Received September 12, 2020, accepted September 27, 2020, date of publication October 1, 2020, date of current version October 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3028240

A Deep Learning Approach for Mobility-Aware and Energy-Efficient Resource Allocation in MEC

ZAIWAR ALI[1], SADIA KHAF[2], ZIAUL HAQ ABBAS[2], GHULAM ABBAS[3], (Senior Member, IEEE), FAZAL MUHAMMAD[4], AND SUNGHWAN KIM[5], (Member, IEEE)

[1]Telecommunications and Networking (TeleCoN) Research Laboratory, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan
[2]Faculty of Electrical Engineering, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan
[3]Faculty of Computer Sciences and Engineering, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan
[4]Department of Electrical Engineering, City University of Science and Information Technology, Peshawar 25000, Pakistan
[5]School of Electrical Engineering, University of Ulsan, Ulsan 44610, South Korea

Corresponding author: Sunghwan Kim (sungkim@ulsan.ac.kr)

Following are results of a study on the "Leaders in INdustry-university Cooperation +" Project, supported by the Ministry of Education and National Research Foundation of Korea.

ABSTRACT Mobile Edge Computing (MEC) has emerged as an alternative to cloud computing to meet the latency and Quality-of-Service (QoS) requirements of mobile devices. In this paper, we address the problem of server resource allocation in MEC. Due to the dynamic load conditions on MEC servers, their resources need to be used intelligently to meet the QoS requirements of the users and to minimize server energy consumption. We present a novel resource allocation algorithm, called Power Migration Expand (*PowMigExpand*). Our algorithm assigns user requests to the optimal server and allocates optimal amount of resources to User Equipment (UE) based on our comprehensive utility function. *PowMigExpand* also migrates UE requests to new servers, when needed due to the mobility of users. We also present a low cost Energy Efficient Smart Allocator (EESA) algorithm that uses deep learning for energy efficient allocation of requests to optimal servers. The proposed algorithms consider varying load of incoming requests and their heterogeneous nature, energy efficient activation of servers, and Virtual Machine (VM) migration for smart resource allocation and, thus, is the first comprehensive approach to address the complex and multidimensional resource allocation problem using deep learning. We compare our proposed algorithms with other resource allocation approaches and show that our approach can handle the dynamic load conditions better. The proposed algorithms improve the service rate and the overall utility with minimum energy consumption. On average, it reduces 26% energy consumption of MESs and improves the service rate by 23%, compared with other algorithms. We also get more than 70% accuracy for EESA in allocating the resources of multiple servers to multiple users.

INDEX TERMS Mobile edge computing, resource allocation, computational offloading, deep learning, energy efficient.

I. INTRODUCTION

As opposed to traditional computers, mobile devices process only a handful of tasks, and have certain inherent limitations, such as low processing power, memory and battery life. On the other hand, mobile applications are becoming more complex every day, especially when it comes to video games and graphics processing [1]–[3]. Similarly, there are applications involving speech processing and recognition, biomedical image data processing, video games, etc. that require the kind of processing capabilities that even the modern day mobile phones cannot provide [4]–[6]. An obvious solution is to offload the processing of compute-intensive applications to powerful remote servers. This idea, called cloud computing [7], gained popularity in the past few years as a viable solution to execute complex applications on mobile devices without using much resources of mobile devices. However, the distance between mobile devices and

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

cloud servers is the source of enormous latencies. The transmission, execution, and reception times become too large for acceptable user experience on a mobile device [8], e.g., in Google Assistant, Siri, and Alexa [9].

A more realistic approach is to bring the network of cloud servers closer to the mobile device, to minimize the overall application execution delays [10]. This approach of bringing a distributed network of powerful servers closer to end users is known as Mobile Edge Computing (MEC) that has attracted much attention recently [11]. Once an application is offloaded to the central control unit of MEC, the next task is to select the best possible server to entertain this request, and to intelligently allocate server's resources to this request so as to guarantee acceptable performance for the mobile user as well as maximize the rewards for the Mobile Edge Server (MES), under the constraints of its energy consumption. The resources here refer to the CPU, RAM, disk space, and time that an MES allocates to a mobile user/User Equipment (UE).

A suite of algorithms for resource allocation in cloud computing and mobile edge computing are presented in [12], in which the authors consider only the CPU as user request. However, some of the important and realistic resources of MES (i.e., RAM, hard disk space and the time for which these resources are requested) and the mobility of UEs are not considered in their work. In this paper, we present an energy efficient resource allocation algorithm, called Power Migration Expand (*PowMigExpand*), which considers the mobility of UEs and migrates the Virtual Machine (VM) of UEs from one server to another, when required, with high utility for MESs. We consider a realistic multi-user multi-server scenario and try to answer the following two questions: *Which MES will be the optimal server for an incoming request by a UE?* and *How much resources of the MES should be allocated to that particular UE?* The answers to these questions depend on several factors, such as the amount of resources requested by the UE, the reward for the MES, available resources of the MES, the distance between the UE and the MES, and MES energy consumption. Energy consumption is an important parameter since cloud computing data centers and distributed edge computing servers are leaving a large carbon footprint on the planet [13] because of unconstrained resource allocation methodologies. A responsible approach would be to optimize the resource allocation on MEC servers such that maximum possible requests are served using the minimum required servers, and the remaining servers can be kept idle to save their energy.

A. NOVELTY AND CONTRIBUTION

The novelty of this paper can be highlighted as follows:

- Formulation of a comprehensive distance-aware utility function which considers the mobility of users.
- Energy-aware VM migration based on mobility-aware utility function.
- A deep learning based smart allocator, which reduces the computational complexity.

TABLE 1. Possible allocation options when MESs=2 and UEs=3.

Options	MES-1	MES-2
1	UE-1, UE-2, UE-3	Idle
2	UE-1, UE-2	UE-3
3	UE-1, UE-3	UE-2
4	UE-2, UE-3	UE-1
5	UE-1	UE-2, UE-3
6	UE-2	UE-1, UE-3
7	UE-3	UE-1, UE-2
8	Idle	UE-1, UE-2, UE-3

To the best of our knowledge, in utility based approaches, no such comprehensive mathematical model has been proposed previously for UE request and utility function in MEC resource allocation with UEs mobility considerations. We solve the resource allocation problem under the constraints of limited range and energy consumption of MESs. We present an algorithm that considers user mobility and migrates VMs to other MESs when needed. The comprehensive utility function considers all the realistic resources for utility calculations. Therefore, the frequent VM migrations and comprehensive utility calculations pose an additional overhead for the said resource allocation. The overhead grows with number of MESs (n) and number of UEs (m) because there are n^m possible options for an allocation scheme to select from. For example, for 2 MESs and 3 UEs, the total possible options that a scheme will have to check are $2^3 = 8$, as given in Table 2.

An allocation scheme selects the best option in terms of low energy consumption and high utility and service rate with different approaches. In our proposed work, we consider all n^m possible options and then select the option with high utility and low energy consumption. However, as the number of MESs or UEs increases, the overhead increases exponentially. To reduce the computational overhead and make the decision faster, we also provide a deep learning approach that reduces this overhead by making it a simple multiplication problem $O(mn)$ using a pre-trained Deep Neural Network (DNN). We generate an exhaustive dataset by using the comprehensive mathematical formulation and then use the dataset to train a two-layered DNN, which then acts as the smart allocator. The proposed algorithm maximizes the service rate for UEs and utility for MESs. The optimal selection of MES and energy-aware server activation also minimize the energy consumption.

The remainder of this paper is organized as follows. Section II presents a review of the related work. Section III describes the system model and the mathematical formulation of *PowMigExpand*. Section IV explains the resource allocation schemes and the proposed algorithms. Sections V presents simulation results and discussion, and Section VI concludes the paper.

II. RELATED WORK

A detailed survey on the limitations and challenges in resource allocation problem has been presented in [14].

TABLE 2. Comparison of various algorithms for resource allocation in MEC.

Algorithm	Considers utility function for MES?	Uses deep learning?	MES resources considered	Mobility considered?	MES energy considered?
Cardosa et al. [12]	Yes	No	CPU	No	Yes
Follow-me Behaviour [18]	No	No	CPU	Yes	No
DRLRA [19], MOACO [20]	No	Yes	CPU, time	Yes	No
ARCES [21]	No	No	CPU	No	Yes
Ng et al. [22, 23]	No	No	Power, antennas	No	Yes
OSRM [24], Stackelberg dynamic game [25]	No	No	Power, time	No	No
Jin et al. [26]	No	No	CPU, time	Yes	No
Zhang et al. [27]	No	No	CPU, time	No	No
Sandpiper [28]	No	No	CPU, time	Yes	No
Dolphin partner technique [29]	No	No	CPU, memory, time	Yes	Yes
Particle swarm optimization [30]	No	No	CPU, time	Yes	No
Non-cooperative game [31]	Yes	No	Time	No	No
Cooperative game [32]	No	No	Time	Yes	No
Zakarya et al. [33]	Yes	No	CPU, time	Yes	Yes
Our proposed algorithm	Yes	Yes	CPU, RAM, disk space, time	Yes	Yes

Most of the existing literature focuses on the resource optimization in the tasks, i.e., the offloading process [15], minimization of overall energy consumption of UEs [16], and the optimization of MEC resource allocation [17]. The resource allocation problem in this paper refers to the resources of the MES, and not the network conditions, since the varying network conditions scenario has already been addressed in [15].

A mobility-aware dynamic resource allocation approach is presented in [18]. The authors discuss the resource allocation strategies and the migration strategies adopted specifically to MEC. Their aim is to implement the follow-me behavior of mobile edge resources, which depends upon the location of the mobile users. They map physical areas to logical resource communities, and the movement of a UE triggers the call for migration. The approach, however, does not take the heterogeneous nature of applications into account. The requested resource is assumed to be CPU only and the approach does not describe any clear formulation of the utility for servers.

Deep learning approaches are used in [19] and [20] to minimize the service time in MEC resource allocation. However, the energy consumption and the utility function for MESs and the mobility of UEs are not considered in the mathematical models. The authors in [21] propose a computing-plus-communication energy model to minimize the energy consumption of MESs. They use a hybrid-powered MES and switching techniques of transmission drivers, however, the VM migration and utility for MES is not considered.

The authors in [22] and [23] propose a resource allocation algorithm for OFDMA networks with large numbers of BS antennas as a non-convex optimization problem. An efficient iterative algorithm with optimized power allocation, subcarrier allocation, and antenna allocation policies are proposed by using fractional programming. However, the VM migration and consideration of CPU, RAM, and hard disk space are not considered in the mathematical model. For MEC networks, we need to consider the CPU, RAM, hard disk space, and time of an MES in the utility function for a more realistic scenario.

The authors in [24] use Lyapunov optimization technique with a stochastic approach for wireless-powered MEC to minimize the energy consumption of UEs by optimizing the transmission power of MESs. A dynamic game-based approach is presented in [25] for resource allocation in wireless powered MEC. In this approach, the resources are optimally allocated by computing optimal transmission power and optimal task offloading. However, the VM migration and utility function for the MESs are not considered in [24] and [25].

The authors in [26] present an energy efficient resource allocation for MEC considering one MES and multiple UEs. However, the focus of the paper is on the energy consumption of UEs. In this paper, we present an energy efficient resource allocation at MESs side for multiple MESs and UEs. Similarly, the authors in [27] propose the dynamic task offloading and resource allocation for edge computing. However, the energy consumption of MESs is not considered. The main focus of their work is on the offloading problem and the energy consumption of UEs are considered.

