IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 March 2024, accepted 14 May 2024, date of publication 21 May 2024, date of current version 30 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3403721

RESEARCH ARTICLE

A Hyper-Heuristic Approach for Quality of Experience Aware Service Placement Scheme in 5G Mobile Edge Computing

SAFIQUL ISLAM[1], MAHADI AHAMMED[1], NURA ALAM SIDDIQUE[1], PALASH ROY[1,2], MD. ABDUR RAZZAQUE[2], (Senior Member, IEEE), MOHAMMAD MEHEDI HASSAN[3], (Senior Member, IEEE), AND KASHIF SALEEM[4]

[1]Department of Computer Science and Engineering, Green University of Bangladesh, Narayanganj, Dhaka 1461, Bangladesh
[2]Green Networking Research Group, Department of Computer Science and Engineering, University of Dhaka, Dhaka 1000, Bangladesh
[3]Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
[4]School of IT and Engineering, Melbourne Institute of Technology, Sydney, NSW 2000, Australia

Corresponding author: Md. Abdur Razzaque (razzaque@du.ac.bd)

This work was supported by the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia, under Project IFKSUOR3-282-2.

ABSTRACT The 5^{th} Generation (5G) Mobile Edge Computing (MEC) addresses the problem of high end-to-end delay experienced by traditional cloud computing users by ensuring fast accessible and reliable computing resources. However, the deployment of service instances in MEC resources requires migration due to user mobility. While Proactive Migration of service instances at multiple MECs increases users' Quality-of-Experience (QoE), Reactive Migration might reduce the deployment cost at the expense of user QoE. In this paper, we have developed a framework, that distributes service instances proactively among the Edge Nodes depending on user movement trajectories to ensure faster migration of the service instances and deliver higher QoE within minimum VNF deployment cost considering users' budgets. The aforementioned Proactive Service Placement (PSP) problem is formulated as a Multi-Objective Linear Programming (MOLP) that brings a trade-off between these two conflicting objectives, maximizing user QoE and lowering VNF deployment cost. For large networks, the PSP problem is proven to be an NP-hard problem. Thus, we have developed an artificial intelligence-based Hyper-heuristic algorithm for PSP, called HPSP, which can provide a high-performing solution within polynomial time. The HPSP exploits Tabu Search Optimization as a high-level meta-heuristic algorithm that selects one of the three lower-level meta-heuristic algorithms- Golden Eagle Optimizer, Sine Cosine Optimization, and Jellyfish Search Optimization depending on the situation. The results of numerical analysis describe that the HPSP system outperforms the other state-of-the-art works in terms of user QoE, cost, and the ratio of proactive to reactive service placements.

INDEX TERMS Quality of Experience, 5G mobile edge computing, service instances, deployment cost, hyper-heuristic approach.

I. INTRODUCTION

In recent years, smart mobile devices such as cell phones, wearable technology, and smart automobiles have become increasingly prevalent in our everyday lives as a result of the widespread use of cellular mobile networking and the fast growth of 5^{th} Generation (5G) networks. The 5G cellular network is one of the emerging technologies for providing a more consistent user experience, greater data speeds, ultra-low latency, increased reliability, vast network

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

capacity, Quality-of-Service (QoS), etc [1], [2], [3]. The 5G network technology has gained much popularity for providing data-intensive and compute-intensive applications like Virtual Reality (VR), face recognition, and video conferencing by delivering quicker, more consistent lower latency, data rates, and lower cost [4]. According to IMT-2020 specifications, 5G is designed to offer peak data rates of up to 20 Gbps, energy savings of between 80 and 90 percent, and an increase in device connection of up to 100 times that of 4G network [5].

Additionally, by leveraging virtualization, Software Defined Networks (SDN) and Network Function Virtualization (NFV) can significantly enhance the functionality of 5G network architecture [6], [7]. The NFV concept assists the mobile customer in keeping the network operations of the mobile applications in the cloud rather than on any individual hardware device. Executing application codes for real-time delay-sensitive applications in the distant master cloud requires higher latency. Thus, utilizing cloud infrastructure for mobile edge networks in the MEC environment reduces response time for real-time 5G network applications [8]. There are multiple ENs in a MEC environment. A user device automatically establishes connections to the ENs covering that area when it moves from one location to another. Depending on their needs, they can access services from the previous EN where their service replica was running or migrate it to the new EN. This may increase the service delay and, as a result, it decreases the user QoE. Therefore, service instance replication and a quicker migration technique should be employed to deliver real-time services for the user computationally intensive and latency-sensitive tasks to boost user QoE.

There are currently two migration processes available for migrating the service instance: the first is known as proactive migration (PM), whereas the second is known as reactive migration (RM), [9]. In the case of RM, when a user travels from one EN to another EN, then the user's data is migrated to a new service instance at the current EN from the prior EN, and the service instances on the prior one are switched off. Before the migration procedure is finished, users need to use the previous EN's service. On the contrary, service instances are proactively deployed under the PM strategy in nearby ENs by considering the geographical proximity of the users, which in turn drastically reduces the migration time and incurs additional deployment costs. Fig. 1 depicts a typical service migration model in the MEC. User 1 moves from one location to another and requests the application within the coverage of Edge Node 3, and the system launches the application's primary service copy i.e., VNF in Edge Node 3. The copies of the same application service are proactively deployed in the Edge Nodes 4, 2, and 1 to ensure quicker migration when user 1 connects to any of these ENs. However, if the user connects to any of the other ENs that are not in the user's trajectory, then reactive migration is invoked.

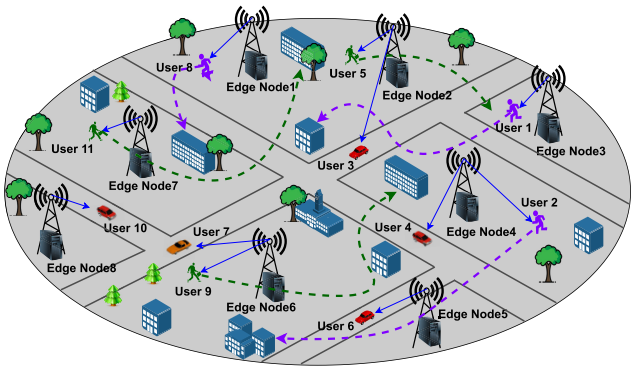


FIGURE 1. Service migration in 5G MEC.

The proactive distribution of service instances substantially enhances the user's QoE. As opposed to that, when reactive migration is raised, response time increases, lowering VNF deployment costs and decreasing user QoE. As a result, selecting the appropriate EN for service instance distribution while maximizing user QoE and lowering VNF deployment costs has become a critical research problem. In the topic of MEC, a considerable number of works have been carried out to deploy the service instances in the MEC environment. Roy et al. [10] have presented the procedure for placing service instances in the cloud data center to improve the QoE of the end users by trading off between the service instances' relocation time and communication delay. They have not, however, taken into account the user's route prediction model or the cost of deployment when placing the service instances. Another group of authors in [11] has developed two optimization strategies, one aims to lessen reactive migration by proactively deploying additional service instances in the ENs. And other one aims to reduce the deployment costs by restricting the placement of service instances in advance. However, they have deployed the service instances without considering users' movement trajectory, which in turn wastes the reserved resources for users. Furthermore, the deployment cost in the former strategy can be infinite, which is not realistic in a real-world situation. In [12], the authors have considered the placement of service instances including the directions of user mobility while simultaneously attempting to strike a trade-off between the QoE and the costs of placement using a meta-heuristic algorithms-based solution. One major weakness of meta-heuristic algorithms is that they may be very effective for a certain scenario but are unable to provide optimal results in constantly changing various scenarios due to getting stuck in the local optima [13]. To overcome this limitation, a hyper-heuristic can be used which selects the best heuristic from multiple heuristics in a constantly changing network environment.

