status monitoring services of our SDN controller quickly identify these changes.

2) RESILIENCE TO MALICIOUS 5G DEVICES

Malicious mobile devices significantly affect the communication systems' functionality and security. The ability of SAGE to differentiate between authentic and malicious devices is of utmost importance. The SAGE security controller cannot carry out device re-authentication until it has received all of the messages in the series. Our authentication technique allows us to greatly reduce the computational and communication needs for identifying malicious users of devices. This test compares the computational expenses of the standard approach and SAGE in the scenario of malicious cellular devices. The vertical axis of Figure 21 utilizes a log scale to represent computational latency for simplicity

3) MALICIOUS EDGE SERVERS

Figure 22 depicts the correlation between the average delay in task offloading and the number of time slots when one or more malicious ESs attempt to disrupt the offloading process within the framework. SAGE outperforms other schemes by reducing the average task offloading delay by 18%, 27.9%, and 36.5% respectively. This improvement is achieved by taking into account queuing delay awareness, handover-cost awareness, and awareness of trust. As a result, the SAGE DESO algorithm in the controller frequently chooses the detrimental ESs, which decreases the occurrence of delays and failures in job offloading. The FIFS scheme, when compared to other benchmarks, performs poorly. This is because CEFO does not take trustworthiness into account and often selects ESs with malicious tendencies or low connection probabilities. This leads to frequent task offloading failures and significant performance degradation. Figure 23 depicts the median selection times for malicious ESs to the total sample size of malicious ESs. Simulation results demonstrate that when there is only one malicious edge server, SAGE can decrease the selection of malicious ESs by 65%, 85.4%, and 90.2% compared to p-MEFP, CEFO, and FIFS, respectively. Due to its trust awareness, this algorithm consistently outperforms other algorithms even as the count of malicious ESs increases. The SAGE system experiences fewer delays and failures in task offloading because it is highly sensitive to detecting and mitigating malicious ESs.

D. KEY FINDINGS AND RESULTS DISCUSSION

The experiments in our performance evaluation compared the various aspects and key metrics in this problem space with other competitive schemes and benchmarks - computation overhead and running times. It can be observed from Table 7, which summarizes Key Findings from Evaluations, that SAGE may generate a nearly optimum solution, unlike the benchmark. With a dynamic SDN-based algorithm, SAGE outperforms all other schemes regarding computing efficiency and flexibility in re-configuring the process at run time.

V. LIMITATIONS AND FUTURE WORK

Despite the various benefits the proposed architecture provides for secure offloading of tasks in 5G MEC and future applications, there remain unresolved concerns that must be tackled. In spite of the proven success of utilizing integrated methods to improve performance metrics and security posture, it is essential to recognize the limitations and challenges associated with this approach, which are briefly discussed below.

A. CELLULAR TECHNOLOGY

The 5G devices are primarily focused because 5G network offers ultra-low latency, high bandwidth, and massive device connectivity necessary to support real-time and secure task offloading at scale. The dynamic nature of task migration,

TABLE 7. Key findings and results.