In [28], the authors show the placement of VMs using migrations to optimize the load, however, the migration depends on the usage of the CPU and Service Level

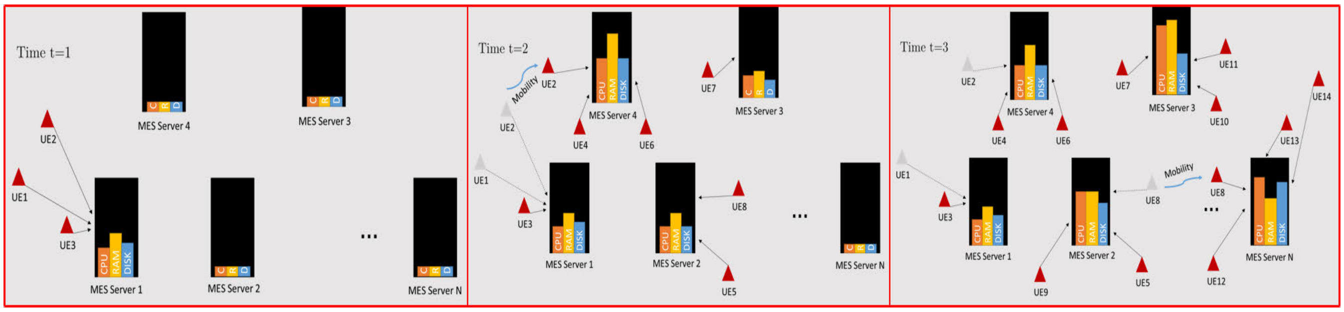


FIGURE 1. State of the system in time slots 1, 2, and 3. UE-2 and UE-8 move to another MES in time slots 2 and 3. Requests for UE-1 and UE-2 have been completely serviced in time slot 3.

Agreement (SLA) violations. Our proposed work places the VMs according to the user request and the available resources in the feasible MESs. The migration of the VMs depends on the mobility of the users, the availability of resources on the MESs, migration utility, and the time left to complete the user request.

The authors in [29] consider the resource allocation problem for the streamline security of VMs by the Dolphin Partner Optimization (DPO) method. The main concern of the paper is security. The paper also minimizes the energy consumption due to VM migration. However, the approach is not a utility-based approach. Moreover, VM migrations are not based on the mobility of users.

The authors in [30] consider a multi-user multi-server scenario to minimize the overall delay in the application execution. They consider UE mobility and VM migration. The focus of their work is on computational offloading and service delay minimization. In this paper, we propose a resource allocation algorithm on the MES side. Our work focuses on considering UE mobility as well as minimizing the overall energy consumption of the MES.

A game-based approach is presented in [31]–[33] for computational offloading and resource allocation in MEC. However, the main focus of [31] is on computational offloading and the energy consumption for MESs is not considered. Similarly, the authors in [32] solve the resource allocation problem using cooperative game theory. However, they do not consider the mobility of UEs and energy consumption for MESs. The authors in [31]–[33] only consider the CPU and time as resources requested by UEs and the utility functions do not depend upon the realistic resources such as CPU, RAM, and disk space.

A suite of techniques are presented in [12] for the resource allocation problem on the MES side. The authors consider only CPU as user request and take utilities according to requested CPU only. They present multiple energy-aware allocation schemes to allocate server resources to UEs. However, all the schemes in [12] consider only CPU as user request and take utilities for MES according to the requested CPU only. UEs are also considered to be stationary in [12]. The objective of this paper is to maximize the server utility and the UE service rate, while minimizing the MES energy consumption, in a realistic multi-user multi-server scenario considering UE mobility.

III. SYSTEM MODEL

We consider a multi-user multi-server scenario. The number of incoming requests is modeled according to the Poisson distribution. The system consists of a central control unit that detects the incoming UE requests, selects the optimal MES for each request, and allocates MES resources to these requests. We assume that the task of UE is successfully received as a UE request to the central control unit of MEC. The task offloading and the impact of physical communication channel are studied in [34] and [15].

Figure 1 shows the system model used in this paper. All the servers are idle initially and there are three incoming UE requests in time slot 1 that are served at MES-1. In time slot 2, UE-2 moves from its initial location and becomes closer to MES-4 so its request is moved to MES-4. The resource usage of MES-1 decreases and that of MES-4 increases accordingly. A similar trend can be seen in time slot 3 where UE-8 moves, and requests of UE-1 and UE-2 have been fulfilled. The resource allocation depends upon various factors, such as the amount of resources requested by the user, the amount of resources available at the particular MES, distance between the MES and the UE, and the allocation scheme. The request vector that the UE sends to the central control unit contains the information of the requested resources (CPU, RAM, disk space), the time for which the user is requesting these resources, and the location of the UE. The central control unit can use the location of the UE by converting it into a distance vector, containing the distance of the UE from each MES.

We propose a more realistic UE request model in which the UE transmits requested resources, time, and its own location. We assume that the UE continuously transmits its location so that if it goes far away from one MES and closer to another MES, the central control unit can issue a command to *migrate* its VM to the other MES. The incorporation of the UE mobility makes the model more realistic. In time slot i, if a user j has distance d_{jk} from MES k, due to its mobility, it may change in the next time slot and our proposed algorithm, *PowMigExpand*, takes that mobility into account.

TABLE 3. Notations.

Notations	Meaning
c_j	CPU allocated to UE j
$c_{j_{max}}, c_{j_{min}}$	Maximum and minimum CPU requested by UE j
$c_{k_{av}}$	Available CPU at MES k
$c_{k_{total}}$	Total CPU of MES k
$\mathbf{c}_{max}$	Row vector of the maximum CPU requested by UEs
$\mathbf{c}_{min}$	Row vector of the minimum CPU requested by UEs
C_P_k	Capacity of the MES k
c_{total}	Sum of CPUs of all the MESs
$\mathbf{D}, \mathbf{D}^i$	Distance matrix and distance matrix at i^{th} time slot
d_{jk}	Distance between UE j and MES k
d^i_{jk}	Distance between UE j and MES k at i^{th} time slot
d_{max}	Maximum distance within which all MESs operate
E_{ck}, E_{hk}, E_{rk}	Energy consumption due to the usage of CPU, hard disk, RAM
E_k	Energy consumption per unit time to keep MES k ON
$E_{k_{total}}$	The total energy consumption of MES k
e_{max}	Sum of E_k for all MESs
e_{puu}	Energy consumption per unit utility
E_{total}	Sum of energy consumptions of all the MESs
$\mathbf{f}_j$	The feasibility vector of UE j
f_{jk}	The feasibility of MES k for UE j
$\gamma_1, \gamma_2, \gamma_3, \gamma_4$	Unit balancing coefficients
$\mathbf{h}$	Row vector of the disk space requested by UEs
h_j	Disk space requested by UE j
$h_{k_{av}}$	Available disk space at MES k
$h_{k_{total}}$	Total disk space of MES k
h_{total}	Sum of disk space of all MESs
i	The time index
j	The UE index
k	The MES index
m	Total number of UEs
n	Total number of MESs
N_{R_S}	The number of requests served
N_{R_T}	The total number of requests
p_k	Energy consumption per unit capacity of MES k
$\mathbf{Q}$	User request matrix
r_j	RAM allocated to UE j
$r_{j_{max}}$	The maximum CPU requested by UE j
$r_{j_{min}}$	The minimum CPU requested by UE j
$r_{k_{av}}$	Available RAM at MES k
$r_{k_{total}}$	Total RAM of MES k
$\mathbf{r}_{max}$	Row vector of the maximum RAM requested by UEs
$\mathbf{r}_{min}$	Row vector of the minimum RAM requested by UEs
r_{total}	Sum of RAMs of all MESs
s^*_j	The optimal MES for UE j
s_k	Status of the MES k
S_R	Service rate
$\mathbf{t}$	Row vector of time requested by UEs
t_{active}	Server active time
t_j	Time allocated to UE j
$t_{k_{active}}$	Server active time of MES k
t_{max}	The maximum time a UE can request
$\mathbf{U}$	Utility matrix
$\mathbf{U}^i$	Utility matrix at i^{th} time slot
u_{jk}	Utility of UE j at MES k
u'_{jk}	Penalized utility of UE j at MES k
$u_{jk_{max}}$	Maximum utility of UE j at MES k
$u_{jk_{min}}$	Minimum utility of UE j at MES k
$u_{k_{total}}$	Total utility of MES k
u_{put}	Utility per unit active time
u_{total}	Sum of utilities of all MESs
w_1, w_2, w_3, w_4	Weighting coefficients

We consider n number of servers and m number of UE requests. The MESs are first sorted into the increasing order of their energy consumption per unit capacity and labeled accordingly so as to use the most profitable (least expensive) servers first, and the more expensive ones only in times of high demand. The central control unit finds the optimal server for each incoming UE request and then allots MES resources to it. For this purpose, we propose a mobility-aware utility function that considers the amount of resources requested, time, and distance between the UE and MES. Table 3 gives a summary of the notations used in this paper.

A. USER REQUESTS

The user request matrix contains the information about the resources requested by the UE, the time for which these resources are requested, and the distance of the UE from each MES (calculated by the central control unit from the location of the UE). Since a UE should specify the minimum and the maximum requested resources [12], we model the request matrix into two parts to incorporate the mobility of UEs: (i) the requested resources matrix $\mathbf{Q}$, and (ii) the distance matrix $\mathbf{D}$. The resource matrix $\mathbf{Q}$ is given by

$$\mathbf{Q} = [\mathbf{c}^T_{min}, \mathbf{c}^T_{max}, \mathbf{r}^T_{min}, \mathbf{r}^T_{max}, \mathbf{h}^T, \mathbf{t}^T], \tag{1}$$

where $\mathbf{c}^T_{min}$ and $\mathbf{c}^T_{max}$ represent the vectors[1] of the minimum and maximum amount of CPU in m UE requests, $\mathbf{r}^T_{min}$ and $\mathbf{r}^T_{max}$ represent the vectors of the minimum and maximum amount of RAM in UE requests, $\mathbf{h}^T$ is the vector of the amount of disk space in UE requests, and $\mathbf{t}^T$ is the vector of the number of time slots specified in the UE requests for which these resources are needed. The disk space is not modeled as a variable resource since, in most real word scenarios, an application needs a fixed amount of disk space to store the data. The central control unit must know about these six parameters of m UE requests. Therefore, the order of matrix $\mathbf{Q}$ is $m \times 6$. These vectors can be written as

$$\begin{aligned}
\mathbf{c}_{min} &= [c_{1_{min}}, c_{2_{min}}, \cdots, c_{m_{min}}], && (2)\\
\mathbf{c}_{max} &= [c_{1_{max}}, c_{2_{max}}, \cdots, c_{m_{max}}], && (3)\\
\mathbf{r}_{min} &= [r_{1_{min}}, r_{2_{min}}, \cdots, r_{m_{min}}], && (4)\\
\mathbf{r}_{max} &= [r_{1_{max}}, r_{2_{max}}, \cdots, r_{m_{max}}], && (5)\\
\mathbf{h} &= [h_1, h_2, \cdots, h_m], && (6)\\
\mathbf{t} &= [t_1, t_2, \cdots, t_m], && (7)
\end{aligned}$$

where $c_{1_{min}}$, $c_{1_{max}}$, $r_{1_{min}}$, $r_{1_{max}}$, h_1, and t_1 represent the minimum CPU, maximum CPU, minimum RAM, maximum RAM, disk space, and time requested by UE-1, respectively. Similarly, $c_{m_{min}}$, $c_{m_{max}}$, $r_{m_{min}}$, $r_{m_{max}}$, h_m, and t_m represent the requested data of UE m. In general, we can say that $c_{j_{min}}$, $c_{j_{max}}$, $r_{j_{min}}$, $r_{j_{max}}$, h_j, and t_j represent the minimum CPU, maximum CPU, minimum RAM, maximum RAM, disk space, and time requested by UE j, where $j = 1, 2, \ldots, m$. The dimension of vectors $\mathbf{c}_{min}$, $\mathbf{c}_{max}$, $\mathbf{r}_{min}$, $\mathbf{r}_{max}$, $\mathbf{h}$, and $\mathbf{t}$ is $1 \times m$.