Some real-life challenges related to service instance placement in MEC due to user mobility have yet to be covered in the literature. The subsequent concerns persist unexplored and thus the research questions to be answered in this paper are listed below.

1) In the context of user mobility, how to achieve an appropriate trade-off between the QoE and the service deployment costs?
2) How can we achieve the highest QoE while minimizing service deployment expenses when a user has a limited budget?
3) How can we determine the efficient approach for proactive and reactive service placement decision-making exploiting a hyper-heuristic approach?

In this paper, we have developed a framework for mobility-aware **P**roactive **S**ervice **P**lacement (PSP) problem, with the objective of getting a trade-off between maximizing user QoE and minimizing the deployment cost of the service instances. The aforementioned optimization problem has been formulated using Multi-Objective Linear Programming. The system needs to deploy a large number of service instances to enhance the user QoE, which in turn also increases the system's VNF deployment cost. Therefore, the PSP framework brings a trade-off between these two conflicting objectives. For large networks, the optimization formulation will be an NP-hard problem. Therefore, a **H**yper heuristic based **P**roactive **S**ervice **P**lacement (HPSP) is also devised, which adopts Tabu Search Algorithm as a higher level heuristic and Sine Cosine Optimization algorithm for PSP (SCPSP), Jellyfish Search Optimizer for PSP (JSPSP), and Golden Eagle Optimizer algorithm for PSP (GEPSP) as the lower level heuristic algorithms, which can provide solutions within polynomial time and improve QoE and minimize system's VNF deployment cost for large networks. The following are the main contributions of this paper:

- We develop an optimization framework for the mobility-aware PSP problem as a Multi-Objective Linear programming one that brings a trade-off between maximizing user QoE and minimizing the system's VNF deployment cost under budget and resource constraints.
- Due to the NP-hardness of the developed optimization framework, a hyper-heuristic algorithm, namely HPSP has been developed, which dynamically exploits several selected heuristics to merge their strengths and mitigate their weaknesses in the search space.
- We have done extensive numerical analysis using Python programming version 3.8.0 and the results show that the HPSP system achieves significant performance improvements concerning QoE and VNF deployment costs by 20% and 18%, respectively.

The organization of the remaining sections of this paper is as follows: Section II presents the state-of-the-art works on service replica placement and service migration strategies in MEC. In Section III, we provide a detailed description of our developed system model and assumptions. The developed optimization framework for the PSP method is discussed in Section IV. The Hyper-Heuristic Algorithm for the developed PSP system is presented in Section V, and the performance of the proposed HPSP system is evaluated in Section VI. Finally, in Section VII, we conclude the paper and suggest potential areas for future research.

II. RELATED WORK

In this section, we have investigated the research work concerning service placement and replication strategies that are regarded as emerging research areas in the 5G cloud infrastructure. Several studies have been conducted on the service placement, migration, and replication scheme in the distributed cloud, or MEC environment [14], [15], [16], [17], [18].

To obtain real-time service and satisfy the user QoE considering mobility, a quicker live virtual machine (VM) migration strategy needs to be used. The VM handover approaches in the MEC system that accommodate the dynamic behavior of the cloudlets are discussed by the authors in [19] and [20]. However, these strategies lengthen the overall VM transmission delay, which also lowers the user QoE. The authors in [21] have provided some approaches for VMs to ensure high reliability to minimize the difficulties of service replication. This method necessitates constant error detection, and it is recoverable by shifting to an available instance. Two methods exist for identifying errors; the first approach is record-and-reply; and the second one is, check-pointing. Large amounts of data must be sent over a special link on the replica side and saved in the primary replica in the first strategy. The second technique depends heavily on checkpoint frequencies. To bring an energy and latency trade-off of mobile users, Gu et al. [22] have developed energy-efficient binary and partially offloading strategies that minimize energy consumption while taking into consideration how long the tasks are to be processed. However, they have not considered user mobility and service deployment costs during the offloading of the services.

The Ultra-Reliable Low Latency Communications (URLLC) services that the 5G network provides are focused on providing real-time low-latency access to the applications. Hence, in [23], the authors have proposed the Follow Me Cloud (FMC) architecture, which lowers the migration time by allowing service migrations between the edges. This architecture always links the users to the most advantageous edge. To minimize downtime during migration, the researchers of [15] have proposed two migration methodologies based on the FMC concept. In one scheme, it was assumed that mobile user routes were known, but in the other scheme, unknown directions were assumed. The goal of both approaches was to improve the efficiency of total migration procedures. The authors in [24] have devised a particle swarm optimization (PSO) based technique for improving user QoS and minimizing the server energy consumption due to user mobility by utilizing transmission power control. Even so, the authors neglected the accessible resource competence of the ENs and service deployment costs. Nashaat & zhou et al. [25], [26] have presented decentralized techniques for efficiently placing service instances to reduce the utilization of network assets. Subsequently, in [27] another group of authors has developed an optimization framework to bring a trade-off between service latency and availability to satisfy the

URLLC service requirement. However, they have not taken into account the cost of implementing the services in the MEC.

The service instance migration technique suggested by the authors in [10] has brought a trade-off between service migration time and network communication delay. Users have the option to get service from the present data center (DC), provided that the required service instance is running in that DC. If this is not possible, they migrate the service instances from the preceding DC to the current one or get service from the prior DC. The first scenario increases the migration time, and the latter scenario enhances communication latency. However, the authors have not utilized any route prediction model during the placement of service replicas. It may be possible to dramatically cut down on migration time if service replicas could be placed proactively across the path nodes. Shah et al. [28] have developed two optimization models for service replica placement in the 5G MEC scenario due to user mobility. Therefore, they have used a mobility pattern to predict the user's final destination. Reducing the number of RM (Min-RM) and decreasing the cost of PM (Min-NSR) are two key objectives of their work. Using the Min-RM technique, they sought to restrict RM. As a consequence, they have deployed the service instance proactively as much as possible without considering any service deployment cost. In such a scenario, the cost of replication can be infinite, which is implausible. On the other side, the Min-NSR has aimed to reduce service deployment costs without considering any experience quality.