Aspects Evaluated	Key Results and Discussions
Queuing Delay Section IV.B 1)	In a situation with intense competition for resources, the method's noticeable queuing delay gap becomes more pronounced as the number of jobs grows. The average disparity in queuing delay reaches 100 ms with 50 tasks, whereas the maximum discrepancy is 60 ms with 30 jobs. This demonstrates the efficacy of the SAGE strategy in reducing task queuing duration.
Total Delay Section IV.B 1)	In a high-contention environment, the number of edge servers that may provide computing resources is relatively small, and the use of edge servers for tasks is more obvious. The total delay of the four methods is equivalent when there are few jobs, with the SAGE method being just slightly superior. The difference between the best and worst performance, when there are 20 jobs, is only 10 ms. The efficacy of SAGE is directly proportional to the number of users, demonstrating its effectiveness in mitigating task delay in a scenario characterized by high resource rivalry. Nevertheless, the rate of delay escalates rapidly as the number of tasks grows.
Delay with Task type Section IV.B 1)	Compute offloading, data caching, and four various sensing services (such as encryption/decryption, video streaming, alert messages, temperature, and sensing for wearable devices) are the job categories. All four strategies are employed to calculate and transfer a set of tasks using twenty distinct directed acyclic networks within the same network environment. The experimental findings demonstrate that the task's degree of parallelism impacts the clarity of the optimization following compute offloading and the time savings.
Stages overhead Section IV.B 2)	We evaluate the system's total end-to-end performance and the time required for the main stages in a task offloading process. This series of tests shows that the offloading method using multiple MEC servers can improve performance by an average of 60% in comparison to the method using a single MEC server (with DESO dynamic migration/offloading scheme in a multi-edge server environment),
Scaling with Edge Servers Section IV.B 3)	This section compares the total processing times of four algorithms—benchmark FIFS, CEFO, p-MEFP, and SAGE— across a range of edge server numbers. Overall, SAGE is still successful at lowering latency, and it is obvious that the number of edge servers greatly impacts how quickly tasks are executed. With the ongoing addition of edge servers, the average task execution latency will be reduced to one-third of what it was.
Load Balancing Section IV.B 4)	This test presents the load balancing results for various numbers of mobile devices and applications. As can be shown, SAGE outperforms the other two approaches in many situations. When the number of user applications or users increases, the final load-balancing values of the three strategies are roughly the same. This is because each ES is balanced, and all its resources are being used.
Resource Utilization Section IV.B 5)	The average resource utilization of the four approaches is discussed for various numbers of MDs and applications. The SAGE framework outperforms the other approaches in all the scenarios. The key reason is that for a range of application sizes, SAGE analyses a variety of hyperparameters to assign applications more sensibly and enhance average resource utilization.
Blockchain Processing Section IV.B 6)	Gas consumption is directly proportional to the number of transactions. Our strategy surpassed existing tactics by up to 30% in overall transaction throughput and enhanced the time interval between transactions by up to 85%.
Security Properties Section IV.C 1)	We conducted these attack case studies in real-time attack scenarios and demonstrate how an adversary can easily compromise the offloading process in the edge network. While performing the offloading processes, SAGE can defend against such attacks using pre/post-condition policies.
Resilience to Malicious devices Section IV.C.2)	The computing delay and offloading costs of SAGE and the conventional method in the case of malicious cellular devices are compared in this test. With our technique, we can significantly minimize the calculation and communication requirements for detecting malicious device users.
Resilience to Malicious Edge servers Section IV.C.3)	The tests illustrate the relationship between the average delay in task offloading and the number of slots, while considering the presence of one or more hostile edge servers that attempt to disrupt the offloading process within the framework. SAGE outperforms comparable methods by reducing the average task offloading delay by 36.5%. This improvement is achieved by taking into account queuing-delay awareness, handover-cost awareness, and truthfulness awareness. The SAGE DESO algorithm in the controller chooses the detrimental Edge servers with lower frequency, hence decreasing the occurrence of delays and failures in work offloading.

combined with the need for trust-aware decisions, requires a network infrastructure capable of handling high-speed and low-delay communications — a feature that 4G and earlier generations cannot reliably guarantee. Moreover, MEC is most effective when tightly integrated with 5G infrastructures, as MEC was designed with 5G architecture in mind to minimize round-trip delays and enhance computational offloading performance. Older generation devices

and networks typically lack native support for MEC and SDN, which are critical to our proposed approach. The underlying principles of the proposed architecture — such as secure offloading using blockchain and adaptive learning — can theoretically be extended to non-5G environments, but would require adjustments to account for limitations in latency, throughput, and edge computing integration. These extensions are considered out of the current study's scope but are identified as potential directions for future research.

B. INTEROPERABILITY

It might be a concern when it comes to 5G cellular (eSIM) technologies, as various manufacturers and providers may employ different approaches, leading to compatibility problems. Efforts should be undertaken to standardize connection and Over-the-Air (OTA) management platforms, as well as the devices that participate in them, to guarantee compatibility.

C. SCALABILITY

The proliferation of connected devices and the ever-increasing data volume make this a formidable task. The architecture's ability to supply, manage, and interface with devices and cloud services is crucial for supporting large-scale deployments.

D. SECURITY

Participating devices are protected by 5G cellular-enabled technology, however they are not immune to attacks. In order to protect against cybercrime, data breaches, and unauthorized access, stringent security measures are required.

E. COST

Smaller firms may struggle to integrate offloading technology in participating devices due to cost.

F. REGULATION

Offloading at the Edge/Fog infrastructure needs more regulation to ensure device participation. Device data security and privacy must be protected by Government and regulatory policies.

