Received June 27, 2020, accepted July 14, 2020, date of publication August 12, 2020, date of current version August 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3016046

Computation Offloading for Distributed Mobile Edge Computing Network: A Multiobjective Approach

FARHAN SUFYAN, (Graduate Student Member, IEEE), AND AMIT BANERJEE, (Member, IEEE)
Department of Computer Science, South Asian University, New Delhi 110021, India

Corresponding author: Amit Banerjee (amit@cs.sau.ac.in)

ABSTRACT Mobile edge computing (MEC) is emerging as a cornerstone technology to address the conflict between resource-constrained smart devices (SDs) and the ever-increasing computational demands of the mobile applications. MEC enables the SDs to offload computational-intensive tasks to the nearby edge nodes for providing better quality-of-services (QoS). The recently proposed offloading strategies, mainly consider a centralized approach for a limited number of SDs. However, with the growing popularity of the SDs, these offloading models may have the scalability issue and can be susceptible to single point failure. Although there are few distributed offloading models in the literature, they ignore the vast computational resources of the cloud, load sharing between the MEC servers, and other optimization parameters. Toward this end, we propose an efficient computation offloading scheme for a distributed load sharing MEC network in cooperation with cloud computing to enhance the capabilities of the SDs. We formulate a nonlinear multiobjective optimization problem by applying queuing theory to model the execution delay, energy consumption, and payment cost for using edge and cloud services. To solve the formulated problem, we propose a stochastic gradient descent (SGD) algorithm based solution approach to jointly optimize the offloading probability and transmission power of the SDs for finding an optimal trade-off between energy consumption, execution delay, and cost of the SDs. Finally, we perform extensive simulations to demonstrate the effectiveness of the proposed offloading scheme. Moreover, compared to the other solutions, the proposed scheme is scalable and outperforms the existing schemes.

INDEX TERMS Computation offloading, Internet of Things (IoT), mobile edge computing (MEC), queuing theory, smart devices (SDs).

I. INTRODUCTION

A. BACKGROUND

Recent advancements in mobile technology and high data-rate wireless networks have surged the innovation of new smart devices, including smartphones, fitness bands, connected cars, and smart-watches. These devices are also collectively known as *mobile devices* [1], *smart mobile devices* [2] or *Internet of Things (IoT)* devices [3]; however we refer them as *smart devices* (SDs). SDs can collect data from the surroundings and take collective decisions by sharing data with other devices. The connected network of SDs forms an intelligent ecosystem of knowledge, which changes the perception of modern lifestyle and revolutionizes a new industrial age, commonly known as *Industry 4.0* [4]. With the advent of IoT, many novel applications and services emerged and gained enormous popularity. With increasing demand, the IoT devices are becoming more resourceful in terms of processing capability, memory, and battery. However, such advancements are not sufficient for supporting computation-intensive or delay-sensitive applications, such as health-care and autonomous driving, virtual, and augmented reality [5], [6]. Thus, providing such services on the SDs is a challenging task.

Conventionally, the mobile cloud computing (MCC) platform is used to meet the soaring demand of these emerging applications. SDs can *offload* the computational-intensive tasks to a remote centralized cloud server for maintaining

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

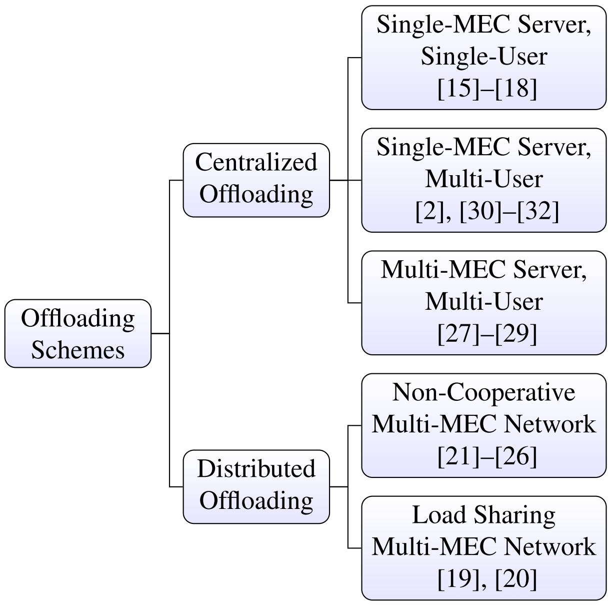

Offloading Schemes
Centralized Offloading
Distributed Offloading
Single-MEC Server, Single-User [15]–[18]
Single-MEC Server, Multi-User [2], [30]–[32]
Multi-MEC Server, Multi-User [27]–[29]
Non-Cooperative Multi-MEC Network [21]–[26]
Load Sharing Multi-MEC Network [19], [20]

FIGURE 1. **Broad categorization of offloading schemes.**

the required quality of services (QoS) of an application. Although the cloud infrastructure can provide ample storage and computational resources for the IoT applications, due to high communication latency, it may not be suitable for the delay-sensitive applications [7]. Moreover, in the era of big data, delivering a large amount of data from SDs to the cloud for processing can cause a heavy link burden on the core network [8]. Hence, it is efficient to bring computation and storage closer to the SDs. For this, *Mobile Edge Computing (MEC)* [9], also known as *Multi-Access Edge Computing (MEC)* [3] or *Fog Computing* [10], has emerged as a key enabler for supporting the real-time processing for IoT applications. MEC is a small-scale cloud platform that provides computing resources and storage services at the edge of the radio access networks in the proximity of SDs [11]. With the emergence of different computational platforms, the complexity of offloading increases, as the offloading algorithm needs to make an efficient trade-off between various decision problems, like *why, what, when, where*, and *how*, based on the current application requirements [12]. Recently, the problem of utilizing the MEC network for computation offloading has gained many research efforts. However, efficient computation offloading is still a challenging problem.

B. RELATED WORKS

The computation offloading problem is also referred to as *cyber foraging* [13] or *remote execution* [14]. In general, researchers have proposed various centralized and distributed frameworks for studying the offloading problem in the MEC platform. Further classification can be done on the number of users, the number of MEC servers, and the sharing of available resources between the MEC servers, as shown in Fig. 1. In the following section, we present a brief account of the related scholarly works.

1) SINGLE-MEC SERVER AND SINGLE-USER SCENARIO

Recently, authors have proposed various models for the single SD and a single MEC scenario to optimize the offloading decision problems [15]–[18]. In [15], the authors propose a 1-D search algorithm to optimize tasks scheduling on the MEC server for minimizing the execution delay. Similarly, in [16], [17], authors jointly optimize the task offloading decision, task scheduling, and power allocation for a single-user MEC system with multiple independent tasks to minimize the weighted sum of execution delay and energy consumption. A two-step approach based on transparent computing (TC) is proposed in [18], to minimize the energy requirements of the IoT device by offloading delay constrained tasks to the TC storage server. The first step analyzes the code block, including the data associated with the task to determine if offloading is required. The second step schedules the offloading tasks for further minimizing the energy consumption of the IoT device.

2) SINGLE-MEC SERVER AND MULTI-USER SCENARIO

In addition to the above, researchers have studied the offloading problem for the multi-user scenario, i.e., multiple users sharing resources of a single-MEC server. For example, Zhang *et al.* [2] propose an iterative search algorithm that jointly optimizes the local computing resources, channel allocation, and transmission power for obtaining an optimal trade-off between energy consumption and execution latency. In [30], the authors formulate the energy management problem as a stochastic optimization programming by using the Lyapunov optimization-based online algorithm. Wu *et al.* [31] propose a delay-aware computation offloading algorithm with joint optimization of secrecy-provisioning, computation workload, and radio resource allocation to minimize the overall execution delay. Authors in [32] propose a binary-search water-filling algorithm to optimize the MEC server's available resources by investigating resource allocation policy in a multi-user scenario. In the above works, researchers consider a single MEC server to cater to the incoming offloading requests. Nevertheless, for a large number of SDs, the MEC server is liable to be congested, which may affect the QoS of applications.

3) CENTRALIZED MULTI-MEC NETWORK AND MULTI-USER SCENARIO

Many works consider multi-user and multiple MEC servers for modeling the computation offloading problem. In [27], authors use queuing theory to formulate a multiobjective problem for minimizing the energy consumption, execution delay, and payment cost of the mobile devices. Liu *et al.* [28] consider the social relationships of the mobile devices with energy harvesting (EH) capabilities, to design a game theory-based dynamic computation offloading scheme for the fog computing system to minimize the social group execution cost. The above studies consider homogeneous fog nodes for modeling the computation offloading problem. Whereas, in [29], authors consider a heterogeneous fog network and propose a fair and energy-minimized task offloading (FEMTO) algorithm to offload the tasks from the IoT devices effectively.

4) DISTRIBUTED MULTI-MEC NETWORK WITHOUT LOAD SHARING

In addition to centralized approaches, researchers have studied the offloading problem for the distributed MEC network. However, most of these studies do not consider load sharing among the MEC servers. Guo *et al.* [21], [22] propose an offloading model for ultradense IoT networks, whereby the mobile devices offload its service requests to the edge servers placed at the mobile base stations (MBS) and small cells (SCs). The objective of both papers is to minimize the overall computation overhead, but approaches for achieving the objectives are different. [21], propose a two-tier game-theoretic greedy computation offloading scheme for dynamically changing computational resources at the edge servers, whereas in [22], authors formulate an MINLP and propose an iterative searching based offloading algorithm.

Along with the above frameworks, some studies consider MEC's collaboration with the cloud platform for solving the distributed offloading problem. For example, in [23], the authors consider a collaborative MCC and MEC setup and propose an iterative heuristic algorithm to solve the formulated MINLP problem for reducing the execution latency. Likewise, [25] is a computation offloading scheme for the Maritime mobile cloud network for ships and vessel terminals. The authors propose an improved Hungarian algorithm to select the optimal task execution location for minimizing the energy consumption of the vessel terminals and the execution delay of the computation task. Moreover, in [26], the authors propose a generic architecture of collaborative computation offloading with a centralized cloud to provide support for multi-user multi-MEC scenarios over the hybrid FiWi access network. The authors formulate the constrained optimization problem to minimize mobile devices' energy consumption while satisfying execution time constraints. In addition to the above works, Zhang *et al.* [24] consider a blockchain empowering MEC to solve a joint computation offloading and coin loaning problem, which resembles the cost for executing tasks on remote servers. The problem is formulated as a non-cooperative game to minimize the monetary cost of mobile equipment.