To decrease service unavailability and improve service quality, the authors of [29] have studied the Markov decision process and a deep Q-network-based service function chaining (SFC) migration strategy by considering user mobility and have proposed a migration timing decision (DQN-MTD) algorithm. However, they have not taken into account the service deployment cost and user service costs for getting services. Another group of authors in [30] have studied a greedy method to reduce the amount of VNF migration using traffic prediction by exploiting the LSTM model. Compared to random and first-fit approaches, a significant improvement can be seen in the results of their study. However, they have not considered the resource cost for training the LSTM model. Moreover, the service deployment cost has been ignored in their work. Subsequently, the authors in [12] have developed an optimization framework namely POPP to bring a trade-off between two conflicting objectives, service deployment cost, and QoE by exploiting the service instances in the user movement trajectory. Due to the NP-hardness of their developed optimization model for large networks, they have proposed a meta-heuristics Binary Particle Swarm Optimization (BPSO) based service instance deployment problem. But depending on different network scenarios different heuristics may perform better which is not considered here. The study conducted in [31] delved into the issue of service migration, especially the process of migrating an active service from the present edge server to a selected target edge server. To solve this issue, the researchers have developed a Deep Recurrent Q-learning-based service migration decision algorithm namely DRQNSM. The major goal of this method is to reduce both user latency and energy usage during the migration process. This included analyzing user data to determine if a specific task should be handled locally, partially inside a single edge node, or transferred to the next edge node. It's important to note that their migration strategy was purely reactive, which enhances the overall migration delay. In their analysis, they solely took into account the aspect of user delay, omitting considerations such as user task completion deadlines, as well as user costs and budget constraints.

Most of the existing works have not focused on mobility-aware proactive service placement method that addresses both optimizing QoE and VNF deployment costs. As a result of focusing primarily on QoE improvement, VNF deployment costs rise. Again, reducing VNF deployment costs does not improve the quality of the user experience significantly. When installing service replicas in ENs, no prior effort has taken into consideration the user's route. Moreover, different heuristics solutions show different performances based on the dynamic network environment. These observations have inspired us to develop a system that strikes a compromise between enhancing user QoE and decreasing deployment expenses by exploiting a hyper-heuristic approach.

III. SYSTEM MODEL AND ASSUMPTIONS

A. SYSTEM OVERVIEW

We consider a 5G MEC environment, which comprises of M ENs, represented by $\mathbb{M} = \{1, 2, 3, \ldots, M\}$, and, each EN is referred to as an index $1 \leq m \leq M$. Let, N indicate the number of mobile users who request a service represented by $\mathbb{N} = \{1, 2, 3, \ldots, N\}$, where, the index of a user is $1 \leq n \leq N$. We additionally assume that there is an edge orchestrator (EO) that regulates the ENs, $\mathbb{M}$, and makes the appropriate placement plan for the service replicas. We presume that one user will request only one service at a given time, and the user must have a task (live video streaming or online live gaming) that requires an excessively low delay. Let, $J = \{j_1, j_2, \ldots, j_n\}$ be used to represent the set of tasks of all users, where, j_n indicates the requested task of user $n \in \mathbb{N}$. There is a deadline Δ_n for each task $j_n \in J$, and each user $n \in \mathbb{N}$ has a service cost budget of B_n for completing the task. Every mobile customer has a computationally demanding and delay-sensitive task that must be processed on an EN. As users request real-time delay-sensitive applications, it requires placing the service replicas proactively on the user's movement route to receive uninterrupted service. The EO uses the user movement trajectory and contextual data from the path prediction model, to be detailed in Section III-B and analyze the current load status of each EN. In the context of stateful service instances, we explore the concept of Flexible and Low-Latency State Transfer, namely FAST [18],

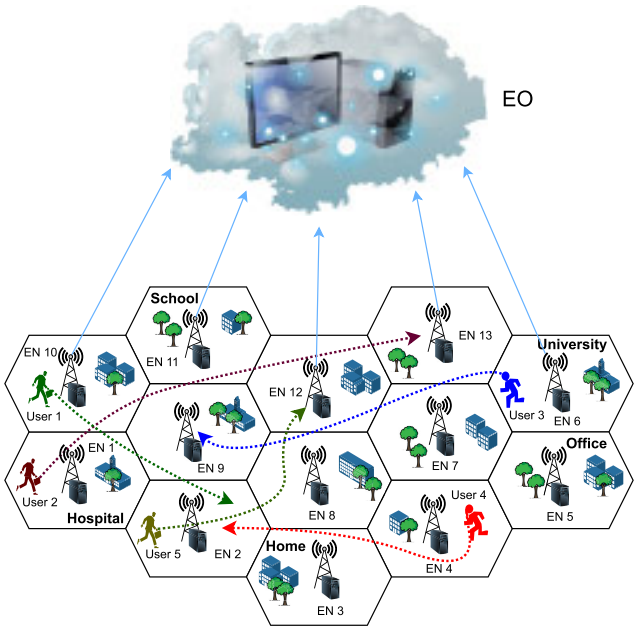


FIGURE 2. A general service migration model in 5G MEC system.

TABLE 1. Notation table.

Notation	Description
$\mathbb{M}, \mathbb{N}$	Set of ENs and users, respectively
j_n	Task of mobile user $n \in \mathbb{N}$
λ_n, O_n	Input and output data sizes of tasks
$H_{m,n}$	Channel gain between EN m and user n
W_m	Channel bandwidth between EN m and user n
Δ_n	Time deadline of task j_n
$r^u_{m,n}, r^d_{m,n}$	Uplink and downlink data transmission rates, respectively, between EN $m \in \mathbb{M}$ & mobile user $n \in \mathbb{N}$
I_n	Number of instructions in task j_n
ζ_m	Maximum VNF holding capacity of EN $m \in \mathbb{M}$
$\xi_{m,n}$	Total delay experienced by a user $n \in \mathbb{N}$ for a task $j_n \in J$ from EN $m \in \mathbb{M}$
$\mu_{m,n}$	Binary variable for predicted path nodes
$X_{m,n}$	Binary variable for service placement
$\mathcal{S}$	Set of initial solutions
X^k	Position vector of solution $k \in \mathcal{S}$
T_n^{deploy}	Deployment period of a task $j_n \in J$
$\mathbb{E}_{m,n}$	Service deployment cost of user $n \in \mathbb{N}$ at EN $m \in \mathbb{M}$
B_n	The service cost budget of user $n \in N$ for a task $j_n \in J$
α	Weight factor, where, $\alpha \in [0, 1]$
$\overline{Q}_{m,n}$	QoE enjoyed by user $n \in \mathbb{N}$ for deploying a service instance at EN $m \in \mathbb{M}$

a programmable state transfer for synchronization state information across multiple instances of the application. The EO governs the direct transfers of application state from the source application instances to the destination. This process leverages switches enabled by SDN and deploys the service instances accordingly.

In Fig. 2, as an illustration, one customer $n \in \mathbb{N}$ may change his/her location after uploading a task $j_n \in J$ to its nearest edge server $m \in \mathbb{M}$ for execution. We assume that our scenario includes multiple tasks and multiple ENs that adhere to a specific migration path and try to reduce the service delay and service deployment cost while maximizing QoE by placing service instances among multiple ENs. Nonetheless, due to limited budget, B_n of user $n \in \mathbb{N}$, and limited VNF holding capacity ζ_m of EN $m \in \mathbb{M}$, the proactive migration will be restricted. Let, X be a D-dimensional binary matrix, where, $D = M \times N$, and each entity $X_{m,n}$ refers to a decision variable. If a user $n \in \mathbb{N}$ has a service instance on EN $m \in \mathbb{M}$, then the value of $X_{m,n}$ is 1, otherwise, it is 0. The main notations that are used in this paper are displayed in Table 1.