G. CLOUD SERVICE VULNERABILITIES

Cloud services consist of complex software components that may have security vulnerabilities. Existing cloud systems need to address the misconception that all distributed cloud services are reliable. Therefore, if a component version in a cloud service is not updated or if there is a misconfiguration, it can allow attackers to spread assaults to other cloud services, thereby endangering the cloud resources of any user. Communications between services, hosts, nodes, and mechanisms are secure and confidential with the SAGE framework. Nevertheless, even if a cloud service is compromised, it might still exhibit malicious behavior or propagate attacks. We will thoroughly analyze the existing security measures in place to protect against vulnerabilities in cloud services and misconfigurations by mobile operators. This analysis will be conducted in a detailed manner with a focus on quality. The suggested framework for safe and intelligent task offloading is not just for 5G MEC using minor adjustments; it may be extended to mobile edge computing (MEC) using Vehicle Fog Servers (VFS) and the hybrid scenario with MEC and VEC. The edge server in the hybrid system may establish its legitimate identity in the anonymous blockchain network by creating a key pair and acquiring a certificate to connect with the cars and the base station. Future research will concentrate on privacy-conscious compute offloading for scientific process applications and gearbox control in MEC-enabled automobile networks. In our forthcoming project, we plan to enable security services to offload tasks in the MEC (Multi-access Edge Computing) setting, where the Edge Servers (ESs) and Mobile Devices (MDs) will need to create their security frameworks. Deep reinforcement learning is a model-free method used to enable job offloading, and computation for D2D communication and meet specified performance criteria.

VI. CONCLUSION

After examining the delay and energy use of edge computing offloading in the 5G network, the research's goal of reducing job delay was set. Employing a delay-aware offloading approach that has been expanded for multi-user situations is suggested. Dividing computational work into several smaller jobs decreases the overall completion time of device applications. Additionally, the algorithm has been adjusted to consider probable resource competition. The efficacy of the proposed strategy has been verified through simulation studies. The results indicate that the recommended work can substantially decrease the overall project delay in comparison to the current work. This study examines the security and energy-efficient collaboration of task offloading in device-to-device (D2D) communications. It introduces a unique security framework for mobile devices. The number of CPU cores, CPU frequency, and data size are all considered in this model. This security paradigm may be used in a heterogeneous D2D situation. The collaborative task offloading problem is then defined to lower MDs' energy and time-average delays while ensuring security. The combined design of communications and computation resource allocations and partial offloading ratios was studied to optimize the TSOD under latency and secrecy constraints. An efficient approach that converges rapidly and outperforms several heuristics was used to tackle this problem. Enough tests and analyses have proven our proposal's efficacy and advantage over the alternatives in various scenarios. Our investigation of combining several emerging technologies in a highly dynamic and advanced framework lays the groundwork for further research in this exciting new area.

VII. STATEMENT OF COMPETING INTERESTS

The authors do not have any conflicting interests relevant to this article.

REFERENCES

[1] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update*, Cisco Systems, INC. San Jose, CA, USA, 2014.

[2] Y. Li, D. Jin, P. Hui, and Z. Han, "Optimal base station scheduling for device-to-device communication underlaying cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 27–40, Jan. 2016.

[3] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[4] A. Ndikumana, S. Ullah, T. LeAnh, N. H. Tran, and C. S. Hong, "Collaborative cache allocation and computation offloading in mobile edge computing," in *Proc. 19th Asia–Pacific Netw. Operations Manage. Symp. (APNOMS)*, Sep. 2017, pp. 366–369.

[5] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.

[6] C. Xu, K. Wang, P. Li, S. Guo, J. Luo, B. Ye, and M. Guo, "Making big data open in edges: A resource-efficient blockchain-based approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 870–882, Apr. 2019.

[7] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

[8] J. Du, C. Jiang, H. Zhang, Y. Ren, and M. Guizani, "Auction design and analysis for SDN-based traffic offloading in hybrid satellite-terrestrial networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2202–2217, Oct. 2018.

[9] Y. Liu, F. Richard Yu, X. Li, H. Ji, and V. C. M. Leung, "Decentralized resource allocation for video transcoding and delivery in blockchain-based system with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11169–11185, Nov. 2019.