5) DISTRIBUTED MULTI-MEC NETWORK WITH LOAD SHARING

Next, we discuss the articles that consider load sharing in the distributed MEC network, along with the collaboration of cloud infrastructure. In [19], the authors study the computation offloading problem where the fog nodes share the processing load in a distributed manner to minimize the service delay. The architecture of [19] does not consider any centralized entity for distributing the load between the fog nodes. However, it uses the neighboring node information for distributing the tasks in the fog network. Meanwhile, in [20], the authors propose an online approach to select a set of neighboring nodes to form a fog network under uncertainty on the arrival of the fog nodes. The goal is to minimize the computational task latency by optimizing the task distribution in a hybrid fog-cloud network. However, in [19], [20], the authors consider only a single-objective problem of minimizing the execution delay. In this paper, we use the Jackson network for modeling a load-sharing distributed MEC network by formulating a more complex multiobjective optimization problem for optimizing execution delay, energy consumption, and financial cost of the SDs.

Apart from the MEC paradigm, researchers have also proposed the computation offloading model for infrastructure-less distributed mobile stream processing systems [33]–[35]. The system enhances mobile devices' computation capability in highly dynamic and critical scenarios, such as military operations and disaster response. The online system is deployed on a cluster of mobile devices to execute computation-intensive streaming applications that can quickly adapt to the changing environment (fluctuating bandwidth and intermittent connectivity) to achieve high-performance.

C. MOTIVATION

Table-1 summarizes the recently proposed offloading schemes for the MEC platform. It is evident from the table that most of the distributed offloading models do not consider load sharing in the MEC network and ignores the support of the cloud infrastructure from its model. Moreover, these works only consider the objective of energy minimization or execution delay but overlook the optimization of payment cost for executing tasks on the remote servers. The limitations of the existing works, motivate us to study the computation offloading problem in multi-user and distributed load sharing MEC network in collaboration with the cloud, and correspondingly improve the SDs performance. Although the centralized model may be efficient for small IoT networks, while distributed MEC model is more applicable for integrating multiple heterogeneous IoT devices. Distributed MEC model can be more scalable, less vulnerable to single-point failure, and the cost of deployment can be significantly less compared to sophisticated centralized controller-based solutions [19]. The requirements of a distributed offloading model include (a) sharing of load between the MEC nodes, which is essential for better utilization of the available resources in the MEC network. (b) The offloading framework must consider cloud infrastructure resources to support the demand for different applications. (c) The framework also needs to consider the minimization of payment cost for remotely executing tasks on the edge node or cloud along with energy consumption and execution delay.

D. PROBLEM STATEMENT AND CONTRIBUTION

In this paper, we address the computation offloading problem for a large number of SDs in a collaborative MEC network supported by the cloud infrastructure. We assume that

TABLE 1. Summary of recent offloading models in MEC.

References	Offloading System Consideration				Optimization Parameters		
	System Framework	Offloading Platform MCC	Offloading Platform MEC	Resource Sharing	Execution Delay	Energy Consumption	Payment Cost
Mao *et al.* (2017) [16]	Centralized	×	✓	×	✓	✓	×
Shan *et al.* (2019) [18]	Centralized	×	✓	×	✓	✓	×
Liu *et al.* (2018) [28]	Centralized	✓	✓	×	✓	✓	✓
Zhang *et al.* (2019) [29]	Centralized	×	✓	×	×	✓	×
Zhang *et al.* (2018) [2]	Centralized	×	✓	×	✓	✓	×
Wu *et al.* (2019) [31]	Centralized	×	✓	×	✓	×	×
Guo *et al.* (2018) [21]	Distributed	×	✓	×	✓	✓	×
Zhang *et al.* (2019) [24]	Distributed	✓	✓	×	×	×	✓
Yang *et al.* (2019) [25]	Distributed	✓	✓	×	✓	✓	×
Yousefpour *et al.* (2018) [19]	Distributed	✓	✓	✓	✓	×	×
Lee *et al.* (2019) [20]	Distributed	✓	✓	✓	✓	×	×
This paper	Distributed	✓	✓	✓	✓	✓	✓

a service request generated by a SD can be processed locally or offloaded to the MEC server or edge nodes. Offloading introduces an additional communication delay on the execution of the service request and payment cost for utilizing the MEC or cloud servers' resources. Although the local execution of a service request does not incur a communication delay, it can consume considerable battery life. In our model, we assume that the SDs are competing with each other to access the MEC network's resources to maximize their quality of experience (QoE), including battery life. In particular, the significant contributions of this paper are summarized as follows:

- We introduce a generic three-tier distributed offloading framework for an *IoT-edge-cloud* scenario. The bottom layer consists of the SDs that often need to offload their tasks to the edge nodes for enhancing there performance and processing capabilities. Interconnected heterogeneous edge nodes form the middle layer or distributed MEC network. A service request can enter the system through any edge node and be routed independently between the nodes until it is processed or forwarded to the cloud for execution. The top layer is formed by cloud infrastructure, Fig. 2.
- We use queuing theory to simulate the network traffic to model energy consumption, execution delay, and cost of executing the tasks at different network entities. We consider $M/M/1$ queue to model the service request generated by the SDs. The load sharing distributed MEC network is modeled using the open queuing network or *Jackson Network*. $M/M/\infty$ queue is used for the central cloud, which is consistent with the resources of a data center.
- We formulate a constrained nonlinear multiobjective optimization problem to address the computation offloading challenges in IoT-edge-cloud networks. To solve the formulated problem, we use the Stochastic Gradient Descent (SGD) based solution approach combined with the Penalty method. The algorithm jointly optimizes the transmission power and the offloading probability of the SDs to achieve the offloading objectives, i.e., minimizing of energy consumption, processing delay, and payment cost.
- We perform extensive simulation analysis to investigate the performance of the proposed scheme. Simulation results show the offloading framework's effectiveness under different parameter settings for the increasing number of SDs. We also show that the proposed scheme yields better performance in comparison to the other existing solutions.

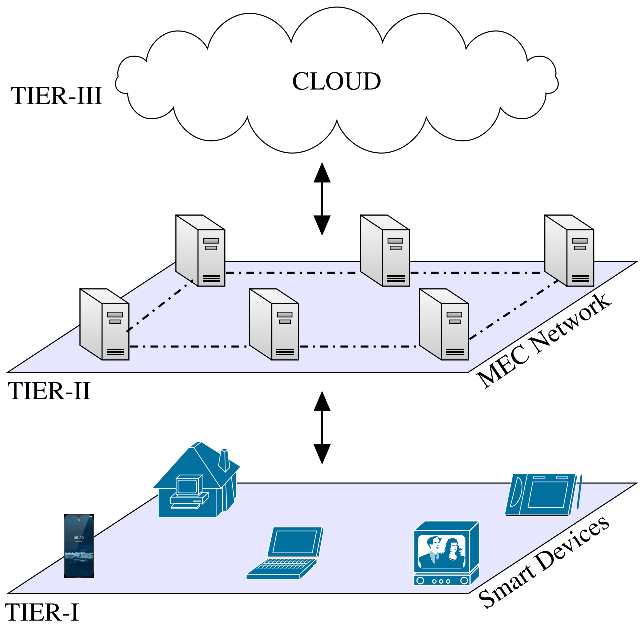


FIGURE 2. IoT-edge-cloud architecture.

E. PAPER OUTLINE

The rest of the article is organized as follows. Section II presents the *IoT-edge-cloud* architecture and formally introduce the analytical model for evaluating energy consumption, service delay, and monetary cost characteristics at different components in the system. In Section III, we formulate the computation offloading as the multiobjective optimization problem and propose an iterative searching-based offloading scheme to solve the formulated problem. Section IV implements the proposed algorithm, and extensive simulation provides the illustrative results to validate our proposed scheme and make comparisons with existing offloading models. Finally, we conclude and discuss future directions in Section V.

II. SYSTEM MODEL

In Fig. 2, we consider an underlying three-tier generic IoT-edge-cloud architecture with the bottom-most layer consisting of resource-constrained SDs, interested in offloading tasks to the middle tier. The middle tier is formed by a distributed network of the edge nodes or MEC servers, providing service to the SDs. Finally, the topmost layer is the cloud infrastructure. The necessary interaction between the three layers are as follows:

- Mobile applications executing on the SDs generate many service requests that are atomic and independent of each other. These requests can be executed either locally or offloaded for the remote execution [17], [26].
- MEC server can process the received request, forward requests to other edge nodes in the MEC network, or send it to the cloud if the load on the MEC network is higher than its processing capacity.

The proposed model's main objective is to determine the amount of work that can be offloaded from the SDs for remote execution on the edge server or the cloud while minimizing the response time, energy consumption, and remote execution cost.

We utilize queuing theory to model our system, which is widely used for resource contention and delay analysis in the computing and communication systems [36]. Queuing theory abstracts the nodes' underlying hardware and avoids the need to decide for an individual offloading task [37]. The frequently used key notations are given in Table 2 along with its corresponding description. In the table, we use the super-script m, e and c to denote the SDs, edge nodes, and cloud, respectively. The sub-script i indicates a specific SD, whereas j and k represents a particular edge node, respectively. Note that the term edge node, MEC node, and MEC server used interchangeably to refer the nodes in the middle tier, i.e., MEC network. In the rest of the paper, we also refer to the incoming service requests as requests, tasks, or jobs interchangeably. We also mention that we have used and applications and users mutually.

A. LOCAL EXECUTION MODEL

We assume that there are N numbers of single processor SDs in the system. The applications running at the SDs generate many service requests that are atomic and independent of each other. The arriving service requests pending for execution on the SD are stored in a first-in-first-out (FIFO) queue. The *Poisson* process is the most common and widely used stochastic process to model the arrival of tasks within a time-span for a single server computing system [38]. Thus, the process queue at the SDs is modeled using the $M/M/1$ queue, which approximates the task arrival process with a Poisson distribution rate of λ_i^m [20], [36], [39]. The service processing rate is *Exponentially distributed* with an average processing rate μ_i^m of an i^{th} SD. The *offloading probability* p_i^m $(0 \leq p_i^m \leq 1)$ impacts the rate at which a SD can offload its service requests for the remote execution. If the offloading probability p_i^m is increased, the service requests are offloaded more frequently, and vice-versa. The *offloading rate* of a SD follows the Poisson process with an average rate of $p_i^m\lambda_i^m$. The requests that are processed locally also follow the Poisson process with an average rate of $(1-p_i^m)\lambda_i^m$, referred to as *local execution rate* [39]. Hence, the average response time (T_i^m) and energy consumption (E_i^m) for executing a request locally is given by [40]:

$$T_i^m(p_i^m) = \frac{1}{\mu_i^m - (1-p_i^m)\lambda_i^m} \tag{1}$$

$$E_i^m(p_i^m) = \eta_i^m T_i^m(p_i^m) = \eta_i^m\left(\frac{1}{\mu_i^m - (1-p_i^m)\lambda_i^m}\right) \tag{2}$$

where, η_i^m is the energy coefficient, representing the per time energy consumption of an i^{th} SD. The parameter η_i^m depends upon the hardware specifications and architecture of the SD [41]. In our implementation, we assume that the value of η_i^m is the same for all SDs during the computation process.