We assume that a required task of a UE can be executed by minimum as well as maximum resources with different utility and QoS. Here, minimum and maximum resources refers to the minimum and maximum size of the VM which is assigned to a UE in an MES. If the central control unit allocates minimum requested resources to the UE, the UE can execute a smaller number of tasks and pay a lower premium to the MES. Otherwise, if the central control unit allocates the maximum requested resources to the UE, the UE can perform more tasks and pay a higher premium.

Definition 1: Premium: In this paper, a premium paid by a UE is a *reward* for the MES and is referred to as a unit-less quantity, called ***utility***.

[1] As a convention, all the vectors and matrices are represented in boldface.

The location of the UE is translated into a distance vector by the central control unit in terms of its distance from each MES. The distance matrix $\mathbf{D}$ containing the distances of m UEs from n MESs is given by

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix}, \tag{8}$$

where d_{11} is the distance of UE-1 from MES-1. Similarly, d_{mn} is the distance between UE m and MES n. In general, d_{jk} represents the distance of UE j from MES k. We consider the mobility of users in multi-user multi-server scenario, therefore, this distance matrix may change in the next time slot, depending upon the mobility of users. The proposed work updates the mobility-aware distance matrix $\mathbf{D}$ in each time slot as shown in Figure 2. The dimension of $\mathbf{D}$ is $m \times n$ and for a time slot i it is represented as

$$\mathbf{D}^i = \begin{bmatrix} d^i_{11} & d^i_{12} & \cdots & d^i_{1n} \\ d^i_{21} & d^i_{22} & \cdots & d^i_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d^i_{m1} & d^i_{m2} & \cdots & d^i_{mn} \end{bmatrix}. \tag{9}$$

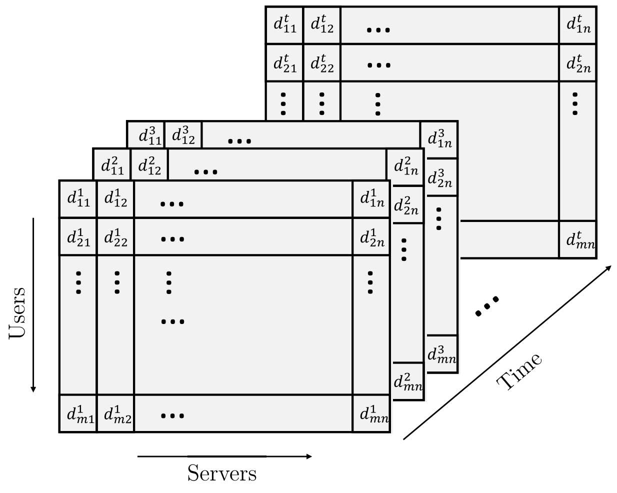


FIGURE 2. Distance matrix containing distances of *m* UEs from *n* MESs in *t* time slots.

B. THE UTILITY FUNCTION

The UEs request MES resources to execute their applications remotely and this gives them a chance to execute their applications in a more efficient way by using powerful servers. The servers, in return for allowing their resources to be used, get a utility. This utility depends upon the amount of CPU, RAM, and disk space requested by the UE, the time for which the UE has requested these resources, and upon the distance between the UE and MES. Therefore, the utility function is directly proportional to the requested resources and time. The QoS that the UE receives is inversely proportional to the distance between the UE and the MES due to network conditions, transmission range of the UE and MES, frequent disconnections, and the latency in communication. The utility is directly related to the QoS and, hence, is inversely proportional to the distance between the UE and the MES, and is given as

$$u_{jk} = \frac{(\gamma_1 c_j + \gamma_2 r_j + \gamma_3 h_j)\gamma_4 t_j}{d_{jk}}. \tag{10}$$

Here, u_{jk} is the utility for MES k for serving UE j. Moreover, c_j, r_j, and h_j, respectively, denote the amount of CPU, RAM and hard disk space allocated to UE j by MES k, t_j is the time for which these resources are allocated to UE j, and d_{jk} is the distance of UE j from MES k. The requested resources are weighted and normalized by γ_1, γ_2, and γ_3 as follows

$$\gamma_1 = \frac{w_1}{c_{total}}, \tag{11}$$
$$\gamma_2 = \frac{w_2}{r_{total}}, \tag{12}$$
$$\gamma_3 = \frac{w_3}{h_{total}}. \tag{13}$$

Here, c_{total}, r_{total}, and h_{total} are the combined total CPU, RAM, and disk space of all the servers, respectively, whereas, w_1, w_2, and w_3 are the weighting coefficients. For example, the value of w_1 shows the relative contribution of CPU to the utility function. These coefficients can be adjusted to represent the expensiveness of different resources. Similarly, to keep the utility function unit-less, t_j and d_{jk} are normalized with γ_4, i.e.,

$$\gamma_4 = \frac{d_{max}}{t_{max}}. \tag{14}$$

Here, d_{max} is the maximum distance within which all the MESs and UEs are operating, and t_{max} is the maximum time that a UE is allowed to request. The upper threshold on time is set such that a UE cannot hog all the resources and prevent other users from getting the service.

The utility for each MES is different from other MESs for serving the UE request because of the UE's location. A server receives higher utility by serving the nearby UEs than by serving the ones that are far away from it because the UEs closer to the server get better QoS and pay a higher premium for it. The utility of n servers for entertaining the same UE j can, thus, be represented by a vector $\mathbf{u}$ as follows

$$\mathbf{u} = [u_{j1}, u_{j2}, \ldots, u_{jn}]. \tag{15}$$

Similarly, the matrix of utilities for m UEs and n severs is defined as

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mn} \end{bmatrix}, \tag{16}$$

where $\mathbf{U}$ is an $m \times n$ matrix of utilities. Here, u_{11} is the utility of UE 1 at MES 1, and u_{mn} is the utility of UE m at MES n. Since the distance d_{jk} for several UEs may be updated in the next time slots because of their mobility, their utilities will also be updated accordingly. In general, the distance of UE j

from MES k in time slot i can be represented by d^i_{jk} and their utility can be represented by u^i_{jk}. The mobility-aware utility matrix can, therefore, be given as

$$\mathbf{U}^i = \begin{bmatrix} u^i_{11} & u^i_{12} & \cdots & u^i_{1n} \\ u^i_{21} & u^i_{22} & \cdots & u^i_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u^i_{m1} & u^i_{m2} & \cdots & u^i_{mn} \end{bmatrix}. \tag{17}$$

According to a UE request, the minimum and maximum utility that an MES can get by serving the request can be represented by

$$u_{jk_{min}} = \frac{(\gamma_1 c_{j_{min}} + \gamma_2 r_{j_{min}} + \gamma_3 h_j)\gamma_4 t_j}{d_{jk}}, \tag{18}$$

$$u_{jk_{max}} = \frac{(\gamma_1 c_{j_{max}} + \gamma_2 r_{j_{max}} + \gamma_3 h_j)\gamma_4 t_j}{d_{jk}}, \tag{19}$$

where $u_{jk_{min}}$ and $u_{jk_{max}}$ represent the minimum and maximum utilities for MES k from UE j.

Our proposed algorithm finds the MES with the maximum utility for each UE since that will provide the best QoS to the UE. From a set of n available servers, we find a server k, such that $k \in \{1, 2, 3, \ldots, n\}$, for which the utility of a UE j is maximum. This MES k is the optimal MES s^*_j for UE j, and is represented as

$$s^*_j = \arg \max_{k \in \{1,2,\ldots,n\}} u_{jk}, \quad j = 1, 2, \ldots, m. \tag{20}$$

C. FEASIBILITY OF SERVERS

A feasible MES k for UE j has enough available resources to serve UE j. If the current available resources of MES k are less than the requested resources of UE j, then this MES k will not be feasible for UE j. The feasibility status of MES k for UE j, f_{jk}, can be written as

$$f_{jk} = \begin{cases} 1 & c_{j_{min}} \le c_{k_{av}} \text{ and } r_{j_{min}} \le r_{k_{av}} \\ & \text{and } h_j \le h_{k_{av}} \\ 0 & \text{oherwise} \end{cases}, \tag{21}$$

where $c_{k_{av}}$, $r_{k_{av}}$, and $h_{k_{av}}$ are the available resources at MES k. $f_{jk} = 1$ means that MES k is feasible for UE j and $f_{jk} = 0$ means that MES k is not feasible for UE j. For UE j, the feasibility vector of n available MESs, $\mathbf{f}_j$, can be written as

$$\mathbf{f}_j = [f_{j1}, f_{j2}, \cdots, f_{jn}], \tag{22}$$

where the order of $\mathbf{f}_j$ is $1 \times n$, and $\mathbf{f}_j$ represents the feasibility status of all MESs for UE j. For example, $\mathbf{f}_j = [0, 0, 0, 0, 1, 0, 0, 0, 0, 1]$ means that only MES-5 and MES-10 are feasible for UE j. Therefore, we can serve UE j only at MES-5 and MES-10.

D. ENERGY AWARE MES PRIORITY

The total number of resources of MES k is defined as the capacity, C_P_k, and can be calculated as

$$C\_P_k = \gamma_1 c_{k_{total}} + \gamma_2 r_{k_{total}} + \gamma_3 h_{k_{total}}, \tag{23}$$

where $c_{k_{total}}$, $r_{k_{total}}$ and $h_{k_{total}}$ are the total CPU, RAM, and disk space, respectively, of MES k. Since different MESs may have different capacities and types of hardware, therefore, the energy consumption per unit time of an MES for keeping itself ON (activated) is different from other MESs. If E_k is the energy consumption per unit time in keeping MES k ON, then p_k, the energy consumption per unit capacity of MES k, can be written as

$$p_k = \frac{E_k}{C\_P_k}. \tag{24}$$

An MES with lower value of p_k means higher capacity and lower energy consumption for keeping it ON. Thus, it should be employed more often as compared to MESs with higher p_k.