[10] Y. Chen, N. Zhang, Y. Zhang, and X. Chen, "Dynamic computation offloading in edge computing for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4242–4251, Jun. 2019.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[12] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between the mobile device and Cloud," in *Proc. 6th Conf. Comput. Syst.*, Apr. 2011, pp. 301–314.

[13] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 1350–1354.

[14] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[15] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," 2017, *arXiv:1705.00704*.

[16] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 695–708, Jan. 2019.

[17] L. Xiao, Y. Ding, D. Jiang, J. Huang, D. Wang, J. Li, and H. Vincent Poor, "A reinforcement learning and blockchain-based trust mechanism for edge networks," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5460–5470, Sep. 2020.

[18] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[19] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.

[20] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11008–11021, Nov. 2018.

[21] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[22] T. DeMarco, *Structured Analysis and System Specification*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1979.

[23] C. Johnson, L. Badger, D. Waltermire, J. Snyder, and C. Skorupka, "Guide to cyber threat information sharing," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. 800-150, 2016, doi: 10.6028/NIST.SP.800-150.

[24] Y. Zhai, T. Bao, L. Zhu, M. Shen, X. Du, and M. Guizani, "Toward reinforcement-learning-based service deployment of 5G mobile edge computing with request-aware scheduling," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 84–91, Feb. 2020.

[25] X. Xu, Q. Huang, Y. Zhang, S. Li, L. Qi, and W. Dou, "An LSH-based offloading method for IoMT services in integrated cloud-edge environment," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 16, no. 3s, pp. 1–19, Oct. 2020.

[26] L. Chen, J. Wu, J. Zhang, H.-N. Dai, X. Long, and M. Yao, "Dependency-aware computation offloading for mobile edge computing with edge-cloud cooperation," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2451–2468, Oct. 2022.

[27] M. Wang, T. Ma, T. Wu, C. Chang, F. Yang, and H. Wang, "Dependency-aware dynamic task scheduling in mobile-edge computing," in *Proc. 16th Int. Conf. Mobility, Sens. Netw. (MSN)*, Tokyo, Japan, Dec. 2020, pp. 785–790.

[28] Y. Liu, S. Wang, Q. Zhao, S. Du, A. Zhou, X. Ma, and F. Yang, "Dependency-aware task scheduling in vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4961–4971, Jun. 2020.

[29] V. Uc-Cetina, N. Navarro-Guerrero, A. Martin-Gonzalez, C. Weber, and S. Wermter, "Survey on reinforcement learning for language processing," *Artif. Intell. Rev.*, vol. 56, no. 2, pp. 1543–1575, Feb. 2023.

[30] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Secure computation offloading in blockchain based IoT networks with deep reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 4, pp. 3192–3208, Oct. 2021.

[31] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.

[32] M. Li, F. R. Yu, P. Si, W. Wu, and Y. Zhang, "Resource optimization for delay-tolerant data in blockchain-enabled IoT with edge computing: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9399–9412, Oct. 2020.

[33] S. Guo, Y. Dai, S. Guo, X. Qiu, and F. Qi, "Blockchain meets edge computing: Stackelberg game and double auction based task offloading for mobile blockchain," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5549–5561, May 2020.

[34] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. M. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703, Mar. 2020.

[35] Z. Li, M. Xu, J. Nie, J. Kang, W. Chen, and S. Xie, "NOMA-enabled cooperative computation offloading for blockchain-empowered Internet of Things: A learning approach," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2364–2378, Feb. 2021.

[36] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.

[37] *What is Ethereum? Ethereum.Org*. Accessed: Sep. 10, 2024. [Online]. Available: https://ethereum.org/en/whatis-ethereum/

[38] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[39] H. Wang, F. Xu, Y. Li, P. Zhang, and D. Jin, "Understanding mobile traffic patterns of large scale cellular towers in urban environment," in *Proc. Internet Meas. Conf.*, Oct. 2015, pp. 225–238.

[40] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[41] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.

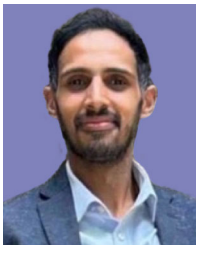
[42] A. Jøsang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *Proc. 29th Australas. Comput. Sci. Conf. (ACSC)*, Jan. 2006, pp. 85–94.