TABLE 2. Frequently used notations in the paper.

Notations	Meanings
N	Number of SDs in the system
λ_i^m	Average request generation at i^{th} SD
μ_i^m	Service rate of i^{th} SD
p_i^m	Offloading probability of i^{th} SD
η_i^m	Local execution power of i^{th} SD
d_i^m	Data size for transmission in each request of i^{th} SD
B	Channel bandwidth
P_i	Transmission power of i^{th} SD
P^{max}	Maximum transmission power of i^{th} SD
g_i	Channel gain between i^{th} SD and the BS
σ^2	Background noise power
M	Number of edge nodes in the system
λ_j^e	Total arrival rate at j^{th} edge node
Λ_j^e	External arrival rate at j^{th} edge node
μ_j^e	Service rate of j^{th} edge node
p_{kj}	Routing Probability of a job moves from edge node k to neighbouring edge node j
λ^c	Total arrival rate of request to the cloud
μ^c	Service rate of the cloud
r^c	Uplink rate of request to the cloud
$\tilde{E}$	Expected energy of SD in the system
$\tilde{T}$	Expected delay of SD in the system
$\tilde{C}$	Expected cost of SD in the system

B. COMMUNICATION MODEL BETWEEN SMART DEVICE AND MEC NETWORK

In our model, we assume the SDs are equipped with a single network interface card (NIC) and single radio access transmission (RAT) antenna for offloading its service requests to a particular edge node at a time [17]. We refer to Shannon's theorem [42] to get wireless uplink data rate r_i^m of an i^{th} device:

$$r_i^m = B\log_2\left(1+\frac{P_i g_i}{\sigma^2+\sum_{j\epsilon N, j\neq i} P_j g_j}\right) \tag{3}$$

where B is the channel bandwidth, g_i is the channel gain between an i^{th} SD and the MEC network, and σ^2 is the background noise power. The power required by the SD for transmitting data to the edge nodes is P_i, $(0 < P_i < P^{max})$. We adopt the wireless interference model [19], whereby many SDs can share the same spectrum resource simultaneously. Thus, the interference caused by other SD's, can be calculated as $\sum_{j\epsilon N, j\neq i} P_j g_j$.

The transmission time for offloading tasks from an i^{th} SD to the edge node can be obtained by dividing the product of *offloading rate* $(p_i^m \lambda_i^m)$ and size of the service request (d_i^m) by the received uplink rate (r_i^m), as shown in (4).

$$T_i^t(p_i^m, P_i) = \frac{p_i^m \lambda_i^m d_i^m}{r_i^m} = \frac{p_i^m \lambda_i^m d_i^m}{B\log_2\left(1+\frac{P_i g_i}{\sigma^2+\sum_{j\epsilon N, j\neq i} P_j g_j}\right)} \tag{4}$$

Similarly, (5) shows the energy required for transmitting the data to the edge nodes, which is calculated by multiplying the power (P_i) required for transmitting the data and the transmission time (T_i^t).

$$E_i^t(p_i^m, P_i) = P_i T_i^t = \frac{P_i p_i^m \lambda_i^m d_i^m}{B\log_2\left(1+\frac{P_i g_i}{\sigma^2+\sum_{j\epsilon N, j\neq i} P_j g_j}\right)} \tag{5}$$

According to *Kleinrock* independence approximation theorem, the total incoming traffic on the MEC network can be approximated as a *Poisson process* [43]. Equation (6) shows the total offloaded service requests load (λ_T^m) from the SDs to the MEC network, which is obtained by adding the individual offloading rates from the SDs connected to the edge nodes.

$$\lambda_T^m = \sum_{i=1}^{N} \lambda_i^m p_i^m \tag{6}$$

C. MOBILE-EDGE COMPUTING MODEL

In our model, the distributed MEC network consists of interconnected edge nodes that interact with each other in an ad-hoc manner to provide computational services to the SDs. We assume that the edge nodes are connected to the primary power grid, so that we can concentrate on the energy consumption of the SDs, as considered in [29], [32]. Further, we also assume that network delay in forwarding the service requests within the MEC network is negligible, as the edge nodes are connected to a backhaul network [21], [22], [26]. The service requests can enter in the MEC network for processing via any edge node. On receiving a service request, the edge node can either execute the task or forward it to another edge node for processing. The services requests are routed between the edge nodes depending upon their resource availability, load on the current MEC server, and total load of the MEC network. Finally, the processed request depart from the system, and the result can be stored in the cloud or sent back to the user.

We use the *Jackson network* to model the distributed resource sharing MEC network, which is an open queuing network consisting of multiple nodes [44]. Previously, [45] used the Jackson network for modeling the cloud architecture with a single entry point. Also, in [46], the authors model the sensor network as an Open Jackson network to balance the computational load, which improves the overall system performance. According to *Jackson's* theorem, if the equilibrium exists at each edge node, i.e., server utilization $\rho_j = \lambda_j/\mu_j < 1$, then the *joint state distribution* for M edge nodes in the system is given by the product of the individual queue equilibrium distributions. This can be represented by the following product-form solution [44].

$$Q(n_1, n_2 \ldots, n_M) = \prod_{j=1}^{M} Q_j(n_j) \tag{7}$$

where Q is the probability of overall system state $(n_1, n_2 \ldots, n_M)$ for M edge nodes, Q_j is the probability that there are n tasks in the queue of edge node $j \in M$.

The total service arrival rate λ_j^e at the j^{th} MEC server $j \in \{1, 2, \ldots, M\}$ is given by (8). In (8), Λ_j^e is the arrival rate of the service request from the SDs. p_{kj} is the routing probability that a job moves from edge node k to the neighboring edge node j and leaves the network with probability $1-\sum_{j=1}^{M} p_{kj}$. Hence, the incoming traffic from the neighboring edge nodes to the j^{th} edge node is $\sum_{k=1}^{M} p_{kj}\lambda_k$. Notice that, for an open network, we must have at least one queue with $\Lambda_j^e \neq 0$ [47].

$$\lambda_j^e = \Lambda_j^e + \sum_{k=1}^{M} p_{kj}\lambda_k \tag{8}$$

Fig. 3 shows an open queuing network with four edge nodes $\{e_1, e_2, e_3, e_4\}$, that are connected with each other to form a load sharing distributed network. The incoming service request from SDs to the edge node is $\{\Lambda_1^e, \Lambda_2^e, \Lambda_3^e, \Lambda_4^e\}$ and the processing service rate of the edge node is $\{\mu_1^e, \mu_2^e, \mu_3^e, \mu_4^e\}$. In the figure, $\{p_{12}, p_{21}, p_{23}, p_{32}, p_{34}, p_{43}\}$ are the associated routing probabilities between the edge nodes. For example, e_2 forwards its traffic load to e_1 and e_3 with routing probabilities p_{21} and p_{23}, respectively. Thus, the traffic load forwarded by e_2 to e_1 and e_3 is $p_{21}\lambda_2^e$ and $p_{23}\lambda_2^e$ and processes the remaining services locally with $(1-p_{21}-p_{23})\lambda_2^e$. The notation λ_2^e shows the total incoming traffic on e_2 from SDs and edge nodes e_1 and e_3, i.e. $(p_{12}\lambda_1^e + p_{32}\lambda_3^e + \Lambda_2^e)$.

In our model, the edge nodes with independent queues form the Jackson network, that receives the incoming service

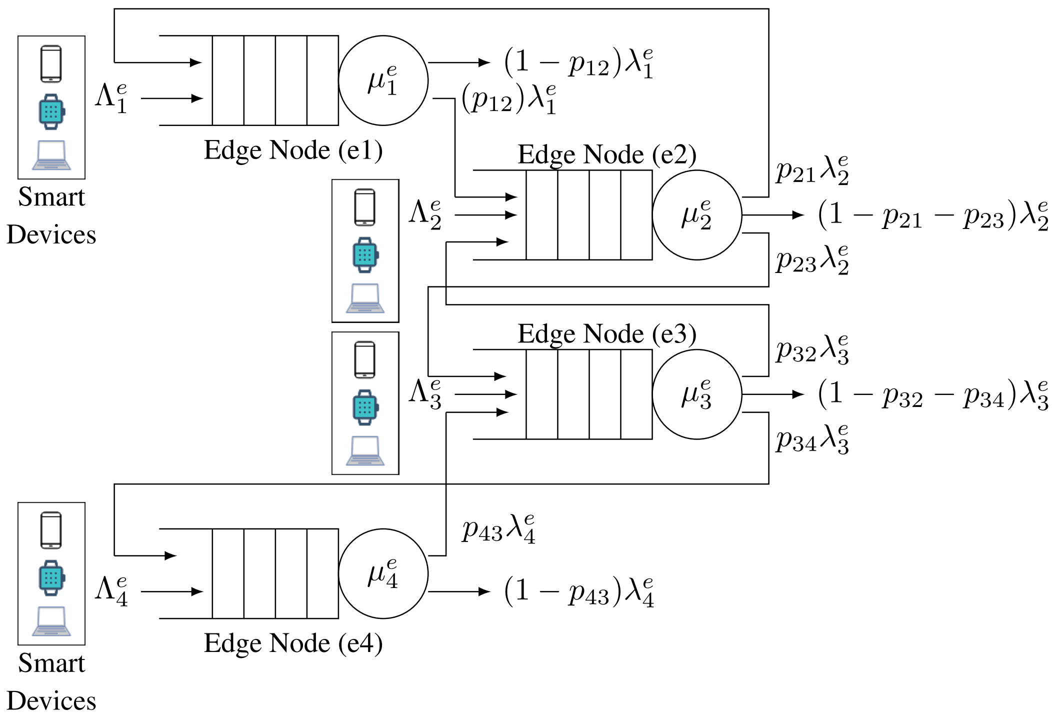


FIGURE 3. Distributed MEC network.