B. PATH PREDICTION MODEL

We anticipate the path of each user's (pedestrian/vehicular) course of travel by adopting the destination path prediction model (DPPM), which is introduced in [32], with the assumption that the DPPM has prior information about the user's destination. Several frequently visited locations inside a road network are denoted by their coordinates (i.e., longitude and latitude), also each of such places are identified by a unique ID. The edge orchestrator keeps the user contextual database, and historical movement database, which tracks the user movement and contextual information like user characteristics, specific days, and time of visiting places, respectively. The path prediction algorithm begins from the origin and repeats till it gets to the destination. In every repetition, the previous road section's endpoint becomes the current position, and then the potential nearby sections are chosen depending on the predetermined threshold. The best neighboring segment is then chosen using a combination of the penalty function, the transition probability based on historical data, and the connecting point with the previous road segment. When the selected road portion appears to be a dead end, we have here set the penalty function value to zero. In the end, it will return a list of road sections orderly denoted by R_n corresponding to user $n \in \mathbb{N}$.

When a user goes from one site to another along the route, he or she needs to obtain services by the most appropriate node based on its geographical location and received signal power. Due to the curves in the roadways, it may be divided between a certain number of straight road portions, which may be served by one or several ENs. Depending on the coordinates and connection intensities, the EO links a single EN or an ordered collection of ENs necessary to serve that segment and appends it to the set $\mathbb{M}$ of ENs to assist user $n \in \mathbb{N}$ on the predicted trajectory $r \in R_n$.

IV. DESIGN OF PSP SYSTEM

In this section, we have established the detailed design of PSP, which is executed on the EO. Here, we have calculated the QoE and VNF deployment cost in terms of delay and service deployment periods, respectively. Then, we have presented the optimal formulation of the PSP problem as a MOLP one, which brings a trade-off between the aforementioned two conflicting objectives.

A. CALCULATION OF QOE

Without loss of generality, we adopt a widely used parameter model to describe task a $j_n \in J$. Uplink data transmission rate between mobile user n and source EN $m \in \mathbb{M}$ can be expressed as:

$$r^u_{m,n} = W_m log_2\left(\frac{1 + p_n H_{m,n}}{\theta_m}\right), \tag{1}$$

where, W_m and $H_{m,n}$ refer to wireless channel bandwidth and channel gain between EN m and mobile user n; p_n denotes the transmission power of user n and θ_m is the white noise power. The upload time between user n and EN m can be calculated as follows:

$$T^u_{m,n} = \frac{\lambda_n}{r^u_{m,n}}, \tag{2}$$

where, λ_n denotes the input data size of task $j_n \in J$. We can calculate the downlink data transmission rate $r^d_{m,n}$ between a user $n \in \mathbb{N}$ and an EN $m \in \mathbb{M}$ similar to uplink data transmission date using Eq. (1). The required time to transfer the result to user n from EN m can be calculated as follows,

$$T^d_{m,n} = \frac{O_n}{r^d_{m,n}}. \tag{3}$$

where O_n is the size of output data. In the computation model, there is a processing time to execute the task at EN m. The execution time depends on the edge server's processing speed i.e., millions of instructions per second (MIPS) value of a VM of the edge server m and required instructions I_n to execute the task $j_n \in J$. Processing time is calculated to execute task j_n in node m is,

$$T^p_{m,n} = \frac{I_n}{MIPS_m}. \tag{4}$$

When a user is connected to an edge server $m \in \mathbb{M}$, where the expected VNF is not deployed proactively, then we need to do reactive migration to migrate the VNF from the previously running edge server. The required migration time to migrate the task $j_n \in J$ with data size λ_n can be calculated as follows,

$$T^{mig}_n = \frac{\lambda_n}{\beta}, \tag{5}$$

where, β indicates the bandwidth between two edge servers. However, in the case of proactive VNF migration, the VNF will be migrated to the EN before the user, and the input data size of the task does not affect the delay, *i.e.*, $\lambda_n = 0$; therefore, $T^{mig}_n = 0$. The total delay is equal to the summation of data uploading time, downloading time, processing time, and migration time, denoted by $\xi_{m,n}$, which can be obtained as follows,

$$\xi_{m,n} = T^u_{m,n} + T^{mig}_n + T^d_{m,n} + T^p_{m,n}. \tag{6}$$

Now, we can measure the QoE of user $n \in \mathbb{N}$ for deploying the service instance at EN $m \in \mathbb{M}$ as follows,

$$Q_{m,n} = 1 - \frac{\xi_{m,n}}{\Delta_n}, \tag{7}$$

where, Δ_n is the application delay deadline of task j_n. Note that, there is an inverse relationship between the total service delay and QoE. Higher service delay indicates a lower QoE and vice versa. By exploiting the user path prediction model, we can calculate the user's QoE as follows,

$$\overline{Q}_{m,n} = \{\psi \times \mu_{m,n} + (1 - \psi) \times (1 - \mu_{m,n})\} \times Q_{m,n} \times X_{m,n}. \tag{8}$$

Here, ψ is the accuracy level of the path prediction model, and the ranges of ψ and $\overline{Q}_n$ are between 0 and 1. If a user $n \in \mathbb{N}$ does not take service from the node $m \in \mathbb{M}$, then $X_{m,n} = 0$, which indicates that the QoE for that user will be 0. Also note that the QoE depends on the value of $\mu_{m,n}$ and the accuracy of the path prediction model ψ. If the node m is the predicted path node for user $n \in \mathbb{N}$, then $\mu_{m,n} = 1$, which implies that the resultant QoE is $\overline{Q}_{m,n} = \psi \times Q_{m,n}$. Otherwise, the QoE is determined $\overline{Q}_{m,n} = (1 - \psi) \times Q_{m,n}$. The lower accuracy value of the path prediction model deviates the users to get services in time which in turn decreases the QoE of the users.

B. CALCULATION OF DEPLOYMENT COST

There is a deployment cost to deploy a service instance in the edge server. Let, $e_{m,n}$ denote the per unit deployment cost to deploy a service instance $n \in \mathbb{N}$ proactively at edge server $m \in \mathbb{M}$. This cost may vary due to the cost of the computational, storage resources, and energy consumption cost for deploying the service instances. For a single user to deploy a service in one edge server, the cost is:

$$\mathbb{E}_{m,n} = e_{m,n} \times T^{deploy}_n \times X_{m,n}, \tag{9}$$

where, T^{deploy}_n denotes the deployment period of the service instance for user $n \in \mathbb{N}$. The normalized deployment cost, $\overline{\mathbb{E}}_{m,n}$, can be determined as follows,

$$\overline{\mathbb{E}}_{m,n} = \frac{\mathbb{E}_{m,n}}{B_n}, \tag{10}$$

where, budget B_n denotes the amount of money a user $n \in \mathbb{N}$ can make expense out of getting computation services and $\overline{\mathbb{E}}_{m,n} \in [0, 1]$.