[43] X. Li and B. Ye, "Latency-aware computation offloading for 5G networks in edge computing," *Secur. Commun. Netw.*, vol. 2021, pp. 1–15, Sep. 2021.

[44] A. Marotta, F. D'Andreagiovanni, A. Kassler, and E. Zola, "On the energy cost of robustness for green virtual network function placement in 5G virtualized infrastructures," *Comput. Netw.*, vol. 125, pp. 64–75, Oct. 2017, doi: 10.1016/j.comnet.2017.04.045.
[45] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Barcelona, Spain, Apr. 2018, pp. 1–6, doi: 10.1109/WCNC.2018.8377343.

N. SETHU SUBRAMANIAN (Member, IEEE) received the B.E. degree in computer science from Bharathiar University, Coimbatore, and the M.S. degree in software systems from the Birla Institute of Technology and Science, Pilani (BITS Pilani). He is currently a Research Scholar with the Department of Cyber Security Systems and Networks, Amrita Vishwa Vidyapeetham, India. With a combined experience of more than 22 years in both teaching and industry, he has successfully supervised several research-oriented and application-driven projects, primarily focusing on areas, including health care, e-learning, cyber security, and open-source software tools. His current research interests include cyber security and the IoT.

PRABHAKAR KRISHNAN (Senior Member, IEEE) was a Visiting Adjoint Professor with the Department of Computer Science, The University of Texas at San Antonio, USA. He is currently a Research Scientist with the Department of Cyber Security Systems and Networks, Amrita Vishwa Vidyapeetham, India. He is a member of the Core-Development Group, OpenAirInterface Software Alliance (OSA) 5G Wireless Community, Eurecom, Europe. He has over two decades of industry experience in the USA. His current research interests include cybersecurity, with a special focus on designing network security and architecture, network softwarization, SDN/NFV, cyber forensics, and the IoT standardization. He is a Senior Member of the IEEE Computer Society.



KURUNANDAN JAIN received the bachelor's degree in mathematics from the University of Leicester, in 2011, and the M.Sc. degree in applied mathematics and the Ph.D. degree in mathematical physics from Imperial College London, in 2017. He is currently an Assistant Professor with the Department of Cyber Security Systems and Networks, Amrita Vishwa Vidyapeetham, India. His research interests include cryptography, authentication and privacy design, cryptanalysis, digital signature, quantum cryptography and applications for Industry 4.0, 5G, 6G, and future networks.

K. B. ANEESH KUMAR received the bachelor's degree in physics and the master's degree in computer application from the University of Calicut, Kerala, India, in 2005. His area of expertise includes network management, orchestration and automation, system architecture and engineering, and security architecture and engineering. He joined CDAC, Thiruvananthapuram, a Research and Development Organization of the Ministry of Electronics and Information Technology (MeitY), in 2005, as a Junior Research Fellow. Currently, he is a 'Scientist E' and is associated with the Cyber Security Section, CDAC. His research interests include autonomic networks, software-defined networking (SDN), 5G core, and network function virtualization (NFV). He is a member of (ISC)2 and the SANS Advisory Board. He holds a CISSP certification from (ISC)2 and a GDSA certification from GIAC.

TULIKA PANDEY is currently a Scientist 'G' and the Senior Director of the Cyber Security Research and Development Group, Ministry of Electronics and Information Technology, Government of India. She is an Electronics and Communications Engineer. She received a certificate in an advanced course on cyber defense and cyber security at IIT Kanpur. She has traversed through the path of microelectronics development, E-learning technology development for Indian languages, digital libraries, E-infrastructure, internet governance and convergence, and communications and broadband technologies. Her current pursuit is addressing the security postures, overseeing the research and development initiatives in cybersecurity, and on global discussions on issues of Cyber4 Digital Inclusion, Cyber4Growth, Cyber Security, and Cyber Diplomacy.

RAJKUMAR BUYYA (Fellow, IEEE) is currently a Redmond Barry Distinguished Professor and the Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, The University of Melbourne, Australia. He also serves as the Founding CEO of Manjrasoft, a spin-off company of the university, commercializing its innovations in cloud computing. He has authored several publications and textbooks, including *Mastering Cloud Computing* (McGraw Hill) and international markets. He is one of the highly cited authors in computer science and software engineering worldwide (H-index of 37, g-index of 304, and more than 100 000 citations).