To conserve energy, we need to avoid the activation of idle MESs as long as possible. We also need to prioritize the already ON MESs for entertaining new UE requests. If the currently active servers are not enough to serve the new incoming traffic, then the next more profitable idle MES is activated with some decrease in the utility as

$$u'_{jk} = \begin{cases} u_{jk} - \gamma_5 p_k & s_k = 0 \\ u_{jk} & s_k = 1 \end{cases}, \tag{25}$$

where $s_k = 0$ means that the MES k is idle and $s_k = 1$ means that the MES k is already in activated state. γ_5 is the unit balancing coefficient, given as

$$\gamma_5 = \frac{w_5}{e_{max}}, \tag{26}$$

where w_5 is the weighting coefficient and its value can be adjusted to make the threshold for activating an idle server higher or lower. e_{max} is the sum of the E_k for all servers, which can be written as

$$e_{max} = \sum_{k=1}^{n} E_k. \tag{27}$$

The performance metrics considered in this paper are service rate (SR), utility (u_{total}), energy consumption per unit utility (e_{puu}), and utility per unit active time (u_{put}) of the MES. The service rate (S_R) is defined as follows

$$S_R = \frac{N_{R_S}}{N_{R_T}} \times 100, \tag{28}$$

where N_{R_T} represents the total number of UE requests, and N_{R_S} represents the number of requests served.

Similarly, the total utility of all the servers is calculated as

$$u_{total} = \sum_{k=1}^{n} u_{k_{total}}, \tag{29}$$

where $u_{k_{total}}$ is the utility of server k for entertaining UE requests that are assigned to it.

Considering only CPU as requested resource is not a practical approach because the nature of application can have a significant effect on the amount and type of resources it needs. Most of the web applications and file hosting applications do not require much of the CPU but need a relatively larger disk space, whereas most of the speech processing applications require a larger amount of CPU than other resources. A video processing application may require another combination of resources. Therefore, CPU, RAM, and disk space should also be considered as requested resources, because their utilization consume different amount of energy of an MES. The energy consumption due to the usage of CPU, RAM, and disk space depends on the instruction type and architecture of the system used in an MES. We assume linear relation between the energy consumption and usage of CPU, RAM, and disk space [35].

The energy consumption due to CPU usage, RAM usage, and disk space usage of MES k, i.e., E_{ck}, E_{rk}, and E_{hk}, respectively, can be given as

$$E_{ck} = E_{ckmin} + (E_{ckmax} - E_{ckmin})G_c, \tag{30}$$

$$E_{rk} = E_{rkmin} + (E_{rkmax} - E_{rkmin})G_r, \tag{31}$$

$$E_{hk} = E_{hkmin} + (E_{hkmax} - E_{hkmin})G_h, \tag{32}$$

where E_{ckmin}, E_{rkmin}, and E_{hkmin}, respectively, represent the energy consumption when the CPU, RAM, and disk space are not in use. Similarly, E_{ckmax}, E_{rkmax}, and E_{hkmax} represent the energy consumption when the CPU, RAM, and disk space, respectively, are fully used. G_c, G_r, and G_h are the utilization of CPU, RAM, and disk space, respectively. The total energy consumption, $E_{k_{total}}$, of MES k, is calculated as

$$E_{k_{total}} = E_k.t_{k_{active}} + E_{ck} + E_{rk} + E_{hk}, \tag{33}$$

where $t_{k_{active}}$ is the total time for which the MES k was active.

The total energy consumption of all the MESs, E_{total}, can be written as

$$E_{total} = \sum_{k=1}^{n} E_{k_{total}}. \tag{34}$$

The total energy consumption alone cannot give a realistic measure of performance of an algorithm since the utility of an algorithm may also increase, and the increase in utility might be sufficient to justify the energy consumed in entertaining UE requests. Hence, e_{puu} is used as a performance metric instead of energy consumption alone, and is defined as the ratio of the total energy and total utility. It can be calculated as

$$e_{puu} = \frac{E_{total}}{u_{total}}. \tag{35}$$

The last performance metric, u_{put}, shows the type of UE requests that an algorithm prioritizes over the others. This is because, for the same amount of server active time, different algorithms can have different utilities depending upon the amount of resources requested and the distance between the UE and MES. Thus, u_{put} can be defined as

$$u_{put} = \frac{u_{total}}{t_{active}}, \tag{36}$$

where t_{active} is the sum of server active time, $t_{k_{active}}$, for all the servers, and is defined as

$$t_{active} = \sum_{k=1}^{n} t_{k_{active}}. \tag{37}$$

IV. ALLOCATION SCHEMES

An allocation scheme refers to the algorithm that a central unit employs to assign UEs to MESs and to allocate specific amount of MES resources to UE requests. We improve upon the work of [12] by considering more realistic resources of MESs and the mobility of users in a multi-user multi-server scenario. We propose two new algorithms, namely, *PowMigExpand* and Energy Efficient Smart Allocator (*EESA*) in this section.

A. BASIC OVER-PROVISIONING

The idea of first-come first-served is used in Basic Over-provisioning (*BO*) [12]. *BO* allocates the maximum requested resources to all UEs until MES runs out of resources. It activates the MESs in the increasing order of their p_k and creates the VM of the incoming UE request at the first available MES.

B. GREEDY MAX

The Greedy Max (*GM*) [12] first sorts the incoming UE requests in the decreasing order of their utility. For example, if there are 5 UE requests, *BO* allocates the maximum requested resources to all of them regardless of any priority, whereas *GM* first allocates the maximum requested resources to those UEs having higher utility. The difference between the performance of *BO* and *GM* is clear in times of high traffic when the MESs start filling up and some of the UEs are denied service. The UEs that are denied service in case of *GM* will always be the ones that offered the lowest utility.

C. MINIMUM EXPAND

The problem with *BO* and *GM* is that even in times of high traffic they keep allocating the maximum requested resources to some UEs and keep denying service to all the others. Minimum Expand (*MinExpand*) [12] tries to solve this problem by allocating the minimum requested resources to UEs for the full time that they requested resources for, and later allocating them more only if there is still room available at the MES after giving the minimum resources to all incoming UE requests. In this way, *MinExpand* gives service to a lot more UEs in times of high traffic than *BO* and *GM*. *MinExpand* is also greedy in nature since it follows the same principal as *GM* for sorting when it is expanding the existing VMs. It allocates the minimum resources to all, but then expands them in the order of their decreasing utility. In this manner, the more profitable VMs get expanded first, and the least profitable later. The expansion takes place until there is room on the MES so if some of the VMs do not get expanded because of the server running out of resources to allocate, they will always be the least profitable VMs. The expansion takes place until the UEs requested maximum has reached or the server runs out of resources, whichever happens first.

D. POWER MINIMUM EXPAND

All the previous algorithms ignore the energy consumption of MES in activating them for any UE request regardless of its utility. For example, when the most profitable server runs out of resources, they turn the next server ON without checking if the utility from new request is beneficial or not. Power Minimum Expand (*PowExpand*) sets a certain threshold of utility to activate an idle server [12]. Hence, *PowExpand* employs the penalized utility function given by (25). If an MES is in idle state, it subtracts a penalty term from the utility function, so that servers that are already in the ON state are prioritized over the idle server.

If the utility from new incoming UE is smaller than the threshold of activating an MES, it will be denied for service. However, the parameter γ_5 can be adjusted to put a stricter or lighter emphasis on threshold.

E. POWER MIGRATION EXPAND

All the previous algorithms ignore the VM migration on the basis of mobility of UEs and utility for MESs. We propose Power Migration Expand (*PowMigExpand*) that considers UE's mobility by taking their location into account in every time slot. Our utility funciton depends upon the distance between UE and MES, therefore, our VM migration is more beneficial in terms of utility. *PowMigExpand* sorts MESs in the increasing order of p_k and calculates the utility from each UE to each MES (hence, the matrix $\mathbf{U}$ given in (16)). Then sorts UEs in the decreasing order of utility and assigns them on the server for which the corresponding utility is the highest. This allocation uses the penalized utility function so that the servers that are already ON are prioritized. The distance matrix of the UEs is updated in every time slot and, hence, their utility for each server is also calculated again in every time slot. When a UE moves away from one MES and becomes closer to another, its utility for the nearest MES becomes the highest and the central control unit migrates its VM to that MES. Such frequent migrations cause an additional overhead on the system, so we need to keep these migrations to a minimum. Hence, our algorithm also sets a threshold on the utility and the remaining time for a request to be completed for VM migration. If the remaining time for the request to be completed is less than 2 time slots then the VM is not migrated.

Algorithm 1 explains *PowMigExpand*. The first part selects the optimal server for each incoming UE request and allocates the minimum requested resources to them. The second part checks for mobility of users and moves them to the new optimal server if needed. The third part expands the VMs, prioritizing the more profitable VMs to their maximum requested resources if there is room available on the server.

The algorithms that do not offer VM migration suffer from a certain disadvantage, i.e., if a UE is too far away from an MES and the allocation scheme does not incorporate

Algorithm 1 PowMigExpand

```
1:  Sort all MES into increasing order of $p_k$
2:  for time $i = 1, 2, \ldots, t$ do
3:      Sort all incoming UEs into decreasing order of
        requested resources.
4:      for UE $j = 1, 2, \ldots, m$ do
5:          Compute feasibility of MES from (21)
6:          Compute penalized utility for UE $j$ at feasible
            MESs from (25).
7:          Find the optimal MES for UE $j$ from (20).
8:          Create VM for UE $j$ at MES $k$ for time $t_j$.
9:          Allocate minimum requested resources.
10:         Update overall utility, service rate, and resource
            usage.
11:         break;
12:     end for
13:     for MES $k = 1, 2, \ldots, n$ do
14:         Check distance of all existing UEs from MES $k$.
15:         if the UE is moving then
16:             Compute feasibility of MES again from (21)
17:             Compute penalized utility again at feasible
                MESs from (25).
18:             Find the new optimal MES.
19:             if $s^*_{j_{new}} \neq s^*_j$ AND remaining time$> 2$ then
20:                 Migrate VM to $s^*_{j_{new}}$ for remaining time.
21:                 Update overall utility, and resource usage
                    of both MESs.
22:             end if
23:         end if
24:         Sort all VMs at MES $k$ into decreasing order of
            their $u_{j_{max}}$.
25:         while $c_{k_{av}} > 0.1c_j$ do
26:             for sorted VMs $j = 1, 2, \ldots$ at MES $k$ do
27:                 Expand VM $j$ to its maximum.
28:                 Update overall utility and resource usage.
29:             end for
30:         end while
31:     end for
32:     Bring the MESs with zero usage to power save mode.
33: end for
```

VM migration, the UE is simply disconnected from the MES instead of being moved to the other MES. As a result, the service rate of all the algorithms that do not consider VM migration is expected to be lower than *PowMigExpand*. The implementation complexity of the proposed algorithm is the same as the existing algorithms because every algorithm has to find the best allocation option for m UEs with n available MESs. There are n^m total possible options from which an allocation scheme selects the best option in terms of energy consumption, service rate, and utility for MES. Since our algorithms considers more realistic resources of an MES (CPU, RAM, disk space), the feasibility of an MES, and the threshold utility to take decision of activating an idle MES, therefore, the proposed algorithm has high computational overhead as compared to the existing algorithms. To avoid such computational overhead, we propose a deep learning approach in which we train a DNN to reduce the complexity as a multiplication problem $O(mn)$. The computational complexity is high only in training phase of the network. Once the DNN is trained, the complexity becomes a simple multiplication problem [36].