requests following a Poisson's process. In our framework, we assume that the edge nodes are using a single queue for handling its service requests [46]. Hence, we use $M/M/1$ queue for modeling the edge nodes. Thus, the average number of tasks (W_j^e) at the j^{th} edge node can be represented in (9) as:

$$W_j^e = \frac{\lambda_j^e}{\mu_j^e - \lambda_j^e} \tag{9}$$

The MEC network's total processing capacity can be expressed as the summation of the service processing rate (μ_j^e) of the M edge nodes, as shown below:

$$\mu_T^e = \sum_{j=1}^{M} \mu_j^e \tag{10}$$

Notice that, if the total offloaded service requests load (λ_T^m) is less than or equal to the total processing capability (μ_T^e) of the MEC network, then all the service requests can be processed within the MEC network. In such a case, the fraction of requests (ϕ^e) processed by the MEC network is equal to 1, as shown below:

$$\phi^e = 1, \quad if \;\; \mu_T^e \geq \lambda_T^m \tag{11}$$

Correspondingly, the actual processing load on the MEC network (λ_T^e), is given as:

$$\lambda_T^e = \phi^e \lambda_T^m = \lambda_T^m, \quad if \;\; \mu_T^e \geq \lambda_T^m \tag{12}$$

Hence, we can use (12) to evaluate the total service request delay (T^e) of the MEC network, given below in (13):

$$T^e(\lambda_T^e) = \frac{\sum_{j=1}^{M} W_j^e}{\lambda_T^e} \tag{13}$$

D. CLOUD COMPUTING EXECUTION MODEL

When total offloaded service requests load (λ_T^m) is greater than the total processing capability (μ_T^e) of the MEC network. In that case, we utilize the huge computing resources of the cloud infrastructure. Otherwise, the system loses its equilibrium, resulting in system failure. We assume that the edge nodes communicate and share their current processing load with other nodes in the MEC network [48], which can be performed by broadcasting or utilizing the cluster heads after partitioning the network. Equation (14), shows the fraction of requests (ϕ^e) processed by the MEC network:

$$\phi^e = \frac{\mu_T^e}{\lambda_T^m}, \quad if \;\; \mu_T^e < \lambda_T^m \tag{14}$$

Thus, the actual processing load on the distributed MEC network can be expressed as follows:

$$\lambda_T^e = \phi^e \lambda_T^m = \mu_T^e, \quad if \;\; \mu_T^e < \lambda_T^m \tag{15}$$

The overloaded request (λ^c) that exceeds the processing capability of the MEC network is transferred to the cloud for execution. Hence, the total arrival rate of request at the cloud (λ^c) is calculated as follows:

$$\lambda^c = \lambda_T^m - \mu_T^e, \quad if \;\; \mu_T^e < \lambda_T^m \tag{16}$$

In our model, we assume that the MEC network is connected to the cloud over a high-speed network [21]. Thus, the time required for transmitting the service requests to the cloud is given by:

$$T_t^c(\lambda^c) = \frac{\lambda^c d_i^m}{r^c} \tag{17}$$

where r^c is the uplink rate of the requests forwarded by the MEC network to the cloud.

Moreover, we assume a $M/M/\infty$ model at the cloud with a service rate of μ^c, which is consistent with our assumption that the cloud has sufficient resources for processing the incoming service requests from the MEC network [7], [49]. Hence, the queuing delay is negligible, so the total delay at the cloud (T^c) includes only transmission and execution time [40], given by:

$$T^c = T_t^c + \frac{1}{\mu^c} \tag{18}$$

In our model, we ignore the timing overhead for delivering the output data from the MEC or cloud server to the SDs. The two main reasons for this assumption are as follows: (a) the input size is generally much larger than the size of the output, and (b) the downlink data rate is much higher than the uplink data rate. Similar assumptions are also considered by the researchers in their offloading models [23], [50].

E. ENERGY, DELAY AND COST MODEL

The total energy consumption (E_i) for executing a task, either locally or remotely, by an i^{th} SD can be obtained by combining (2) and (5), given in (19):

$$E_i(p_i^m, P_i) = (1 - p_i^m)E_i^m(p_i^m, P_i) + p_i^m E_i^t(p_i^m, P_i) \tag{19}$$

Similarly, (20) shows the total response time (T_i) for executing a request by an i^{th} SD, which is obtained by aggregating (1), (4), (13), and (18):

$$T_i(p_i^m, P_i) = (1 - p_i^m)T_i^m + p_i^m\Big(T_i^t + \phi^e T^e + (1-\phi^e)T^c\Big) \tag{20}$$

Finally, (21) and (22) show the average energy consumption (E) and average response time (T) for executing the service requests generated by all SDs in the system. Moreover, the SDs need to pay for utilizing the resources of the MEC nodes and cloud. If we consider that a unit cost for using the resources at the edge node is C^e and cloud is C^c, then the average payment cost (C) can be expressed as (23). In general, the cost of utilizing cloud resources is higher than the cost of resources in the MEC network, both in terms of transmission delay and monetary cost [51]. Therefore, it is advisable to utilize the MEC resources for computation offloading.

$$E(p_i^m, P_i) = \frac{1}{N}\left\{\sum_{i=1}^{N} E_i(p_i^m, P_i)\right\} \tag{21}$$

$$T(p_i^m, P_i) = \frac{1}{N}\left\{\sum_{i=1}^{N} T_i(p_i^m, P_i)\right\} \tag{22}$$

$$C(p_i^m) = \frac{1}{N}\left\{C^e\lambda_T^m + C^c(\lambda_T^m - \mu_T^e)\right\} \tag{23}$$

III. PROBLEM FORMULATION AND PROPOSED ALGORITHM

As stated previously, the objective of the proposed model is to optimize the offloading probability (p_i^m) and transmission power (P_i) to minimize the execution delay, energy consumption, and monetary cost of the SDs. In the following, we discuss the problem formulation by introducing a multiobjective problem for minimizing the following objective function, shown in (24).

$$\operatorname*{Min}_{p_i^m, P_i}\{E(p_i^m, P_i), T(p_i^m, P_i), C(p_i^m)\} \tag{24}$$

The above multiobjective problem is a nonlinear optimization problem. We use the *Scalarization method* [52] to modify (24) into a linear combination of single-objective problems. The scalers/weights $\{\alpha_1, \alpha_2, \alpha_3\}$, where $\alpha_1 + \alpha_2 + \alpha_3 = 1$, are assigned to the objective functions to reflect their relative importance. The $\tilde{E}$, $\tilde{T}$ and $\tilde{C}$ are the maximum energy consumption, execution delay, and payment cost of the SDs [27]. The modified constrained multiobjective problem is given in (25), where $\varepsilon \ll 1$ is a small positive constant to ensure the stability of the entire queuing system [39].

$$\operatorname*{Min}_{p_i^m, P_i}\left\{\alpha_1\frac{E(p_i^m, P_i)}{\tilde{E}} + \alpha_2\frac{T(p_i^m, P_i)}{\tilde{T}} + \alpha_3\frac{C(p_i^m)}{\tilde{C}}\right\} \tag{25}$$

$$\text{s.t. } (1 - p_i^m)\lambda_i^m \le \mu_i^m - \varepsilon \quad \forall i \in N \tag{26}$$

$$\sum_{i=1}^{M}\lambda_i^m p_i^m \le \sum_{j=1}^{M}\mu_j^e - \varepsilon \quad \forall i \in N, \forall j \in M \tag{27}$$

$$\Lambda_j^e + \sum_{k=1}^{M} p_{kj}\lambda_k \le \mu_j^e - \varepsilon \quad \forall j, k \in M \tag{28}$$

$$0 < P_i < P^{max} \quad \forall i \in N \tag{29}$$

$$0 \le p_i^m \le 1 \quad \forall i \in N \tag{30}$$

Depending upon the total offloaded service arrival rate from SDs to the MEC network, i.e., (λ_T^m), the above problem can further be divided into two sub-cases. First, the total service arrival rate from the SDs (λ_T^m) is less than the MEC network's total processing capability (μ_T^e). That is, if $\lambda_T^m \le \mu_T^e$, then $\phi^e = 1$ (from (11)). Substituting this in (25), the optimization problem reduces to the analytical expression (31), as shown at the bottom of the next page. Next, we consider the case of ($\lambda_T^m > \mu_T^e$), hence, $\phi^e = \mu_T^e/\lambda_T^m$ (from (14)). Again substituting in (25), the optimization problem reduces to (32), as shown at the bottom of the next page. Objective functions L_1 and L_2, given in (31) and (32) have similar constraints from (26) to (30). Constraint (26) is derived from (1), which ensures that the locally executed service request rate is less than the processing rate of the SDs. Constraint (27) restricts the total service requests load from the SDs (λ_T^m), such that it does not exceed the service processing capacity (μ_T^e) of the MEC network, which is derived from (6) and (10). Finally, constraint (28), which is obtained from (8), enforces that the incoming service arrival rate at the individual edge node (λ_j^e) does not exceed its processing capability (μ_j^e).The decision variables in the proposed multiobjective optimization problem (formulated in (31) and (32)) that minimizes the objectives (i.e., energy consumption (E), execution delay (D) and monetary cost (C) under the constraints (26) to (30), are offloading probability (p_i^m) and transmission Power (P_i).

There are several methods for solving constrained optimization problems, such as Penalty and Barrier method, Primal method, Lagrange Multiplier, Frank-Wolfe method, Quadratic Programming [53]. The major challenge in solving the constrained optimization problem is to satisfy the feasible region bounded by the constraints. We use the *Penalty method* that remodels our problem into an unconstrained optimization problem, which converges to the original problem's solution [54]. Thus, a penalty is added to the objective function to enforce obstacles on the boundary of the feasible region, so that an iterative solving process remains within the feasible region. Considering X as the feasible space, the penalty function Φ can be defined as (33), where $f(x)$ is the value of the function at point x in the feasible space X and $g_i(x)$ is the constraint function value [55].

$$\Phi(x) = \begin{cases} f(x) + \sum_i g_i(x), & \textit{if} \;\; x \notin X \\ f(x) & \textit{otherwise} \end{cases} \tag{33}$$

After converting the multiobjective problem into a series of linear combinations of single-objective functions and transforming it into an unconstrained optimization problem, we use a well-known *Stochastic Gradient Descent (SGD)* based solution approach, initially developed by Robbins and Monro [56]. SGD algorithm is a stochastic iterative technique that is frequently used in numerous applications such as machine learning, optimization, classification, neural networks, and deep learning [57].