C. OBJECTIVE FUNCTION

Our main objective in this work is to maximize the user QoE while minimizing the system deployment cost, which is formulated as a MOLP problem that is expressed as follows,

$$Maximize: \quad Z = \sum_{n\in\mathbb{N}} \sum_{m\in\mathbb{M}} \{\alpha \times \overline{Q}_{m,n} - (1 - \alpha) \times \overline{\mathbb{E}}_{m,n}\} \tag{11}$$

$$Subject\ to: \quad X_{m,n} \in \{0, 1\} \tag{12}$$

$$\sum_{m\in\mathbb{M}} X_{m,n} \geq 1, \quad \forall n \in \mathbb{N} \tag{13}$$

$$Q_{m,n} \geq Q^{min}_n, \quad \forall n \in \mathbb{N} \tag{14}$$

$$\sum_{m\in\mathbb{M}}\{\mathbb{E}_{m,n}\times X_{m,n}\}\leq B_n,\quad \forall n\in\mathbb{N} \tag{15}$$

$$\sum_{n\in\mathbb{N}} j_n\leq \zeta_m\quad \forall m\in\mathbb{M} \tag{16}$$

Here, Eq. (11) is the objective function of our work, which brings a trade-off between maximizing the user QoE and minimizing the total deployment cost. In this context, there are some constraints which need to be satisfied. $X_{m,n}$ in constraint (12) is a binary variable, whose value is 1 if user n has a task proactively deployed at EN $m \in \mathbb{M}$; 0 otherwise. Constraint (13) refers to the service availability constraint, which ensures that there is at least one service instance available for each user. Constraint (14) is the user's QoE constraint, and it ensures that each user should enjoy a minimum level of QoE. The user's budget constraint in Eq. (15) confirms that the total service instance deployment cost of user $n \in \mathbb{N}$ must be lower than or equal to the user's budget for it. Constraint (16) is the capacity constraint, which shows that the total number of allocated service instances for each EN $m \in \mathbb{M}$ must be less than or equal to the EN's VNF holding capacity ζ_m.

Theorem 1: The PSP problem formulated in Eq. (11) is an NP-Hard problem.

Proof: The proposed MOLP formulation can be reduced to the Multiple Knapsack Problem (MKP) [33], which is also an NP-complete problem. The following are the features of the MKP.

- A set of item $\mathcal{I} = \{1, \ldots, i\}$
- A set of knapsack $K = \{1, \ldots, k\}$
- A profit function $P_{i,k}$ that indicates profit of assigning an item $i \in \mathcal{I}$ to a knapsack $k \in K$
- A weight function δ_i that denotes the weight of an item $i \in \mathcal{I}$.
- An availability function W_k indicates the available capacity of a knapsack $k \in K$. This problem tries to maximize the overall profit by placing the items in the available knapsacks.

$$Maximize: \quad \sum_{i\in\mathcal{I}}\sum_{k\in K} P_{i,k}\times X_{i,k} \tag{17}$$

$$Subject\ to: \quad \sum_{i\in\mathcal{I}}\delta_i X_{i,k}\leq W_k,\quad \forall k\in K \tag{18}$$

$$\sum_{k\in K} X_{i,k}\leq 1,\quad \forall i\in\mathcal{I} \tag{19}$$

$$X_{i,k}\in\{0,1\},\quad \forall i\in\mathcal{I},\quad \forall k\in K \tag{20}$$

The optimal PSP problem can be reduced to MKP by leveraging some constraints. Let, $Y_{m,n} = \alpha\times Q_{m,n}\times\psi\times\mu_{m,n}$. By considering the QoE for deploying the service instances at the path node, our optimization formulation can be reduced to as follows,

$$Maximize: \quad \sum_{n\in\mathbb{N}}\sum_{m\in\mathbb{M}} Y_m X_{m,n} \tag{21}$$

$$Subjectto: \quad \sum_{n\in\mathbb{N}} j_n\leq\zeta_m,\quad \forall m\in\mathbb{M} \tag{22}$$

$$\sum_{m\in\mathbb{M}} X_{m,n}\leq 1,\quad \forall n\in\mathbb{N} \tag{23}$$

$$X_{m,n}\in\{0,1\},\quad \forall n\in\mathbb{N},\quad \forall m\in\mathbb{N} \tag{24}$$

Since the MOLP formulation of the PSP problem can be reduced to the MKP, it is plausible to infer that the optimal proactive service placement problem is at least as hard as MKP. Therefore, the above optimal formulation cannot be solved in polynomial time for larger values of m and n. ■

V. DESIGN OF HYPER-HEURISTIC PSP SYSTEM

Due to the NP-hardness of the PSP system developed in the previous section, we have developed an AI-enabled Hyper-heuristic strategy for the Proactive Service Placement problem. Hyper-heuristic refers to a multi-level computing technique in which a higher-level heuristic algorithm governs the search space of the underlying heuristic methods rather than the solutions of an underlying problem. Meta-heuristics might be a good starting point for their simplicity and can be effective if the environment is somewhat predictable and careful parameter tuning is feasible. Hyper-heuristics hold promise for achieving better performance [34] and self-adaptation in dynamic MEC environments. The main objective in the hyper-heuristic algorithms is to choose the best heuristic from multiple heuristics pools. Because each heuristic or meta-heuristic method has some strengths and limitations while working on a problem area, the Hyper-heuristic approach selects or combines the simple heuristics by combining their strength and compensating for their weaknesses. In the next sections, we will discuss elaborately population representation, and higher and lower level heuristics of our devised HPSP algorithm.

A. POPULATION REPRESENTATION IN PROBLEM DOMAIN

The HPSP algorithm starts with a list of potential solutions, represented by $\mathcal{S}$. The position of a candidate solution $k \in \mathcal{S}$ in the search space is denoted by a D dimensional binary vector, where $D = M \times N$ and $X^k = (x_1^k, x_2^k, \ldots, x_d^k, \ldots, x_D^k)$. Each entry $x_d^k \in X^k$ corresponds to a decision variable $X_{m,n}$, where $n \in \mathbb{N}$, $m \in \mathbb{M}$ and $1 \leq d \leq D$, such that,

$$x_d^k = X_{m,n}, \tag{25}$$

where, $d = \{(m-1)\times N + n\}$, and also, $1 \leq m \leq M$ and $1 \leq n \leq N$.

B. REPRESENTATION OF FITNESS FUNCTION

To find the best solution from each of the lower-level heuristics, we need to calculate the fitness of every solution X^k by using a fitness function $F(.)$. A higher fitness value indicates a better solution. The fitness function $F(.)$ is almost similar to our objective function of Eq. (11). However, we have slightly modified our objective function to assign a positive fitness value for the feasible solution and a negative

value for the infeasible solution as follows,

$$F(X^k) = \begin{cases} \sum_{x_d \in X^k} \{\alpha \times \overline{Q}_d - (1-\alpha) \times \overline{\mathbb{E}}_d + 1\} \\ \qquad \text{if Eqs. (12) to (16) are satisfied} \\ -1 \qquad \text{Otherwise} \end{cases} \tag{26}$$

Here, $\overline{Q}_d = \overline{Q}_{m,n}$, and $\overline{\mathbb{E}}_d = \overline{\mathbb{E}}_{m,n}$, where $d = \{(m-1) \times N + n\}$, $1 \le n \le N$, and $1 \le m \le M$. $\overline{Q}_d$ and $\overline{\mathbb{E}}_d$ indicate the normalized value of QoE and cost at the d^{th} dimension. Here, the fitness function $F(.)$ is almost identical to the objective function of Eq. (11). However, The fitness function $F(.)$ in Eq. (26) is slightly modified to assign the positive value for the feasible solution and the negative value to the infeasible solutions.