F. ENERGY EFFICIENT SMART ALLOCATOR

The complexity of choosing the optimum MES k for UE j for m number of UEs and n number of servers is $O(n^m)$ and that will put an additional overhead on the system. In most practical scenarios where the numbers of UEs and MESs are large, the overhead increases exponentially and creates significant delays in choosing the optimal MES. These delays can be avoided by utilizing the potential of machine learning and neural networks that have the ability to learn the behavior of a decision maker and can act in its place for future unseen data [36]–[38]. Deep learning algorithms can learn complex decision boundaries [39] and complicated patterns in the data and that motivates us to design a deep learning algorithm for efficient resource allocation. Such type of learning is called supervised deep learning. In supervised deep learning, we need a correct dataset for training of the neural network. We use our comprehensive mathematical model of UE request and utility function to generate the training dataset for neural network. Since our mathematical model considers the mobility of UEs and takes into account the energy consumption of MESs, we named this scheme as Energy Efficient Smart Allocator (EESA). We train the EESA over training dataset and after training it is able to predict the labels for unseen test data with a degree of certainty. In doing so, a pre-trained network only has the complexity of $O(mn)$ for deciding which MES is optimal for the UE request.

The dataset is created by running the simulations over *PowMigExpand* and storing the UE requests and the MES resource usage state as the training data. The labels for this training data are the optimal MESs selected by *PowMigExpand*. We divide the total available data into training, validation, and test data parts according to 70%, 15%, and 15% ratios; and train three different networks over this dataset.

There are three type of layers in a neural network; input layer, hidden layer, and output layer, as shown in Figure 3 [39]. Input layer takes the input data for which the decision is required. In our case, this data is the number of MESs, number of UEs, requested resources by UEs, available resources of MESs, distance matrix, and information about the corresponding utilities. The requested resources and its corresponding utility value need 7 neurons (c_{min}, c_{max}, r_{min}, r_{max}, h, t, u_{max}) in the input layer. Similarly, we consider 10 MESs, therefore, distance matrix for a single UE, CPU utilization of 10 MESs, RAM utilization of 10 MESs, and disk space utilization of 10 MESs each need 10 neurons in the input layer. Therefore, we need 47 neurons in the input layers for training, as depicted in Figure 4.

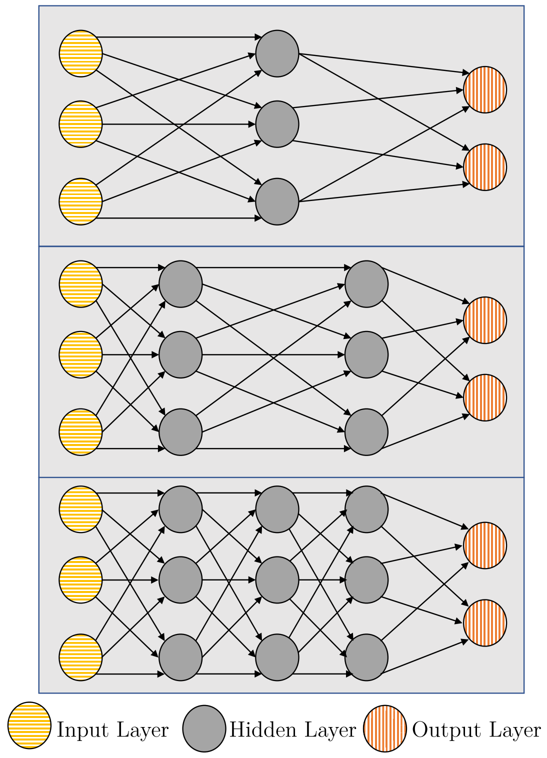
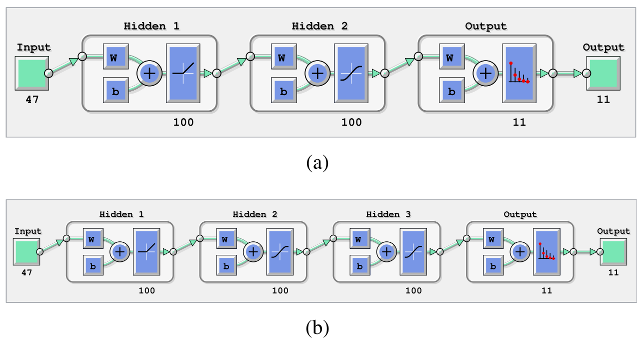


FIGURE 3. **A single layer, two layer, and three layer neural network [39].**



(b)

FIGURE 4. **(a) The DNN with two hidden layers, (b) The DNN with three hidden layers.**

We take 100 neuron per each hidden layer. The output layer takes the desired correct decision as labels during training. In our case, these labels are the optimal MESs for UEs. As we consider 10 MESs, therefore, we choose 11 neurons in the output layer; one for each MES and last one for busy situation. The situation will be busy if all the MESs are fully loaded. After training, the trained network calculates the desired label for unseen input data. The hidden layers are used to find the relation and links between output data and input data.

We use Softmax activation function [36] for the training of our networks. The first network is the simplest artificial neural network with a single hidden layer. Most of the time, such network is not able to model complex decision boundaries and fails to learn the complicated patterns in the data. One way to improve the performance is to increase the

TABLE 4. Simulation Parameters.

Parameter	Value
d_{max}	1000 m
γ_1	0.4
γ_2	0.25
γ_3	0.25
γ_4	0.1
γ_5	5
t_{max}	10

number of hidden layers (deep learning). Thus, the second and third network that we use have two and three hidden layers respectively, as shown in Figure 4.

V. PERFORMANCE EVALUATION

A. SIMULATION SETUP

We use MATLAB (R2019a) on Intel Core i7 CPU @3.4GHz for simulations. User request arrivals are modeled as a Poisson process with mean 5 for all simulations except for varying incoming traffic conditions. Similarly, the number of MESs is kept 10 except for simulations where we try to see the effect of varying the total number of MESs. The results of the resource allocation schemes are recorded for 1000 time slots. The amount of CPU, RAM, and disk space at MESs follow a Normal distribution with mean and variance 15 and 5; 10 and 2; and 25 and 5, respectively. The energy consumption per unit time, E_k, to keep an MES ON is proportional to the amount of resources of an MES. The coverage range of an MES is assumed to be up to 800 m. Since the users are mobile and the distance between users and MESs may change in every time slot, therefore, the distance between users and MESs is taken as a uniform random variable with $d \in [1, 1000]$.

The effect of mobility is incorporated into *PowMigExpand*, i.e., the locations of UEs do not remain constant throughout the application execution. Instead, some of the UEs are assumed to be stationary and some are assumed to be mobile and their distances from each MES are updated in every time slot when their requests are being served at the MES. As a result, some of the UEs move too far away from the MES. Consequently, the algorithms that do not migrate their VMs to the new optimal server cannot complete their requests. We simulate this scenario by disconnecting UEs from MES once they cross a certain distance threshold for the algorithms that do not incorporate VM migration. We consider utility, service rate, and energy consumption of MESs as the performance metrics, which have been derived in Section III. The effects of varying traffic and varying total number of MESs on the performance metrics are presented next.

B. UTILITY OF MESs

Figure 5 shows the utility of different algorithms against different levels of traffic. As the number of incoming requests increases, the utility for all the algorithms increases because

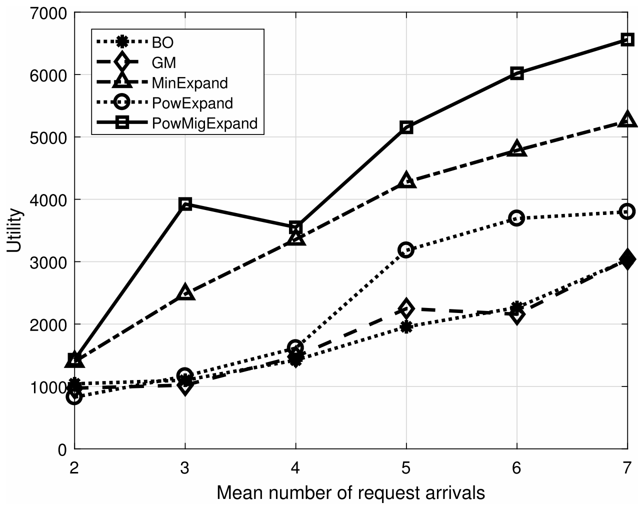


FIGURE 5. Utility of the considered algorithms for varying levels of incoming UE requests.

the server provides services to more number of users. However, the performance of *PowMigExpand* is better than the other algorithms because of its mobility-awareness and a higher service rate. As the number of request arrivals is a random process and also the amount of requested resources is random, therefore, in Figure 5, at mean equal to 4 the utility of proposed algorithm decreases because the VM migration and the activating of idle MESs decrease the utility of the proposed algorithm. Due to these reasons, the proposed algorithm assigns minimum resources to the UEs for entertaining maximum UEs. Therefore, with minimum resources the utility may decrease. Other algorithms do not consider the VM migration so, their utility does not decrease at mean equal to 4.

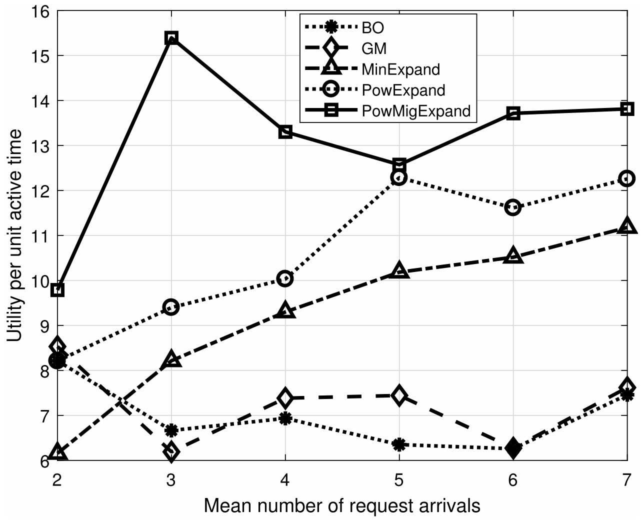


FIGURE 6. Utility per unit server active time of the considered algorithms for varying levels of incoming UE requests.

Figure 6 shows utility per unit active time against different levels of incoming traffic. All the algorithms perform well under low traffic conditions, whereas under high traffic conditions the performance of the algorithms that do not take mobility into account degrades. For the same amount of server active time, *PowMigExpand* provides service to the users with the higher utility, and hence, results in a higher utility per unit server active time. The sudden increase in utility at mean equal to 3 implies the shorter active time of the servers because the proposed algorithm also checks for the feasibility of servers and tries to avoid the idle servers.

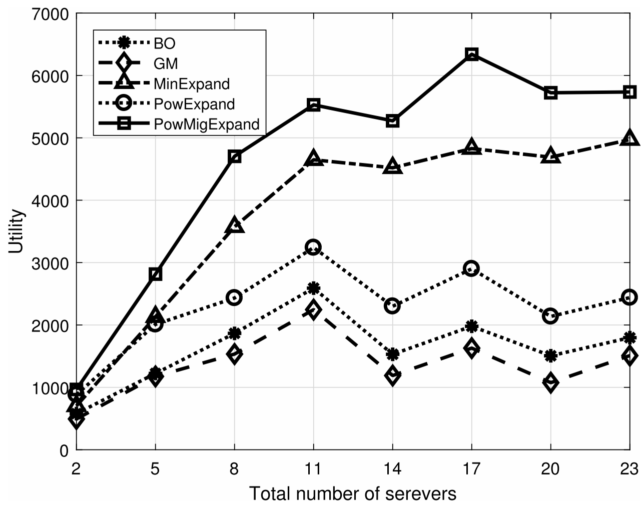


FIGURE 7. Utility of the considered algorithms for different number of MESs.