SGD is an extension of the standard gradient descent (GD) algorithm in which the stochasticity arises at each iteration step by randomly selecting a single training example. To minimize or maximize the objective function in the GD method, the gradient is calculated from the whole training set, thus referred to as *batch gradient descent*. Running the batch GD on a huge dataset with millions of data points is costly because the gradient is calculated from the whole training set at each iteration. However, due to the random shuffling of the training examples, the SGD path to the optimal solution is noisier but much faster because it updates the weights after every iteration. Thus, the SGD is also known as *incremental* or *online gradient descent*. The random shuffling of the training sample has the advantage of escaping the local minima/maxima easily. For the convex function, the SGD takes $\mathcal{O}(1/\epsilon)$ number of iterations to reach the ϵ-neighbourhood of the optimal solution for a sufficiently large number of data samples [58]. Therefore, the convergence of the SGD does not depend upon the size of the dataset. In contrast, the GD requires $\mathcal{O}(n)$ iterations [57], [59] (where n is sufficiently large number of data samples and each iteration requires

$$\begin{aligned}
\underset{p_i^m, P_i}{\text{Min}}\; L_1(p_i^m, P_i) = {} & \alpha_1 \frac{1}{N}\frac{1}{\tilde{E}} \sum_{i=1}^{N} \left\{ (1-p_i^m)\eta_i^m \frac{1}{\mu_i^m - (1-p_i^m)\lambda_i^m} + p_i^m \frac{P_i p_i^m \lambda_i^m d_i^m}{B \log_2\left(1 + \frac{P_i g_i}{\sigma^2 + \sum_{n\epsilon N, n\neq i} P_j g_j}\right)} \right\} \\
& + \alpha_2 \frac{1}{N}\frac{1}{\tilde{T}} \sum_{i=1}^{N} \left\{ (1-p_i^m) \frac{1}{\mu_i^m - (1-p_i^m)\lambda_i^m} + p_i^m \left(\frac{p_i^m \lambda_i^m d_i^m}{B \log_2\left(1 + \frac{P_i g_i}{\sigma^2 + \sum_{j\epsilon N, j\neq i} P_j g_j}\right)} + \frac{\sum_{j=1}^{M} \frac{\lambda_j^e}{\mu_j^e - \lambda_j^e}}{\lambda_T^e} \right) \right\} \\
& + \alpha_3 \frac{1}{N}\frac{1}{\tilde{C}} \left\{ C^e \lambda_T^e \right\} \\
\text{s.t.} \quad & \text{eqs. (26) to (30)}
\end{aligned} \tag{31}$$

$$\begin{aligned}
\underset{p_i^m, P_i}{\text{Min}} L_2(p_i^m, P_i) = {} & \alpha_1 \frac{1}{N}\frac{1}{\tilde{E}} \sum_{i=1}^{N} \left\{ (1-p_i^m)\eta_i^m \frac{1}{\mu_i^m - (1-p_i^m)\lambda_i^m} + p_i^m \frac{P_i p_i^m \lambda_i^m d_i^m}{B \log_2\left(1 + \frac{P_i g_i}{\sigma^2 + \sum_{n\epsilon N, n\neq i} P_j g_j}\right)} \right\} \\
& + \alpha_2 \frac{1}{N}\frac{1}{\tilde{T}} \sum_{i=1}^{N} \left\{ (1-p_i^m) \frac{1}{\mu_i^m - (1-p_i^m)\lambda_i^m} + p_i^m \left[\frac{p_i^m \lambda_i^m d_i^m}{B \log_2\left(1 + \frac{P_i g_i}{\sigma^2 + \sum_{j\epsilon N, j\neq i} P_j g_j}\right)} \right. \right. \\
& \left. \left. + \phi^e \frac{\sum_{j=1}^{M} \frac{\lambda_j^e}{\mu_j^e - \lambda_j^e}}{\lambda_T^e} + (1-\phi^e)\left(\frac{\lambda^c d_i^m}{r^c} + \frac{1}{\mu^c} \right) \right] \right\} \\
& + \alpha_3 \frac{1}{N}\frac{1}{\tilde{C}} \left\{ C^e \lambda_T^e + C^c \left(\sum_{i=1}^{N} \lambda_i^m p_i^m - \lambda_T^e \right) \right\} \\
\text{s.t.} \quad & \text{eqs. (26) to (30)}
\end{aligned} \tag{32}$$

Algorithm 1 Offloading Modules Executing at Different Network Entities

```
Smart Devices (SDs)
1: Initialize: Offloading Prob. ($p_i^m$) and Trans. Power ($P_i$)
2: if ($\lambda_i^m < \mu_i^m$) then
3:     Offload ($p_i^m \lambda_i^m$)
4:     Local Execution ($(1 - p_i^m)\lambda_i^m$)
5: end if

Mobile Edge Computing (MEC) Nodes
1: if ($\lambda_j^e \leq \mu_j^e$) then
2:     Execute Service Request on the MEC node
3: else if ($\lambda_T^m \leq \mu_T^e$) then
4:     Forward request to the neighboring MEC node
5: else
6:     Send service requests to the cloud
7: end if
8: Execute SGD Algorithm

Centralized Cloud
1: Process the incoming service requests ($\lambda^c = \lambda_T^m - \mu_T^e$)
   from MEC network on the cloud servers.
```

Algorithm 2 Proposed Stochastic Gradient Descent (SGD) Based Algorithm

```
Input: $N, \mu_i^m, \eta_i^m, B, \sigma^2, M, \mu_j^e, p_{kj}, \mu^c, r^c$
Output: Optimal values of $p_i^m$ and $P_i$
1: Convert constrained optimization problem ((31) and (32)) into unconstrained optimization problem using
   the Penalty Method and employ the penalty coefficient as discussed in [54].
2: Initialize: nitial feasible point of $\left((p_i^m)^0, (P_i)^0\right)_{i=1}^N$, reduction factor $\beta_j$, iteration $k = 0$, initial value of
   penalty coefficient
3: Define $tol$ as a very small positive real number.
4: while $||((p_i^m)^k, (P_i)^k)_{i=1}^N - ((p_i^m)^{k+1}, (P_i)^{k+1})_{i=1}^N|| > tol$ && $k \neq 100$ do
5:     Calculate Gradient of $L(p_i^m, P_i)$ at $((p_i^m)^k, (P_i)^k)_{i=1}^N$
6:     $(p_i^m)^{(k+1)} = (p_i^m)^k - \beta_j * grad(V(p_i^m, P_i))$
7:     $(P_i)^{(k+1)} = (P_i)^k - \beta_j * grad(V(p_i^m, P_i))$
8:     Replace $(p_i^m)^k = (p_i^m)^{(k+1)}$ and $(P_i)^k = (P_i)^{(k+1)}$
9:     $k = k + 1$
10: end while
11: return Optimal $(p_i^m, P_i)$ to the SDs.
```

n gradient computations) for the same. Hence, solving the optimization problem using SGD is independent of the size of the dataset.

To solve the formulated nonlinear multiobjective optimization problem in (31) and (32), we propose the Algorithm 1 and Algorithm 2. Algorithm 1 shows the execution of the offloading modules on the various networking entities. The proposed optimization problem that minimizes the offloading objectives, i.e., E, D, and C under the constraints, is solved by Algorithm 2. The initial input values of our decision variables, i.e., offloading probability (p_i^m) and transmission Power (P_i), are randomly generated. We provide the input parameters and apply the penalty method to construct an unconstrained optimization problem. Then we apply the iterative approach and calculate the gradient of the objective function to update the value of the (p_i^m, P_i). The reduction factor is a small positive number of $\beta_j \ll 1$, resulting in a faster decline in the penalty coefficient values. Hence, less number of iterations are required to get close to the optimal solution. Finally, the Algorithm 2 returns the optimal value of (p_i^m, P_i) corresponding to E, D and C.

IV. PERFORMANCE EVALUATION

In this section, we investigate the performance of the proposed architecture by studying the effect of different parameters on the offloading objectives and performing the comparative analysis with the existing offloading schemes. For experimental evaluation, we consider a multi-server MEC network with a large number of SDs (from 1 to 100) distributed randomly in a given area. For each SD, the local service rate (μ_i^m) lies within the range of [3.5, 5] MIPS (Million Instructions Per Second) and the service generation rate (λ_i^m) ranges between [1.5, 3.0] MIPS. The size of a service request is randomly generated for each SD by incorporating the *MATLAB rand* function that uniformly generates the request size between [300, 1000] Kb [22], [60]. The energy consumption for local processing at the SDs is set as 3 Joule/sec. We further consider that maximum energy consumption ($\tilde{E}$) and execution delay ($\tilde{D}$) for the SDs is 15 Joule and 2 sec, respectively. The MEC network consists of 6 heterogeneous edge nodes with the processing capability lies between [8, 12] MIPS. The uplink data rate of the optical fiber link connecting the MEC network and the cloud is 1 Gb/s. We use MATLAB simulator for our evaluation, and all results are averaged over a large number of simulation runs. The background noise power (σ^2) is fixed as -100 dBm. Furthermore, we give equal weights to all our offloading objectives, i.e., energy, delay, and cost in the multiobjective optimization problem. Hence, the value assigned to the alpha vector ($\alpha_1, \alpha_2, \alpha_3$) is (1/3, 1/3, 1/3). Finally, we set the maximum payable cost of the SD as 0.2, where the cost of executing a request at the MEC server and the cloud is assumed to be 0.001 and 0.005, respectively. The simulation parameters are taken from [26], [27].

A. EFFECT ON THE OFFLOADING OBJECTIVES

In the following, we investigate the impact of offloading probability (p_i^m), transmission power (p_i^m), and edge nodes M on the offloading objectives (E, D, and C) with an increasing number of SDs. In Fig. 4, we vary the initial offloading probabilities (p_i^m) for the experiment, the initial transmit power (P_i) of all the SDs is set as 10 dBm. Fig. 4a shows that

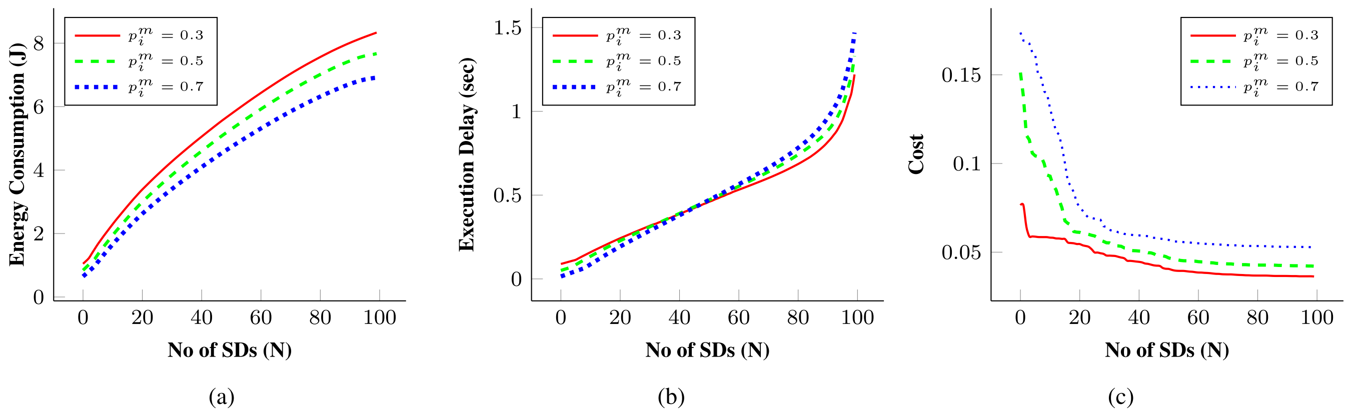


FIGURE 4. Effect of multiple SDs on (a) energy consumption (b) execution delay and (c) payment cost for different offloading probabilities.