The best solution of lower heuristic is denoted as $\mathcal{L}$ which is updated using the following equation,

$$\mathcal{L}(t''+1) = \begin{cases} X^k(t''), & if\ F(X^k) > F(\mathcal{L}) \\ \mathcal{L}(t''), & if\ F(X^k) \le F(\mathcal{L}). \end{cases} \tag{27}$$

Here, Eq. (27) updates the local best solution $\mathcal{L}$ with the current solution X^k, if its fitness value is greater than $\mathcal{L}$. Otherwise, we have restored the previous local best solution as the current local best solution.

C. DESIGN OF HPSP ALGORITHM

The structural design of HPSP is shown in Fig. 3. The developed HPSP algorithm utilizes the concept of the Tabu Search Algorithm as a high-level meta-heuristic algorithm that selects one of the Lower Level Heuristic (LLH) algorithms from the lower-level heuristic pool to find polynomial time solutions based on feedback quality. Local search algorithms can get stuck in local optimal solutions. However, Tabu Search [35] improves the performance by preventing revisiting the same solutions that have been used previously. The algorithm does not consider a potential solution again if it has already been considered within a predetermined short-term window. Instead, it is marked as "tabu". At the beginning, the HPSP algorithm chooses an LLH at random from Sine Cosine optimization, Jellyfish Search optimization, and Golden Eagle optimization. The chosen LLH is adapted to the solution space of PSP. After that, the population will be updated by the selected LLH, and HLH will get the information from the updated feedback. Based on the level of feedback quality from the LLHs, HLH changes the selection of LLH and updates the Tabu list to adapt to the solution space of PSP again.

1) INITIAL SOLUTION GENERATION

The optimization process starts with a collection of random solutions in population-based heuristics. A fitness function evaluates this random collection periodically, and a set of operators enhances it. As a result, HPSP starts by calculating an initial value for a collection of solutions $\mathcal{S}$. Each solution

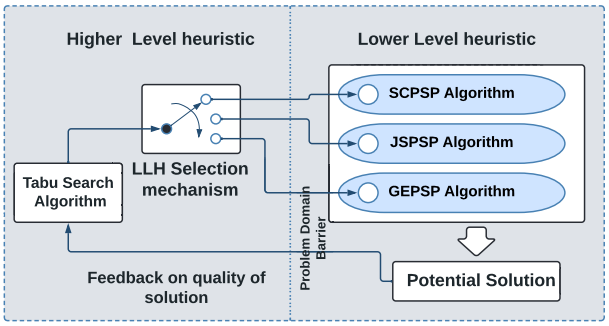


FIGURE 3. Structural design of HPSP algorithm.

Algorithm 1 HPSP Algorithm

Input: T_{max} : Maximum Iterations, $\mathcal{W}$: Number of solution, LLH: {$SCPSP$, $JSPSP$, $GEPSP$}
Output: Service allocation matrix, X

for all $k \in \mathcal{W}$ **do**
  Generate initial solution set, $\mathcal{S} \leftarrow X^k$ Using Eq.(28)
end for
$H_i \leftarrow$ randomly selected from LLH
while $t'' \le T_{max}$ **do**
  $\mathcal{L} \leftarrow$ Update solution using the fitness function (26) of selected heuristic H_i
  if $F(\mathcal{G}) < F(\mathcal{L})$ **then**
    $\mathcal{G} \leftarrow \mathcal{L}$
    Keep H_i
  else
    Tabu List $\leftarrow H_i$
    if Tabu List is Full **then**
      Randomly select a Heuristic $H_{i'}$, where $i' \neq i$ from Tabu list
    else
      Randomly select from LLH
    end if
  end if
end while
for each $d \in \mathcal{G}$ **do**
  $m \leftarrow \lceil \frac{d}{N} \rceil$
  $n \leftarrow \{d - (m-1) \times N\}$
  $X_{m,n} \leftarrow \mathcal{G}_d$
end for
return X

generates in a first fit approach where we assign the value by considering the budget constraint (15), capacity constraint (16), and the path prediction model which can be expressed as,

$$x_b^k = 1 \quad \text{if Eq. (15), Eq. (16) are satisfied.} \tag{28}$$

2) ACCEPTANCE AND SELECTION MECHANISM

At the beginning, the HPSP algorithm randomly chooses a LLH as H_i. The result of iteration t'' is stored as the global best solution and compares with the result of iteration $t''+1$. If the feedback improves then we don't insert H_i into the tabu

list and run the selected LLH again but otherwise, we insert H_i into the tabu list. In each iteration, HPSP chooses an arbitrary one of the LLHs (excluding the tabu list) based on their previous efficiencies in the search space, as shown by the solution feedback. In line 4, the result of H_i will be saved as the local best $\mathcal{L}$. In lines 5 to 15, we will check if the local best $\mathcal{L}$ improves the global best $\mathcal{G}$ or not. If it improves then we will keep the H_i. Otherwise, H_i will be marked as tabu and we will check if the tabu list is full (Line 10-14). If it is full then it will select another Heuristic $H_{i'}$ from LLH where $i' \neq i$.

D. LOWER LEVEL HEURISTIC ALGORITHMS

In our developed Hyper-heuristic solution for the PSP problem, three meta-heuristic algorithms namely, Sine Cosine Optimization algorithm [36] for Proactive Service Placement (SCPSP), Jellyfish Search Optimizer [37] for Proactive Service Placement (JSPSP) and Golden Eagle Optimizer algorithm [38] for Proactive Service Placement (GEPSP) are used as the lower level heuristics to update the solution space of the PSP problem. The descriptions of these algorithms are given next, as per our fitness function and solution set $\mathcal{S}$.

1) SCPSP ALGORITHM

Sine Cosine Optimization is a simple population-based optimization algorithm, which is worked by initially creating several random candidate solutions and then moving them towards or outwards the best solution using a mathematical model based on sine and cosine functions [36]. This algorithm is divided into two phases: exploration and exploitation. The exploration happens when traveling in between the range $[-r, r]$ in sine and cosine, and random solutions are blended with other solutions thus growing a high rate of randomness in order to find out the potential regions of the search space. The SCPSP not only finds (exploration) random solutions but also exploits them to get the optimal solution. The solution space in SCPSP can be updated at each iteration as follows,

$$p_d^k(t''+1) = \begin{cases} x_d^k(t'') + r_1 \times sin(r_2) \times \Delta x_d^k(t''), & r_4 < 0.5 \\ x_d^k(t'') + r_1 \times cos(r_2) \times \Delta x_d^k(t''), & r_4 \geq 0.5 \end{cases} \tag{29}$$

where,

$$\Delta x_d^k(t'') = |r_3 \times \mathcal{L}_d(t'') - x_d^k(t'')| \tag{30}$$

$$r_1 = a_1 - t'' \times \frac{a_1}{t_{max}}. \tag{31}$$

Here, $a_1 = 2$, which is a sinusoidal nature constant, and r_1 is the sinusoidal nature controlling parameter ranges in $[0, 2]$, which determines whether the next position's movement occurs within or outside the region between the solution and the destination. r_2 is a random number between $[0, 2\pi]$, which specifies the magnitude of motion toward or away from the target. r_3 assigns a random weight to the destination, which can stochastically accentuate (if $r_3 > 1$) or de-emphasis (if $r_3 < 1$) the effect of desalination to determine the destination and r_4 defines the alternation between the sine and cosine functions in an equal manner using Eq. (30).