In Figure 7, we present the utility of all the algorithms against different number of servers. It can be observed from the figure that as the number of servers increases, the utility also increases because more number of requests can be entertained on a higher number of servers. However, *PowMigExpand* outperforms all the other algorithms because of its higher service rate.

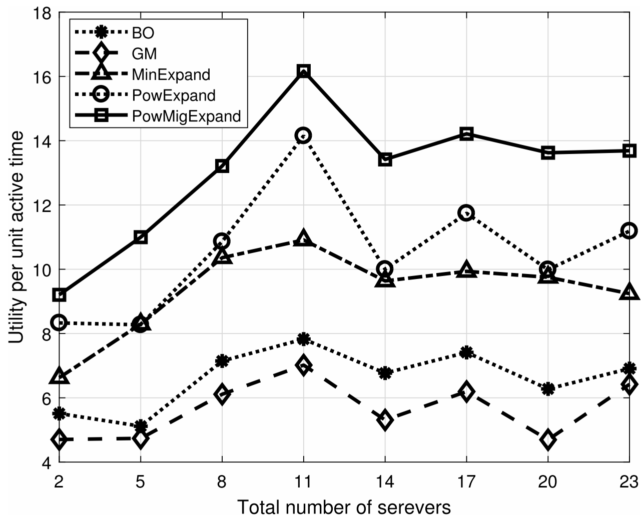


FIGURE 8. Utility per unit server active time of the considered algorithms for different number of MESs.

Figure 8 depicts the variation of utility per unit active time against the number of servers for all the algorithms. As the number of servers increases, the utility as well as the server active time increases for all the algorithms. For the same amount of server active time, however, *PowMigExpand* has a higher utility as compared to the other algorithms. This is because *PowMigExpand* is mobility-aware, and provides service to the UE requests that are closest to the server, resulting in a higher utility for the same amount of server active time. The utility per unit server active time is low for the other algorithms because the server remains active even when the user has been disconnected from it. The sudden increase at total number of servers equal to 11, in Figure 8, is due to the random high traffic for short active time. It means that when we have 11 MESs, it may be possible to get traffic for a short time, therefore, the utility per unit active time increases suddenly.

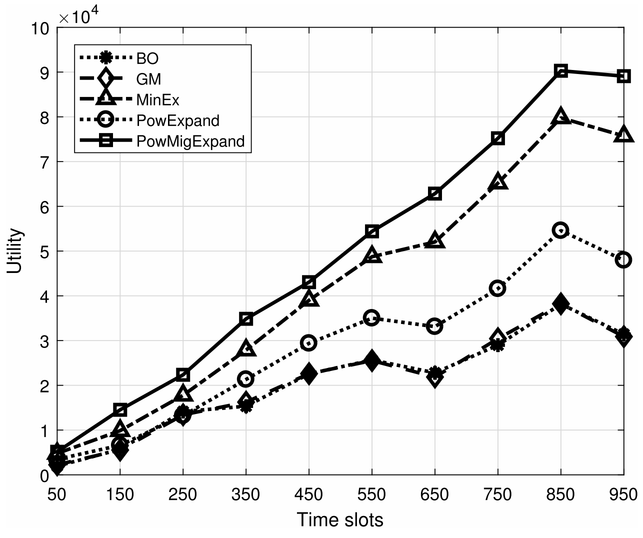


FIGURE 9. Utility of the considered algorithms for total time for which UE requests were taken.

Figure 9 presents the plot of utility against the total time. As depicted int the figure, when the total time increases, the number of UEs and their requested resources also increase. As a result, the utility of all the algorithms increases. The utility of our proposed algorithm is higher than others for all time slots, showing that *PowMigExpand* selects the optimal combination of UE requests to serve.

Figure 10 shows the utility per unit active time against the total time for which the algorithms run. As the time duration of the algorithms increases, the server active time also increases because of providing service to more users. Therefore, the utility per unit active time may increase or decrease depending upon the number of users entertained. The utility rate of our *PowMigExpand* is higher as compared to the other algorithms, because our algorithm considers VM migration during mobility of users and gets a higher service rate. Higher service rate indicates that *PowMigExpand* provides service to a higher number of users as compared to the other algorithms. Therefore, the utility per unit active time is high for *PowMigExpand*. The sudden changes in the trends

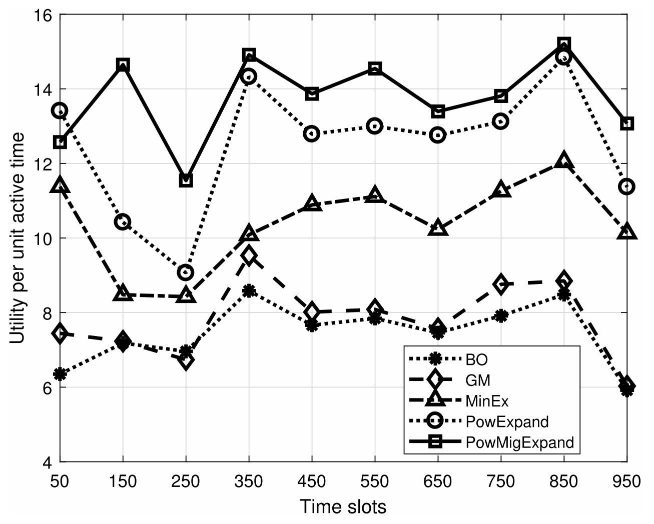


FIGURE 10. Utility per unit server active time of the considered algorithms for total time for which UE requests were taken.

in Figure 10 show the stochastic nature of the incoming traffic and requested resources.

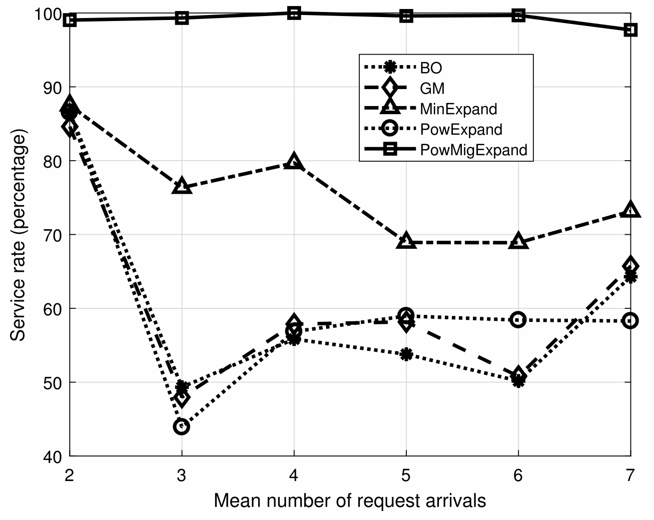


FIGURE 11. Service rate of the considered algorithms for varying levels of incoming UE requests.

C. SERVICE RATE

Figure 11 shows the service rate for different algorithms against varying levels of incoming traffic. It can be observed from the figure that all the algorithms perform well under low traffic conditions. However, under high traffic conditions there are frequent disconnections in case of the algorithms that do not take mobility into account. The x-axis represents the mean of the Poisson process according to which the user arrivals are modeled. A higher mean represents high traffic, whereas a low mean represents low traffic. The service rate of *PowMigExpand* decreases slightly under high traffic conditions because it does not activate idle servers for users with low utility. However, the service rate of *PowMigExpand* is higher than other algorithms for all levels of traffics. The quick decrease in service rate at mean equal to 3 for the algorithms other than *PowMigExpand* is due to the disconnectivity of UEs because they do not consider the VM migration.

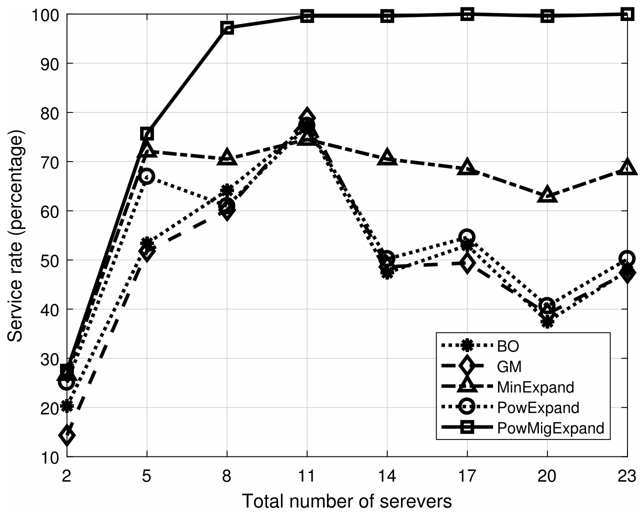


FIGURE 12. Service rate of the considered algorithms for different number of MESs.

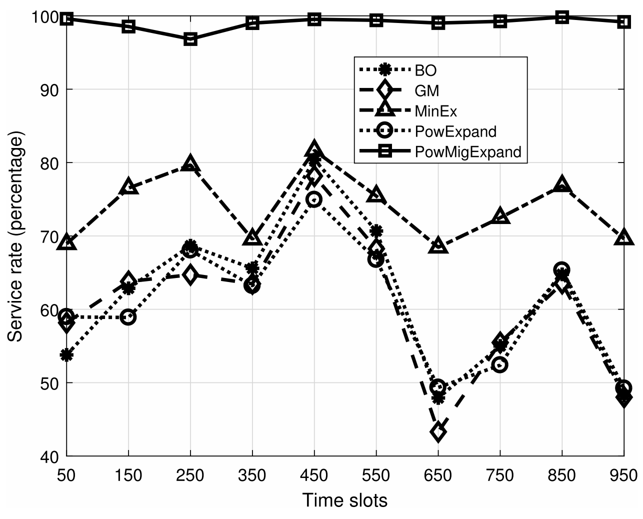


FIGURE 13. Service rate of the considered algorithms for total time for which UE requests were taken.

Figure 12 presents service rate of all the algorithms against varying number of total servers. As depicted in the figure, the service rate is low for small number of servers for all the algorithms because the resources of the servers are not enough to provide service to all the incoming UE requests. As the number of servers increases, the service rate of all the algorithms starts improving. When the number of servers is sufficient to provide service to all the UE requests, the service rate of *PowMigExpand* is higher than the other algorithms because it considers the mobility of users and migrates their requests to other servers instead of disconnecting them when they move. Figure 13 shows the service rate of all the algorithms against the total time for which the algorithms run.

The service rate of *PowMigExpand* is much higher than all the other algorithms for all time slots. This is because *PowMigExpand* takes the mobility of the UEs into account, while the other algorithms assume all the UEs to be stationary. As a result, the UEs that are moving get disconnected in case of all the other algorithms. The significant difference in the service rate of *PowMigExpand* and the other algorithms indicates the importance of taking into account the mobility of users and VM migration. *PowMigExpand* provides service to all UEs regardless of them being stationary or moving, whereas the other algorithms only provide services to stationary UEs. The sudden decreasing trend at different points in Figure 12 and Figure 13 shows the ignorance of the VM migrations in these algorithms.