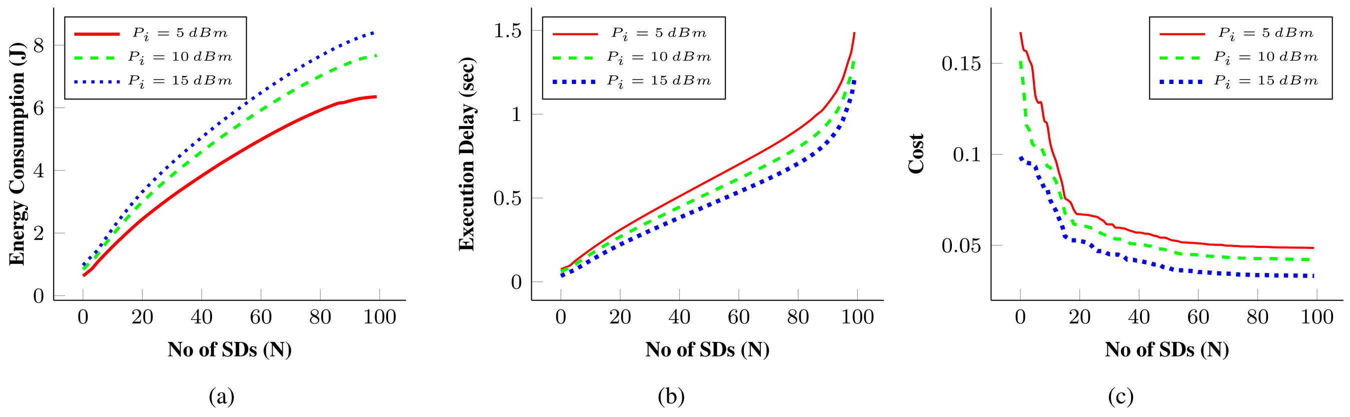


FIGURE 5. Effect of multiple SDs on (a) energy consumption (b) execution delay and (c) payment cost for different transmission power.

energy consumption increases with an increasing number of SDs. This is because the background interference increases with the increasing number of SDs; more transmission power is required to generate a higher uplink rate for sending data to the MEC network. Moreover, with an increasing offloading probability, the number of offloaded service requests increases. Hence, the average waiting time at the MEC network increases, as evident from (13). Also, comparing the three curves in fig. 4b, we find that as the number of SDs increase the execution delay gradually decreases for the low offloading probabilities. This is because the congestion and waiting delay at the MEC node increases when the offloading probability is high. Equation (4), also shows that the transmission delay increases with increasing SDs. Therefore, the overall execution delay increases, Fig. 4b. The increment in transmission power and delay with the increasing number of SDs in the system results in shifting the preference more towards local execution. As a result, the payment cost incurred for remote execution decreases with increasing SDs, Fig. 4c.

In Fig. 5, we study the effect of varying transmission powers (P_i) on the offloading objectives. For this, the initial offloading probability (p_i^m) of all the SDs is 0.5. Fig. 5a shows that energy consumption increases with an increasing number of SDs. The energy consumption is directly proportional to transmission power, which is also evident from (5). In Fig. 5b, the execution delay reduces with an increase in transmission power. This is because increasing the transmission power at the SD generates a higher uplink rate, which in turn reduces the transmission delay required for forwarding a request to the MEC network, as given by (4). However, the payment cost increases for low transmission power because the service request takes more time to reach the edge nodes, which causes less congestion in the MEC network. This encourages offloading, and the payable cost increases, Fig. 5c.

In Fig. 6, we study the proposed scheme by varying the number of edge nodes in the MEC network. For this set of experiments, we set the initial offloading probability as 0.5 and transmit power 10 dBm for all the SDs. We perform the simulation for three sets of edge nodes in the MEC network, i.e., 4, 8, and 12. From Fig. 6a and Fig. 6b, we observe that the energy consumption and execution delay decreases with an increasing number of edge nodes. This is because, with an increasing number of edge nodes, the processing load at the MEC network decreases, allowing SDs to offload more requests for the remote execution. As a result, the payment cost increases with an increasing number of MEC nodes, as shown in Fig. 6c.

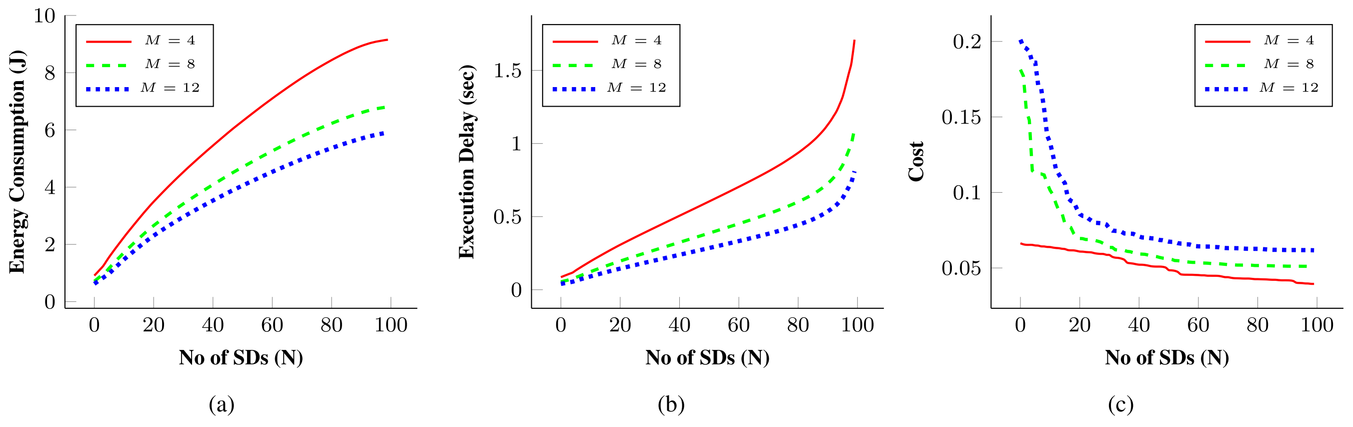


FIGURE 6. Effect of multiple SDs on (a) energy consumption (b) execution delay and (c) payment cost for different number of edge nodes.

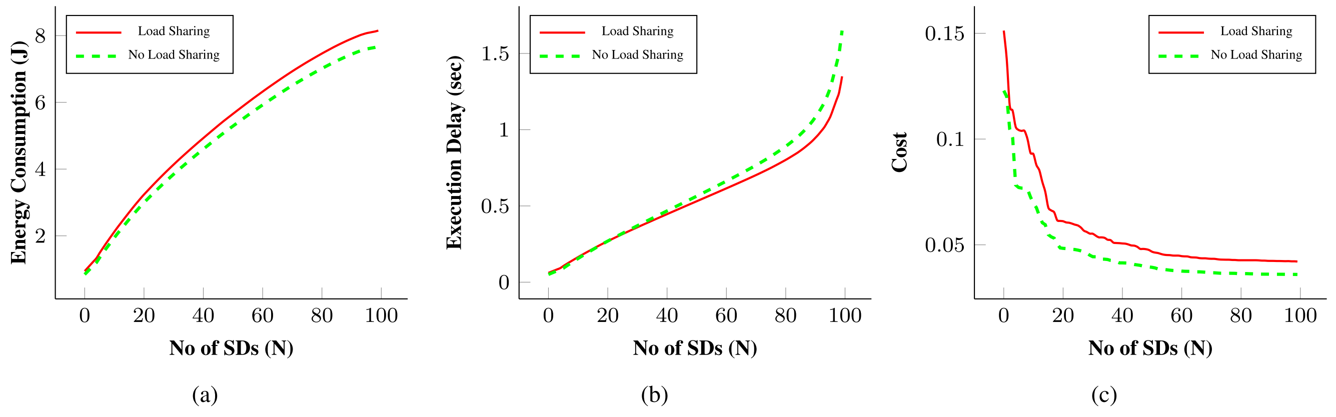


FIGURE 7. Effect of load sharing on (a) energy consumption (b) execution delay and (c) payment cost for multiple SDs.

B. LOAD SHARING VS NO LOAD SHARING

In Fig. 7, we show the effect of load sharing between edge nodes. For this set of experiments, we set the initial offloading probability as 0.5 and transmit power 10 dBm for all the SDs. By setting the routing probability p_{kj} as 0.3 and 0, we compare the performance of the proposed model with and without load sharing between the edge nodes. If the load sharing is denied between edge nodes, the processing load at an individual edge node may increases resulting in high queuing delay, which increases the total execution delay, Fig. 7b. Thus, more tasks are preferred to be executed locally, which increases energy consumption Fig. 7a, but the payment cost is reduced, Fig. 7c. Therefore, the performance of the system improves with the load sharing between MEC nodes.

The routing probability (p_{kj}) is the probability that a job is forwarded from an edge node k to the neighboring edge node j and leaves the MEC network with probability $1 - \sum_{j=1}^{M} p_{kj}$; in other words, routing probability determines the load sharing between the MEC nodes. For simplicity, we assume a constant routing probability of $p_{kj} = 0.3$, which yields a better simulation result. We derive this value empirically by using different values for the routing probabilities (i.e., $p_{kj} = \{0.1, 0.3, 0.5\}$), as shown in Fig. 8.