Algorithm 2 SCPSP Algorithm

Input: t_{max} : Maximum Iterations, $\mathcal{W}$: Number of solution, $\mathcal{S}$: Solution set.
Output: Optimal solution, $\mathcal{L}$

```
Compute F(X^k) using Eq. (26)
while t'' ≤ t_max do
  for all solution X^k ∈ S do
    Update L using Eq. (27)
    for each dimension d ∈ X^k do
      Update the value x_d^k using Eq. (33)
      Update the random parameters r1, r2, r3, r4
    end for
  end for
  Update t'' = t'' + 1
end while
return L
```

The value of $p_d^k(t''+1)$ calculated from Eq. (29) provides continuous value. In PSP, the values must be converted into binary variables. The sigmoid function converts this continuous value into a probability value using Eq. (32). Then we update the position using Eq. (33).

$$\sigma(p_d^k(t''+1)) = \frac{1}{1+e^{-p_d^k(t''+1)}} \tag{32}$$

$$x_d^k(t''+1) = \begin{cases} 1 & \text{if } \sigma(p_d^k(t''+1)) \geq rand(0,1) \\ 0 & \text{Otherwise.} \end{cases} \tag{33}$$

Here, if the probability value is greater than the random value, then x_d^k is 1, indicating that the service instance is deployed at the d^{th} dimension for the k^{th} population. The whole process is summarized in Algorithm 2.

2) JSPSP ALGORITHM

Jelly Fish Search (JS) Optimization algorithm is derived from the social movements of jellyfish in the water, such as how they follow the ocean current and migrate within the swarm [37]. There are two types of movement that the jellyfish follow. They either float with the stream which works as the exploration phase, or swim inside the swarm which resembles the exploitation phase. The movement of Jellyfish depends on the food availability in the ocean, and the amount of food in a place is calculated via an objective function. Jellyfish have a time control system, which works as a regulator to swap between these two movements. The time control system includes $\Omega_f(t'')$ which is a time control function and a constant Ω_{const}. The value of the time control function can be calculated as follows,

$$\Omega_f(t'') = \left|(1 - \frac{t''}{t_{max}}) \times (2 \times rand(0,1) - 1)\right|. \tag{34}$$

The precise value of Ω_{const} is unknown, and value of $\Omega_f(t'')$ fluctuates within 0 to 1. For that reason, the value of Ω_{const} is set as 0.5, which is the average of 0 and 1. When the value of the time control function $\Omega_f(t'')$ in Eq. (34) surpasses Ω_{const}, then the jellyfish moves with the ocean current.

Algorithm 3 JSPSP Algorithm

Input: t_{max} : Maximum Iterations, $\mathcal{W}$: Number of solution, $\mathcal{S}$: Solution set.
Output: Optimal solution, $\mathcal{L}$

```
Find the jellyfish with best location L, in Population set S using Eq. (26)
while t'' ≤ t_max do
  for each k ∈ W do
    Calculate the time control Ω_f(t'') using Eq. (34)
    for each dimension d ∈ X^k do
      if Ω_f(t'') ≥ Ω_const then
        Determine New Position by Eq. (35)
      else
        if rand(0, 1) > (1 − Ω_f(t'')) then
          Determine New Position by Eq. (37)
        else
          Determine New Position by Eq. (38)
        end if
      end if
      Update value of x_d^k Using Eq. (33)
    end for
    Update L using Eq. (27)
  end for
  t'' = t'' + 1
end while
return L
```

The orientation of ocean current is determined by the mean position vector of all jellyfish to the finest jellyfish position. The following equation is used to calculate the new position of the d-th component of a jellyfish toward the ocean current.

$$p_d^k(t''+1) = x_d^k(t'') + rand(0,1) \times (\mathcal{L}_d - rand(0,1) \times 3 \times x_d^{mean}) \quad (35)$$

Here, $\mathcal{L}_d$ is the d^{th} element of the jellyfish with the best position found so far. x_d^{mean} refers to the average position of d^{th} element from all jellyfish which can be calculated as,

$$x_d^{mean} = \frac{\sum_{k=1}^{\mathcal{W}} x_d^k}{\mathcal{W}}. \quad (36)$$

where, $\mathcal{W}$ is total number of jellyfish.

When the value of time control function $\Omega_f(t'')$ is smaller than Ω_{const}, then the jellyfish travels within the swarm. In this case, the swarm jellyfish shows two types of motion which are known as passive and active motions. Passive movement refers to the movement of jellyfish around their own places. Due to this movement, the new position for every jellyfish can be determined as follows,

$$p_d^k(t''+1) = x_d^k(t'') + rand(0,1) \times 0.1 \times (B_{upper} - B_{lower}), \quad (37)$$

where, B_{upper} and B_{lower} are upper bound and lower bound of search space, respectively. To imitate the active motion, a jellyfish k except for the one of interest is arbitrarily chosen, and the direction of flow starts at jellyfish l to the chosen jellyfish k. When the amount of food accessible to the chosen jellyfish k exceeds the amount available to the jellyfish of interest l, the latter redirects its attention to the former. If the amount of food available to a particular jellyfish k is less than the amount available to a different jellyfish l, the latter moves directly away from the former. Therefore, in a swarm, each jellyfish swims in the best direction to obtain food. This can be mathematically represented as,

$$p_d^k(t''+1) = \begin{cases} x_d^k(t'') + rand(0,1) \times (x_d^l(t'') - x_d^k(t'')) \\ \qquad \text{if } F(X^l(t'')) \geq F(X^k(t'')) \\ x_d^k(t'') + rand(0,1) \times (x_d^k(t'') - x_d^l(t'')) \\ \qquad \text{if } F(X^l(t'')) < F(X^k(t'')). \end{cases} \quad (38)$$

This movement is seen as the exploitation of the local search space. Here, the time control system is also used to swap between passive motion and active motion. If the value of $(1-\Omega_f(t''))$ is greater than a random number between 0 to 1 then the passive motion is selected; otherwise, the active motion will be shown. At the end of the JSPSP algorithm, it will return an optimal solution $\mathcal{L}$, which is the deployment vector of the service instances. Algorithm 3 summarizes the above processes.