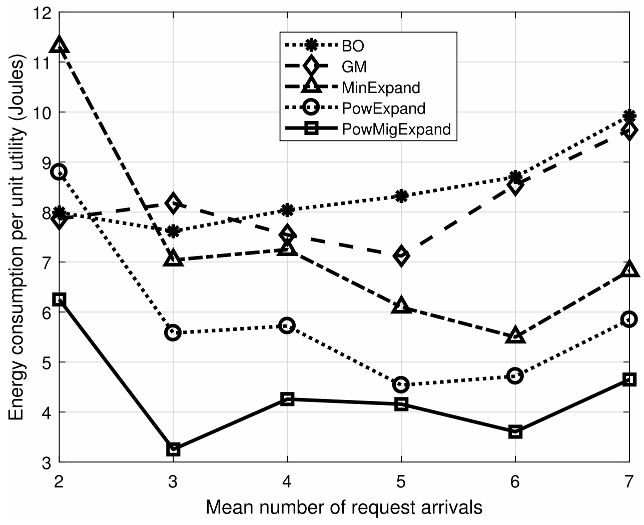


FIGURE 14. Energy consumption per unit utility of the considered algorithms for varying levels of incoming UE requests.

D. ENERGY CONSUMPTION OF MESs

Energy consumption per unit utility provides a meaningful measure of performance to compare different algorithms. Figure 14 presents the energy consumption per unit utility for all the algorithms against different levels of incoming traffic. All the algorithms perform well under low traffic conditions and their energy consumption per unit utility is quite low. However, under high traffic conditions, the algorithms that allocate the maximum requested resources to users have high energy consumption per unit utility ratio. This means that their energy consumption is high for a relatively low level of utility, which can be observed in the utility graph of Figure 5.

Figure 15 shows the energy consumption per unit utility against varying number of servers. As the number of servers increases, the utility as well as the energy consumption of all the algorithms increases. It can be observed that *PowMigExpand* outperforms all the other algorithms because of its higher utility for the same amount of energy consumption. The overall energy consumption of *PowMigExpand* is lowest because of its energy-aware server priority. A quick increase in the plots at MESs equal to 14 shows their decreasing utility at that point.

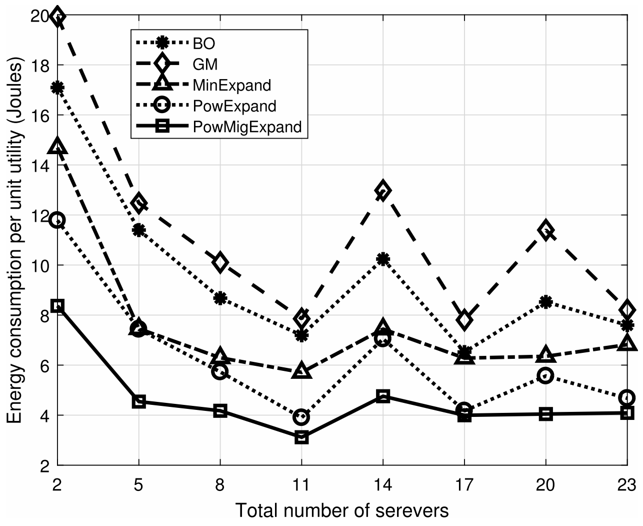


FIGURE 15. Energy consumption per unit utility of the considered algorithms for different number of MESs.

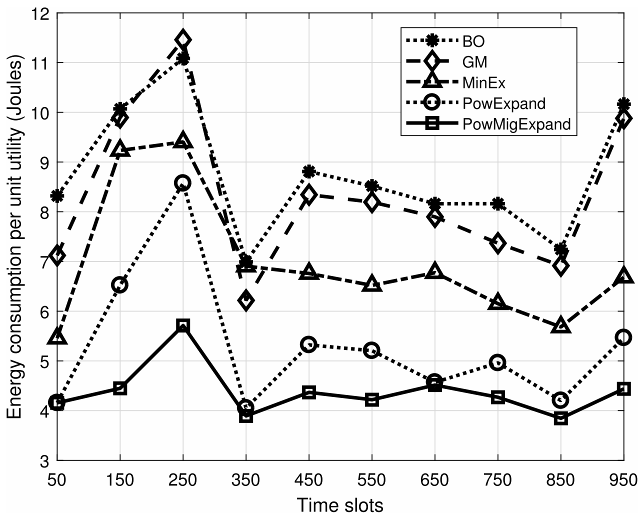


FIGURE 16. Energy consumption per unit utility of the considered algorithms for total time for which UE requests were taken.

Figure 16 shows the variation of energy consumption per unit utility for varying total time for which the UE requests are entertained. When an algorithm runs for a long time, it provides service to more number of UEs and the total energy consumption also increases. The increase in utility is achieved at the cost of the increase in total energy consumption. We can observe from the figure that the ratio of energy consumption per unit utility is low for *PowMigExpand* as compared to the other algorithms. This is because *PowMigExpand* considers the energy per unit capacity for activating an idle server. Our proposed algorithm activates a server only if the reward for activating the server exceeds the cost for turning it ON by setting a utility threshold. Our approach also considers the mobility of UEs, and hence, it provides a better utility for the same amount of energy consumed. Since the number of UEs and their requested resources are random, the algorithms perform the worst when the total time is 250 time slots. This is because of a large number of users requesting a small amount of resources and being located far away from the servers, generating comparatively small utility for a higher energy consumption. The random trends in plots of Figure 16 show the random nature of incoming UE requests and their utility.

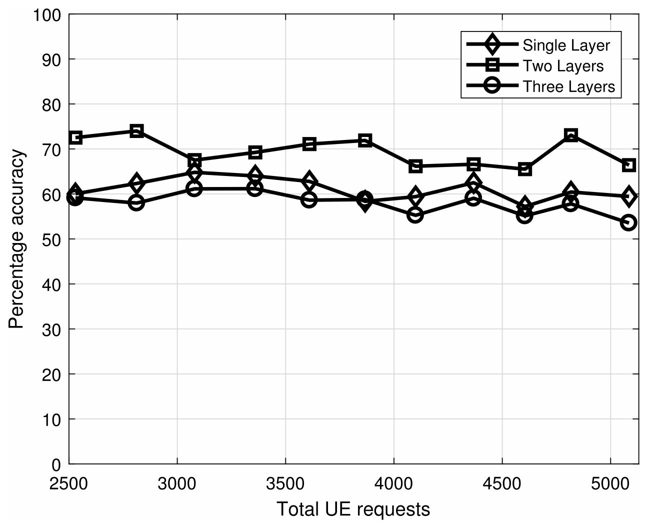


FIGURE 17. Accuracy of neural networks for different sizes of training dataset used in *PowMigExpand*.

E. ACCURACY OF EESA

We get more than 70% accuracy of EESA in deciding the assignment of UEs to the optimal MESs. EESA is able to handle the decision making process of selecting the optimal server for each incoming UE request. MATLAB Deep Learning Toolbox has been used for the results presented in this subsection. Figure 17 shows the performance of three different neural networks for different data sizes (total number of UE requests). Since the results of *PowMigExpand* are better than the other algorithms, in terms of utility, energy consumption, and service rate, we use *PowMigExpand* to generate the training dataset for EESA. These different sizes of datasets are obtained by running *PowMigExpand* for different amounts of time. The figure shows that a two layered network is the optimal choice for this type of data. The input to the network is the state vector containing the incoming request data and the MES usage data, and the output is the optimal MES that selected by *PowMigExpand* for the UE request. The poor performance of a single layer and a three layer network is due to under fitting and over fitting problems. Once trained over a sufficiently large dataset, the network can be used for all future decisions.

F. CRITICAL DISCUSSION

The problem of MEC server resource allocation is a complex multidimensional one and most of the traditional approaches focus only on meeting the user requirements regardless of the energy consumption on the MES side, leading to a significant waste of resources and a large carbon footprint. Most of the literature considers CPU only as UE request and take utilities only dependent on CPU. Similarly static allocation of resources is not an optimal solution due to the mobile nature of UEs and thus, VM migration is necessary but excessive use of it puts an additional load on the server.

The simulation results show better performance of the proposed *PowMigExpand* resource allocation algorithm in terms of service rate, utility, and overall energy consumption. The simulation results show the effect of varying traffic and varying total number of MESs on these performance metrics. Therefore, the better performance of our algorithm for different number of total MESs and varying traffic shows the scalability of our proposed algorithm. This algorithm provides service to the maximum number of users, keeping the MES utility high, and the overall energy consumption to the lowest possible. The better service rate of *PowMigExpand*, as compared to the other algorithms, is because it takes UE mobility into account. In this way, *PowMigExpand* maximizes the server utility, minimizes the energy consumption and provides the highest service rate to the users. The mobility-awareness and energy awareness of the *PowMigExpand* algorithm make it the most feasible algorithm for implementation in real-world scenarios. Since the calculations involved in selecting the optimal server and allocating the optimal amount of resources put additional overhead on the allocation schemes, we have also presented smart allocator (EESA) that uses machine learning for the resource allocation problem and reduces the number of computations. A two-layer neural network has been found to be the appropriate choice for this type of resource allocation.

The simulation results indicate that the performance of all benchmark algorithms is comparable under low traffic conditions but in case of high traffic conditions and high UE mobility, the allocation schemes that are not energy and mobility-aware do a poor job of allocating resources to the incoming requests. Our algorithm performs consistently better than all the other algorithms under all traffic conditions as well as under the varying number of servers. However, EESA is unable to handle the task of deciding where and when to migrate a VM. Our future work will seek to address this issue.

VI. CONCLUSION

The resource allocation problem in MEC is of great importance. Most of the related work considers only CPU resources in calculating the utilities while ignoring the UE mobility. Thus, an excessive use of VM migration may increase the load on MES. We have proposed a balance approach that allows VM migration only when the rewards justify the cost. Our proposed algorithm, *PowMigExpand*, uses conservative activation of MESs that minimizes the energy consumption. Moreover, *PowMigExpand* maximizes the utility for MESs and service rate for UEs. Our algorithm handles a large number of UEs by reallocating MES resources according to the changes in the incoming requests and outperforms eminent approaches under high load conditions. We also propose a deep learning based energy efficient smart allocator algorithm, EESA, to solve the multidimensional optimization problem numerically and reduce the computational overhead by using a pre-trained network for decision making. We get more than 70% accuracy for EESA even in high traffic conditions. As a future work, the joint task of computational offloading and resource allocation can be handed over to the smart allocator.

REFERENCES

[1] P. Bhattacharya, S. Tanwar, R. Shah, and A. Ladha, ‘‘Mobile edge computing-enabled blockchain framework—A survey,’’ in *Proc. ICRIC*, 2020, pp. 797–809.

[2] D. Dechouniotis, N. Athanasopoulos, A. Leivadeas, N. Mitton, R. Jungers, and S. Papavassiliou, ‘‘Edge computing resource allocation for dynamic networks: The DRUID-NET vision and perspective,’’ *Sensors*, vol. 20, no. 8, p. 2191, Apr. 2020.

[3] T. Dreibholz, S. Mazumdar, F. Zahid, A. Taherkordi, and E. G. Gran, ‘‘Mobile edge as part of the multi-cloud ecosystem: A performance study,’’ in *Proc. 27th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Feb. 2019, pp. 59–66.

[4] C. Sun, H. Li, X. Li, J. Wen, Q. Xiong, and W. Zhou, ‘‘Convergence of recommender systems and edge computing: A comprehensive survey,’’ *IEEE Access*, vol. 8, pp. 47118–47132, 2020.