C. COMPARISON OF MEC BASED OFFLOADING WITH MCC OFFLOADING AND LOCAL EXECUTION

In this section, we show the efficiency of the proposed MEC scheme compared to cloud-based offloading (MCC) and local execution. In the proposed MEC based offloading, we consider the availability of edge resources with the collaboration cloud infrastructure providing services to the SDs. On the other hand, MCC-based offloading only considers the cloud computing resources for executing the offloading requests, and local execution indicates that all service requests are executed locally on the SDs. For comparing these scenarios, we define the *Average Ratio (AR)* [23], in (34). In the equation, AS_{MEC} is the aggregated sum of the optimal values of the offloading objectives (E, D, and C) obtained by the SGD algorithm for the MEC based offloading. Whereas, AS_{adj} is the aggregated sum of the objective values computed by the SGD algorithm for either MCC based offloading or the local execution scenario.

$$AR = \frac{AS_{MEC}}{AS_{adj}} \tag{34}$$

From Fig. 9, we see that MEC based offloading achieves better performance than the MCC based offloading and local

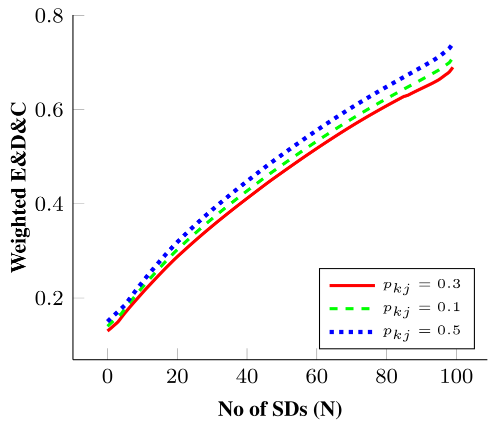


FIGURE 8. Comparison of the offloading objectives for different routing probabilities.

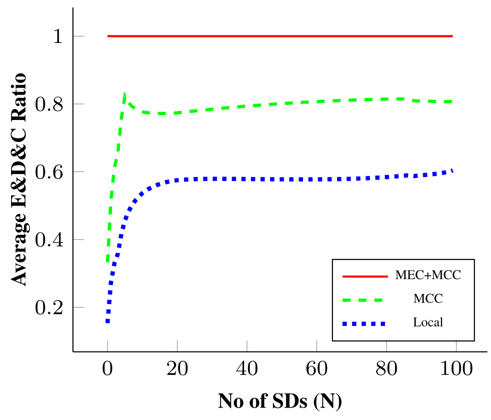


FIGURE 9. Comparison of different offloading paradigms.

execution. We observe that the performance of the MEC based offloading is approximately 20% better than MCC based offloading because the MEC servers process most of the services request; hence the execution delay and cost reduce significantly. In the local execution scenario, there is no execution cost and transmission delay. However, very high energy consumption in processing all the service requests on the SDs, therefore the offloading becomes beneficial in all local execution. From Fig. 9, we observe that MEC based offloading has a significant impact on the performance of the SDs, as the performance of local execution reduces by 40% compared to the MEC based offloading.

D. COMPARISON ANALYSIS AND DISCUSSION

In this section, we perform a comparative analysis of the proposed offloading framework. We begin by comparing the performance of the proposed model with the existing offloading scheme. Further, we compare the performance of the SGD based solution approach with the interior point method (IPM) algorithm on the formulated multiobjective problem.

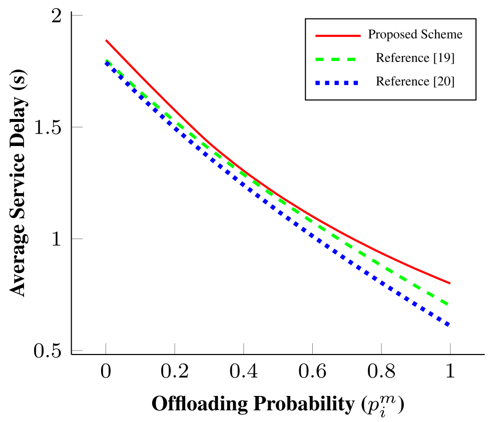
1) COMPARISON WITH PREVIOUS OFFLOADING SCHEMES

In [19] and [20] authors considers a distributed load sharing fog layer for executing the offloading requests from the SDs. The authors in [19] define three modes of processing, i.e., all fog processing (AFP), light fog processing (LFP), and no fog processing (NFP). In LFP, only light processing tasks are offloaded to the fog nodes. In NFP, there is no intermediate fog layer; hence tasks are executed on either SD or the cloud, which represents the MCC-based offloading. While in AFP, both light and heavy requests can be offloaded to the fog layer or cloud for processing. Meanwhile, in [20], authors have discussed a two-step process, i.e., network formation stage and task distribution stage. In the network formation stage, the fog network is formed without any prior information on the neighboring nodes. The task distribution stage offloads the tasks to the nodes on the fog network. From Fig. 10a, we find that the average service delay reduces with an increasing offloading probability. It can also be seen that the performance of the offloading schemes in [19] and [20] are slightly better than the proposed scheme. This is due to the optimization of multiple objectives (E, D, and C) in our model compared to single-objective optimization, i.e., minimization of service delay in [19] and [20].

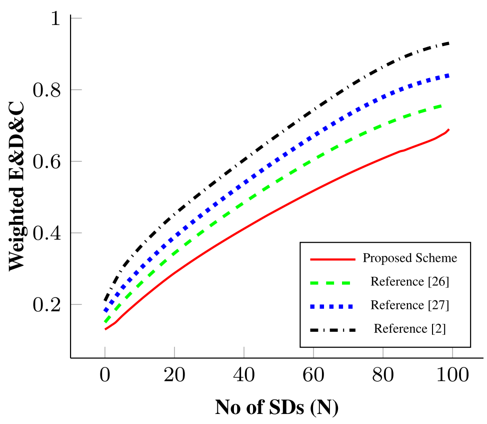
To show the effectiveness of the proposed model, we compare it with existing offloading schemes, i.e., game-theory based offloading scheme [26], IPM based algorithm [27], and iterative search algorithm combining interior penalty function in [2]. Authors in [27] propose a centralized offloading scheme by utilizing queuing theory and introduce an IPM based algorithm for minimizing the energy consumption, execution delay, and payment cost of the mobile devices. Zhang *et al.* [2] propose an iterative search algorithm combining the interior penalty function with D.C. programming (IPDC) to solve the trade-off between the energy consumption of SDs and execution latency of their tasks. Reference [26] propose a distributed collaborative computation offloading scheme using game theory to minimize mobile devices' energy consumption while satisfying their execution time. From Fig. 10b, we observe that, compared with the existing schemes, the proposed model performs better for the increasing number of SDs under variable service arrival rates, which illustrates the effectiveness of this study.

2) PERFORMANCE OF THE OPTIMIZATION ALGORITHMS

In the following, we compare the stochastic gradient descent (SGD) algorithm based solution approach with the interior-point method (IPM). We solve the aforementioned multiobjective problem formulated in (32) with the IPM method to minimize the objective function ($L2$) under the similar experimental setup. From Fig. 11a, we observe that the optimal objective values obtained are slightly better using the SDG algorithm compared to the IPM method. Moreover, there is a huge difference in the execution overhead of both the algorithms, shown in Fig. 11b. Based on our experimental setup, it takes approximately 27 seconds for the SGD and 130 seconds for the IPM to solve the formulated problem for the 100 SDs. This is because, the computational complexity of the IPM algorithm require $\mathcal{O}(\sqrt{n})$ iterations to yield an ϵ-complimentary solution [61] (where n is the sufficiently large number of data samples). SGD has the computational

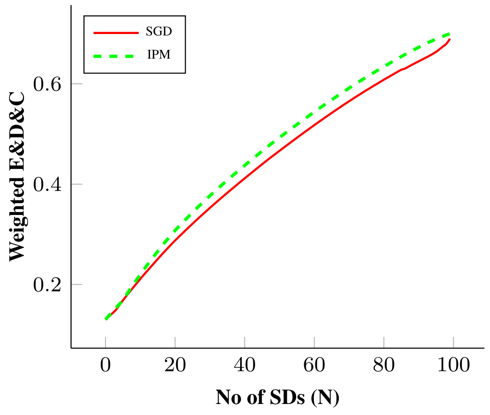


(a) Comparing the average service delay of the proposed model with [19] and [20].

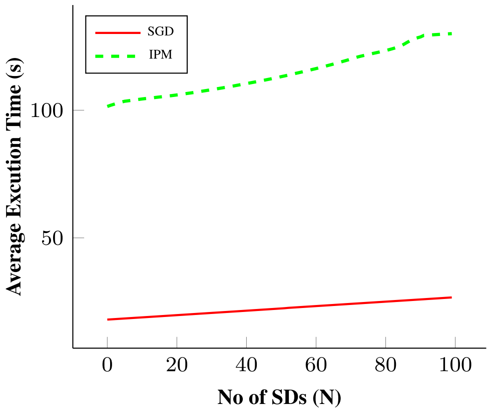


(b) Comparison of the proposed model with different offloading schemes.

FIGURE 10. Comparative analysis of the proposed model.



(a) Comparing the optimal objectives values obtained from SGD and IPM algorithms on the proposed scheme.



(b) Execution time analysis of SGD and IPM algorithms on the proposed scheme.

FIGURE 11. Performance comparison of SGD with IPM algorithms on the proposed model.

TABLE 3. Optimal values of the offloading objectives for the SGD and IPM algorithm.

Set of Weights	IPM			SGD		
	Energy Consumption (J)	Execution Delay (s)	Payment Cost	Energy Consumption (J)	Execution Delay (s)	Payment Cost
$(0.6, 0.2, 0.2)$	7.055	1.587	$5.06E-02$	6.938	1.490	$4.72E-02$
$(0.2, 0.7, 0.1)$	8.805	1.370	$5.52E-02$	8.543	1.294	$5.11E-02$
$(0.1, 0.2, 0.7)$	8.957	1.677	$2.99E-02$	8.880	1.587	$2.89E-02$

complexity of $O(1/\epsilon)$ [57], which is independent of the size of the dataset. Hence, the SGD algorithm converges faster to the optimal solution compared to the IPM algorithm. Therefore, the SGD algorithm is best suited for the delay sensitive application and solving large scale convex unconstrained optimization problems. Further, in Table 3, we show the performance of both the algorithms for different sets of weight factors ($\alpha_1, \alpha_2, \alpha_3$). The weights are selected such that the system emphasizes more on one of the objectives (i.e., either energy, delay, or cost) than the others.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a computation offloading scheme for efficient resource utilization of smart devices (SDs). The model presents a generalized three-tier IoT-edge-cloud framework, considering the collaboration of load sharing distributed MEC network and the cloud infrastructure. The goal is to find an optimal trade-off between the energy consumption of SDs, execution latency of their tasks, and the payment cost for remote execution a task on the edge or cloud server. A nonlinear multiobjective optimization problem is formulated by applying queuing theory at the SDs, MEC network, and cloud to analyze the delay performance, energy consumption at different network entities, and the cost for using edge and cloud services. For solving the formulated problem, stochastic gradient descent (SGD) algorithm based solution approach, in combination with the penalty method, is used to optimize the offloading probability and transmission power jointly. Finally, we study the proposed system by the simulation to understand its effectiveness under various parameters and demonstrate its effectiveness over the available MEC offloading schemes.