[5] W. Sun, J. Liu, and H. Zhang, ‘‘When smart wearables meet intelligent vehicles: Challenges and future directions,’’ *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 58–65, Jun. 2017.

[6] F. Zhou, R. Q. Hu, Z. Li, and Y. Wang, ‘‘Mobile edge computing in unmanned aerial vehicle networks,’’ *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 140–146, Feb. 2020.

[7] P. Sasikala, ‘‘Virtues and Services of Mobile Cloud Computing,’’ *Purakala CARE J.*, vol. 31, no. 15, pp. 453–457, 2020.

[8] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. Fitzek, ‘‘Device-enhanced mec: Multi-access edge computing (MEC) aided by end device computation and caching: A survey,’’ *IEEE Access*, vol. 7, pp. 166079–166108, 2019.

[9] M. Alaa, A. A. Zaidan, B. B. Zaidan, M. Talal, and M. L. M. Kiah, ‘‘A review of smart home applications based on Internet of Things,’’ *J. Netw. Comput. Appl.*, vol. 97, pp. 48–65, Nov. 2017.

[10] I. Sittón-Candanedo, R. S. Alonso, Ó. García, A. B. Gil, and S. Rodríguez-González, ‘‘A review on edge computing in smart energy by means of a systematic mapping study,’’ *Electronics*, vol. 9, no. 1, p. 48, Dec. 2019.

[11] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, ‘‘Edge computing: A survey,’’ *Future Gener. Comput. Syst.*, vol. 97, pp. 219–235, Aug. 2019.

[12] M. Cardosa, M. R. Korupolu, and A. Singh, ‘‘Shares and utilities based power consolidation in virtualized server environments,’’ in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, Jun. 2009, pp. 327–334.

[13] M. Abu Sharkh, M. Jammal, A. Shami, and A. Ouda, ‘‘Resource allocation in a network-based cloud computing environment: Design challenges,’’ *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 46–52, Nov. 2013.

[14] K. Peng, V. C. M. Leung, X. Xu, L. Zheng, J. Wang, and Q. Huang, ‘‘A survey on mobile edge computing: Focusing on service adoption and provision,’’ *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–16, Oct. 2018, doi: 10.1155/2018/8267838.

[15] 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, no. 1, pp. 149623–149633, 2019.

[16] Z. Yang, C. Pan, J. Hou, and M. Shikh-Bahaei, ‘‘Efficient resource allocation for mobile-edge computing networks with NOMA: Completion time and energy minimization,’’ *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7771–7784, Nov. 2019.

[17] L. Zhao and J. Liu, ‘‘Optimal placement of virtual machines for supporting multiple applications in mobile edge networks,’’ *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6533–6545, Jul. 2018.

[18] S. Filiposka, A. Mishev, and K. Gilly, ‘‘Mobile–aware dynamic resource management for edge computing,’’ *Trans. Emerg. Telecommun. Technol.*, vol. 30, no. 6, p. e3626, Jun. 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3626

[19] J. Wang, L. Zhao, J. Liu, and N. Kato, ‘‘Smart resource allocation for mobile edge computing: A deep reinforcement learning approach,’’ *IEEE Trans. Emerg. Topics Comput.*, early access, Mar. 4, 2019, doi: 10.1109/TETC.2019.2902661.

[20] S. Vimal, M. Khari, N. Dey, R. G. Crespo, and Y. Harold Robinson, ‘‘Enhanced resource allocation in mobile edge computing using reinforcement learning based MOACO algorithm for IIOT,’’ *Comput. Commun.*, vol. 151, pp. 355–364, Feb. 2020.

[21] T. Dlamini and A. F. Gambin, ‘‘Adaptive resource management for a virtualized computing platform within edge computing,’’ in *Proc. 16th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2019, pp. 1–9.

[22] D. W. K. Ng, E. S. Lo, and R. Schober, ‘‘Energy-efficient resource allocation in OFDMA systems with large numbers of base station antennas,’’ *IEEE Trans. Wireless Commun.*, vol. 11, no. 9, pp. 3292–3304, Sep. 2012.

[23] D. W. K. Ng, E. S. Lo, and R. Schober, ‘‘Energy-efficient resource allocation in OFDMA systems with hybrid energy harvesting base station,’’ *IEEE Trans. Wireless Commun.*, vol. 12, no. 7, pp. 3412–3427, Jul. 2013.

[24] C. Li, W. Chen, H. Tang, Y. Xin, and Y. Luo, ‘‘Stochastic computation resource allocation for mobile edge computing powered by wireless energy transfer,’’ *Ad Hoc Netw.*, vol. 93, Oct. 2019, Art. no. 101897.

[25] B. Liu, H. Xu, and X. Zhou, ‘‘Resource allocation in wireless-powered mobile edge computing systems for Internet of Things applications,’’ *Electronics*, vol. 8, no. 2, p. 206, Feb. 2019.

[26] S. Jin, Q. Gu, X. Li, X. An, and R. Fan, ‘‘Energy-efficient resource allocation for mobile edge computing system supporting multiple mobile devices,’’ in *Proc. Int. Conf. Internet Things Service*. Xi’an, China: Springer, 2019, pp. 228–234.

[27] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, ‘‘Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN,’’ *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3282–3299, Apr. 2020.

[28] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, ‘‘Black-box and gray-box strategies for virtual machine migration,’’ in *Proc. 4th USENIX Conf. Networked Syst. Des. Implement.*, Berkeley, CA, USA, 2007, p. 17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973447

[29] D. Dhanya and D. Arivudainambi, ‘‘Dolphin partner optimization based secure and qualified virtual machine for resource allocation with streamline security analysis,’’ *Peer-to-Peer Netw. Appl.*, vol. 12, no. 5, pp. 1194–1213, Sep. 2019.

[30] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, ‘‘Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration,’’ *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1287–1300, Sep. 2018.

[31] K. Li, ‘‘A game theoretic approach to computation offloading strategy optimization for non-cooperative users in mobile edge computing,’’ *IEEE Trans. Sustain. Comput.*, early access, Sep. 5, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8454762

[32] F. Zafari, J. Li, K. K. Leung, D. Towsley, and A. Swami, ‘‘A game-theoretic approach to multi-objective resource sharing and allocation in mobile edge,’’ in *Proc. Technol. Wireless Edge Workshop*, 2018, pp. 9–13. [Online]. Available: https://dl.acm.org/doi/10.1145/3266276.3266277

[33] M. Zakarya, L. Gillam, H. Ali, I. Rahman, K. Salah, R. Khan, O. Rana, and R. Buyya, ‘‘EpcAware: A game-based, energy, performance and cost efficient resource management technique for multi-access edge computing,’’ *IEEE Trans. Services Comput.*, early access, Jun. 26, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9127143

[34] V. W. Wong, *Key Technology for 5G Wireless Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[35] X. Zhang, J. Lu, and X. Qin, ‘‘BFEPM: Best fit energy prediction modeling based on CPU utilization,’’ in *Proc. IEEE 8th Int. Conf. Netw., Archit. Storage*, Jul. 2013, pp. 41–49.

[36] Y. LeCun, Y. Bengio, and G. Hinton, ‘‘Deep learning,’’ *Nature*, vol. 521, no. 7553, p. 436, 2015.

[37] R. Collobert and J. Weston, ‘‘A unified architecture for natural language processing: Deep neural networks with multitask learning,’’ in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 160–167.

[38] E. Björnson and P. Giselsson, ‘‘Two applications of deep learning in the physical layer of communication systems,’’ 2020, *arXiv:2001.03350*. [Online]. Available: http://arxiv.org/abs/2001.03350

[39] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2924–2932.

ZAIWAR ALI received the B.S. degree in electronics engineering from the COMSATS Institute of Information Technology, Pakistan, in 2012, and the M.S. degree in electronics engineering from the GIK Institute of Engineering Sciences and Technology, Pakistan, in 2015, where he is currently pursuing the Ph.D. degree in electrical engineering with the Telecommunications and Networking (TeleCoN) Research Laboratory. From 2013 to 2015, he has worked as a Graduate Assistant with the GIK Institute of Engineering Sciences and Technology, where he has been working as a Research Associate with the Faculty of Electrical Engineering, since 2016. His research interests include stochastic processes, the IoT, machine learning, edge computing, and wireless networks. He was a recipient of the highest level of Merit Scholarship.

SADIA KHAF received the B.E. degree from the National University of Sciences and Technology, Islamabad, in 2015, and the M.S. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2018. From 2015 to 2018, she has worked as a Research Assistant with IONOLAB, Turkey. She is currently a Research Associate with the Faculty of Electrical Engineering, GIK Institute of Engineering Sciences and Technology, Pakistan. Her research interests include machine learning, edge computing, and radio resource management.

ZIAUL HAQ ABBAS received the M.Phil. degree in electronics from Quaid-e-Azam University, Pakistan, in 2001, and the Ph.D. degree from the Agder Mobility Laboratory, Department of Information and Communication Technology, University of Agder, Norway, in 2008. He joined the Ghulam Ishaq Khan (GIK) Institute of Engineering Sciences and Technology, Pakistan, as a Research Associate, where he is currently an Assistant Professor. In 2012, he was a Visiting Researcher with the Department of Electrical and Computer Engineering, University of Minnesota, USA. His research interests include energy efficiency in hybrid mobile and wireless communication networks, 4G and beyond mobile systems, mesh and ad hoc networks, traffic engineering in wireless networks, performance evaluation of communication protocols and networks by analysis and simulation, Quality-of-Service in wireless networks, green wireless communication, and cognitive radio.

GHULAM ABBAS (Senior Member, IEEE) received the B.S. degree in computer science from the University of Peshawar, Pakistan, in 2003, and the M.S. degree in distributed systems and the Ph.D. degree in computer networks from the University of Liverpool, U.K., in 2005 and 2010, respectively. From 2006 to 2010, he was a Research Associate with Liverpool Hope University, U.K., where he was associated with the Intelligent and Distributed Systems Laboratory. Since 2011, he has been with the Faculty of Computer Sciences and Engineering, GIK Institute of Engineering Sciences and Technology, Pakistan. He is currently working as an Associate Professor and the Director of the Huawei Network Academy. He is a Co-Founding Member of the Telecommunications and Networking (TeleCoN) Research Laboratory, GIK Institute. His research interests include computer networks and wireless and mobile communications. He is a Fellow of the Institute of Science and Technology, U.K., and the British Computer Society.

FAZAL MUHAMMAD received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2004 and 2007, respectively, and the Ph.D. degree in electronic engineering from the GIK Institute of Engineering Sciences and Technology, Pakistan, in 2017. He is currently working as an Assistant Professor and the Head of the Electrical Engineering Department, City University of Sciences and Information Technology, Peshawar. He is also the Secretary of Institutions of Engineers, Pakistan, Peshawar Center. His research interests include heterogeneous cellular networks, cognitive radio networks, and optical networks.

SUNGHWAN KIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Seoul National University, South Korea, in 1999, 2001, and 2005, respectively. He was a Postdoctoral Visitor with the Georgia Institute of Technology (GeorgiaTech), from 2005 to 2007, and a Senior Engineer with Samsung Electronics, from 2007 to 2011. He is currently a Professor with the School of Electrical Engineering, University of Ulsan, South Korea. His main research interests include channel coding, modulation, massive MIMO, visible light communication, and quantum information.

• • •