In future, we intend to include parameters, like output transmission delay and network delays within the MEC network. We may also extend our model for multi-class traffic and include routing probability optimization for better resource utilization. Additionally, the formulated nonlinear multiobjective problem can be solved using advanced optimization algorithms like AdaBound, Non-dominated Sorting Genetic Algorithm-II (NSGA-II), Strength Pareto Evolutionary Algorithm-2 (SPEA-2), Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) and so on. The trade-off may be better analyzed using such schemes as the results are provided in the form of Pareto-front and may vary depending upon the choice of the algorithm.

REFERENCES

[1] B. Li, Z. Fei, J. Shen, X. Jiang, and X. Zhong, "Dynamic offloading for energy harvesting mobile edge computing: Architecture, case studies, and future directions," *IEEE Access*, vol. 7, pp. 79877–79886, 2019.

[2] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, Aug. 2018.

[3] J. H. Anajemba, T. Yue, C. Iwendi, M. Alenezi, and M. Mittal, "Optimal cooperative offloading scheme for energy efficient multi-access edge computation," *IEEE Access*, vol. 8, pp. 53931–53941, 2020.

[4] C. Garrido-Hidalgo, D. Hortelano, L. Roda-Sanchez, T. Olivares, M. C. Ruiz, and V. Lopez, "IoT heterogeneous mesh network deployment for Human-in-the-Loop challenges towards a social and sustainable industry 4.0," *IEEE Access*, vol. 6, pp. 28417–28437, 2018.

[5] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, 2018.

[6] H. Ke, J. Wang, H. Wang, and Y. Ge, "Joint optimization of data offloading and resource allocation with renewable energy aware for IoT devices: A deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 179349–179363, 2019.

[7] X. Li, C. Zhang, B. Gu, K. Yamori, and Y. Tanaka, "Optimal pricing and service selection in the mobile cloud architectures," *IEEE Access*, vol. 7, pp. 43564–43572, 2019.

[8] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26652–26664, 2019.

[9] H. Wu, H. Tian, G. Nie, and P. Zhao, "Wireless powered mobile edge computing for industrial Internet of Things systems," *IEEE Access*, vol. 8, pp. 101539–101549, 2020.

[10] Q. Wu, H. Ge, H. Liu, Q. Fan, Z. Li, and Z. Wang, "A task offloading scheme in vehicular fog and cloud computing system," *IEEE Access*, vol. 8, pp. 1173–1184, 2020.

[11] Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas, and S. Khaf, "A deep learning approach for energy efficient computational offloading in mobile edge computing," *IEEE Access*, vol. 7, pp. 149623–149633, 2019.

[12] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.

[13] R. K. Balan and J. Flinn, "Cyber foraging: Fifteen years later," *IEEE Pervas. Comput.*, vol. 16, no. 3, pp. 24–30, Jul. 2017.

[14] Y. Fan, L. Zhai, and H. Wang, "Cost-efficient dependent task offloading for multiusers," *IEEE Access*, vol. 7, pp. 115843–115856, 2019.

[15] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 1451–1455.

[16] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2017, pp. 1–6.

[17] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019.

[18] F. Shan, J. Luo, J. Jin, and W. Wu, "Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless IoT environment," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4411–4422, Jun. 2019.

[19] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, Apr. 2018.

[20] G. Lee, W. Saad, and M. Bennis, "An online optimization framework for distributed fog network formation with minimal latency," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2244–2258, Apr. 2019.

[21] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.

[22] H. Guo, J. Zhang, J. Liu, and H. Zhang, "Energy-aware computation offloading and transmit power allocation in ultradense IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4317–4329, Jun. 2019.

[23] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[24] Z. Zhang, Z. Hong, W. Chen, Z. Zheng, and X. Chen, "Joint computation offloading and coin loaning for blockchain-empowered mobile-edge computing," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9934–9950, Dec. 2019.

[25] T. Yang, H. Feng, C. Yang, Y. Wang, J. Dong, and M. Xia, "Multivessel computation offloading in maritime mobile edge computing network," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4063–4073, Jun. 2019.

[26] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, May 2018.

[27] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[28] L. Liu, Z. Chang, and X. Guo, "Socially aware dynamic computation offloading scheme for fog computing system with energy harvesting devices," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1869–1879, Jun. 2018.

[29] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M.-T. Zhou, "FEMTO: Fair and energy-minimized task offloading for fog-enabled IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4388–4400, Jun. 2019.

[30] G. Zhang, Y. Chen, Z. Shen, and L. Wang, "Distributed energy management for multiuser mobile-edge computing systems with energy harvesting devices and QoS constraints," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4035–4048, Jun. 2019.

[31] Y. Wu, J. Shi, K. Ni, L. Qian, W. Zhu, Z. Shi, and L. Meng, "Secrecy-based delay-aware computation offloading via mobile edge computing for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4201–4213, Jun. 2019.

[32] M. Qin, L. Chen, N. Zhao, Y. Chen, F. R. Yu, and G. Wei, "Power-constrained edge computing with maximum processing capacity for IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4330–4343, Jun. 2019.

[33] M. Chao and R. Stoleru, "R-MStorm: A resilient mobile stream processing system for dynamic edge networks," in *Proc. IEEE Int. Conf. Fog Comput. (ICFC)*, Apr. 2020, pp. 64–72.

[34] M. Chao, C. Yang, Y. Zeng, and R. Stoleru, "F-MStorm: Feedback-based online distributed mobile stream processing," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 273–285.

[35] Q. Ning, C.-A. Chen, R. Stoleru, and C. Chen, "Mobile storm: Distributed real-time stream processing for mobile clouds," in *Proc. IEEE 4th Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2015, pp. 139–145.

[36] V. Cardellini, V. De Nitto Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. Lo Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Math. Program.*, vol. 157, pp. 421–449, Jun. 2016.

[37] S.-C. Hung, H. Hsu, S.-Y. Lien, and K.-C. Chen, "Architecture harmonization between cloud radio access networks and fog networks," *IEEE Access*, vol. 3, pp. 3019–3034, 2015.

[38] K.-K. Tse, "Some applications of the Poisson process," *Appl. Math.*, vol. 5, no. 19, p. 3011, 2014.

[39] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *Proc. IEEE 7th Int. Symp. Service-Oriented Syst. Eng.*, Mar. 2013, pp. 494–502.

[40] D. Gross, *Fundamentals Queueing Theory*. Hoboken, NJ, USA: Wiley, 2008.

[41] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4188–4200, Jun. 2019.

[42] L. Rui, Y. Yang, Z. Gao, and X. Qiu, "Computation offloading in a mobile edge communication network: A joint transmission delay and energy consumption dynamic awareness mechanism," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10546–10559, Dec. 2019.

[43] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont, MA, USA: Athena Scientific, 1998.

[44] J. R. Jackson, "Jobshop-like queueing systems," *Manage. Sci.*, vol. 50, no. 12, pp. 1796–1802, Dec. 2004.

[45] J. Vilaplana, F. Solsona, I. Teixidó, J. Mateo, F. Abella, and J. Rius, "A queuing theory model for cloud computing," *J. Supercomput.*, vol. 69, no. 1, pp. 492–507, Jul. 2014.

[46] S. Sthapit, J. Thompson, N. M. Robertson, and J. R. Hopgood, "Computational load balancing on the edge in absence of cloud and fog," *IEEE Trans. Mobile Comput.*, vol. 18, no. 7, pp. 1499–1512, Jul. 2019.

[47] E. P. Kao, *An Introduction to Stochastic Processes*. New York, NY, USA: Dover, 2019.

[48] A. Jonathan, A. Chandra, and J. Weissman, "Locality-aware load sharing in mobile cloud computing," in *Proc. the10th Int. Conf. Utility Cloud Comput.*, Dec. 2017, pp. 141–150.

[49] L. Li, Q. Guan, L. Jin, and M. Guo, "Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system," *IEEE Access*, vol. 7, pp. 9912–9925, 2019.

[50] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, vol. 6, pp. 12825–12837, 2018.

[51] X.-Q. Pham, N. D. Man, N. D. T. Tri, N. Q. Thai, and E.-N. Huh, "A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 11, p. 1550147717742073, 2017.

[52] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Eng.*, vol. 5, no. 1, Jul. 2018, Art. no. 1502242.

[53] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, vol. 2. Cham, Switzerland: Springer, 2016. [Online]. Available: https://doi.org/10.1007%2F978-3-319-18842-3, doi: 10.1007/978-3-319-18842-3.

[54] S. Koziel and X.-S. Yang, Eds., *Computational Optimization, Methods and Algorithms*. Berlin, Germany: Springer, 2011. [Online]. Available: https://doi.org/10.1007%2F978-3-642-20859-1, doi: 10.1007/978-3-642-20859-1.

[55] Z. Ashraf, D. Malhotra, P. K. Muhuri, and Q. M. D. Lohani, "Hybrid biogeography-based optimization for solving vendor managed inventory system," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 2598–2605.

[56] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, Sep. 1951.

[57] Y. Mu, W. Liu, X. Liu, and W. Fan, "Stochastic gradient made stable: A manifold propagation approach for large-scale optimization," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 458–471, Feb. 2017.

[58] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*, Y. Lechevallier and G. Saporta, Eds. Berlin, Germany: Physica-Verlag, 2010, pp. 177–186.

[59] J. Fliege, A. I. F. Vaz, and L. N. Vicente, "Complexity of gradient descent for multiobjective optimization," *Optim. Methods Softw.*, vol. 34, no. 5, pp. 949–959, Sep. 2019.

[60] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for IoT systems: A computation offloading game," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.

[61] I. Pólik and T. Terlaky, *Interior Point Methods for Nonlinear Optimization*. Berlin, Germany: Springer, 2010, pp. 215–276.

FARHAN SUFYAN (Graduate Student Member, IEEE) received the B.Sc. and master's degrees in computer science and applications from Aligarh Muslim University, Aligarh, India, in 2010 and 2014, respectively. He is currently pursuing the Ph.D. degree in computer science with South Asian University (SAU), New Delhi, India. His current research interests include the Internet of Things, mobile cloud computing, and fog computing.

AMIT BANERJEE (Member, IEEE) received the Ph.D. degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2009. After that, he worked for two years as an Engineer with the SoC Technology Center, Industrial Technology Research Institute (ITRI), Taiwan. He is currently working as an Assistant Professor with the Department of Computer Science, South Asian University (SAU), New Delhi, India. He has authored or coauthored papers in peer-reviewed journals and conferences, including the IEEE TRANSACTIONS. His current research interests include distributed computing, the Internet of Things, and edge computing.