3) GEPSP ALGORITHM

Another LLH algorithm for our developed PSP problem is the Golden Eagle Optimizer (GEPSP) Algorithm, which is also a swarm-based meta-heuristic method for global optimization [38]. The notion of this algorithm was inspired by the golden eagle's turning speed during various stages of its circular course while hunting. During the early phases of hunting, golden eagles are more likely to cruise than attack, which is known as the exploration phase. However, in the end, they tend to be more likely to attack than cruise, which is termed as exploitation phase. Each golden eagle keeps the best prey location it has visited in its memory. The eagle must maintain a delicate balance between exploitation and exploration propensities to find the best prey location. The Eagles can randomly choose between the best prey spots visited by other golden eagles and their own best prey location. In each cycle, the eagle picks a targeted meal at random from the flock's memory. The attack and cruise vectors for each eagle are then computed relative to the prey they choose. The choosing procedure is completely random and unaffected by any factors such as distance from the

Algorithm 4 GEPSP Algorithm

Input: t_{max} : Maximum Iterations, $\mathcal{W}$: Number of solution, $\mathcal{S}$: Solution set, $[P_c^0, P_c^{t_{max}}] = [1, 0.5]$: Propensity to cruise, $[P_a^0, P_a^{t_{max}}] = [0.5, 2]$: Propensity to attack, M : population memory
Output: Optimal solution, $\mathcal{L}$

1: $M \leftarrow$ Evaluate $F(X^k)$ for each $X^k \in \mathcal{S}$ using Eq. (26)
2: **while** t'' in t_{max} **do**
3: Update P_a and P_c using Eq. (44) and Eq. (45)
4: **for each** $k \in \mathcal{W}$ **do**
5: $\overrightarrow{X^*} \leftarrow$ Randomly select a solution from $\mathcal{S}$
6: Compute $\overrightarrow{A_k}$ using Eq. (39)
7: **if** $\|\overrightarrow{A_k}\| \neq 0$ **then**
8: Compute $\overrightarrow{C_k}$ and ΔX_k using Eq. (40) and Eq. (42)
9: **for each** dimension $d \in X^k$ **do**
10: Update Position using Eq. (46)
11: Update the value x_d^k using Eq. (33)
12: **end for**
13: Compute $F(X^k)$ using Eq. (26)
14: **if** $F(X^k) > M[k]$ **then**
15: $M[k] \leftarrow F(X^k)$
16: $\mathcal{L} \leftarrow X^k$
17: **end if**
18: **end if**
19: **end for**
20: $t'' = t'' + 1$
21: **end while**
22: **return** $\mathcal{L}$

prey's location. When allocating prey to an eagle, a one-to-one mapping is performed, meaning that an eagle is always allocated just one prey. We can model the golden eagles' attack with a vector $\overrightarrow{A^k}$ that begins at the eagles' current location and finishes at the preys' location. The calculation of the attack vector is as follows,

$$\overrightarrow{A^k} = \overrightarrow{X^f} - \overrightarrow{X^k}, \tag{39}$$

where, $\overrightarrow{A^k} = [a_1^k, a_2^k, a_3^k, \ldots, a_d^k, \ldots, a_D^k]$ and $\overrightarrow{X^k} = [x_1^k, x_2^k, x_3^k, \ldots, x_d^k, \ldots, x_D^k]$ denote the D dimensional attack vector and current location vector of k^{th} eagle. $\overrightarrow{X^f}$ is the best frequented destination or prey.

The exploration phase or the cruise vector $\overrightarrow{C^k}$, is perpendicular to both the attack vector and the spherical path of the eagle, which can be calculated using Eq. (40) and Eq. (41) as follows,

$$\overrightarrow{C^k} = \left[c_1^k, c_2^k, \ldots, c_d^k, c_{d'}^k \ldots, c_D^k\right], \tag{40}$$

$$c_d = \frac{\sum_{x_d \in X^k} h_d x_d - \sum_{d \neq d'} a_{d'}}{a_d} \tag{41}$$

where, $H = [h_1, h_2, h_3, \ldots, h_D]$ represents the normal vector, c_d and a_d represent the d^{th} element of the cruise vector $\overrightarrow{C^k}$ and attack vector $\overrightarrow{A^k}$, respectively.

The eagles' displacements or step vector involves attack and cruise vectors, which can be calculated as follows,

$$\Delta X_k = \overrightarrow{r_1} p_a \frac{\overrightarrow{A_k}}{\|\overrightarrow{A_k}\|} + \overrightarrow{r_2} p_c \frac{\overrightarrow{C_k}}{\|\overrightarrow{C_k}\|} \tag{42}$$

$$\|\overrightarrow{A_k}\| = \sqrt{\sum_{j=1}^{d} a_j^2}, \quad \|\overrightarrow{C_k}\| = \sqrt{\sum_{j=1}^{d} c_j^2}. \tag{43}$$

Here, r_1 and r_2 are random vectors whose elements fall inside the range [0,1]. p_a and p_c represent the coefficients of attack and cruise vectors, respectively, that can be calculated using Eq. (44) and Eq. (45). The value of P_a regulates exploration and exploitation. The algorithm starts with low p_a and high p_c. As the iterations proceed, p_a is gradually increased while p_c is gradually decreased.

$$P_a = P_a^0 + \frac{t''}{t_{max}} \|P_a^{t_{max}} - P_a^0\| \tag{44}$$

$$P_c = P_c^0 - \frac{t''}{t_{max}} \|P_c^{t_{max}} - P_c^0\| \tag{45}$$

Here, t'' and t_{max} denote current iteration number and maximum iterations, respectively. P_a^0 and $P_a^{t_{max}}$ denote the initial and final values of propensity to attack, whereas, P_c^0 and $P_c^{t_{max}}$ are initial and final cruise propensities, respectively. In this paper, we have set $[P_a^0, P_a^{t_{max}}] = [0.5, 2]$ and $[P_c^0, P_c^{t_{max}}] = [1, 0.5]$. This implies that P_a starts at 0.5 and linearly reaches to 2. On the contrary, P_c begins at 1 and is gradually dropped to 0.5.

The previous duty cycle is applied to the new duty cycle to calculate the new location by Eq. (46) that corresponds to the updated duty cycle as follows,

$$p_d^k(t'' + 1) = x_d^k(t'') + \Delta X_d^k(t''), \tag{46}$$

where, $p_d^k(t''+1)$ is the position of the next iteration, $x_d^k(t'')$ is the position of the current iteration, and $\Delta X_d^k(t'')$ is the step vector. If the golden eagle's new location is better than the location it has in its memory, the k^{th} eagle's memory changes to reflect the new location. Otherwise, the eagle restores the same location in its memory and deploys the service instances accordingly. The above processes are summarized in Algorithm 4.

E. COMPLEXITY CALCULATION OF HPSP

We have calculated and evaluated the complexity of algorithms in this section. In the HPSP algorithm, at lines 1-3, $|\mathcal{W}|$ solutions are generated and each solution is a $|D|$ dimensional binary vector. Therefore, the complexity for lines 1-3 is $O(|\mathcal{W}| \times |D|)$. Lines 5-18 iterate T_{max} times using a while loop. Inside the while loop, HPSP selects an LLH from the Lower Level Heuristics pool. The complexity of SCPSP, JSPSP, and GEPSP can be written as $O((|\mathcal{W}| \times |D|) + (t_{max} \times |\mathcal{W}| \times |D|))$. Therefore, the complexity of lines 5-18 sums up as $O(T_{max} \times ((|\mathcal{W}| \times |D|) + (t_{max} \times |\mathcal{W}| \times |D|)))$. From lines 19-23, a loop iterates $|D|$ times. Hence, the complexity

TABLE 2. Simulation parameters.

Parameter	Value
Simulation area	$2000 \times 2500\ m^2$
Number of users	100 - 600
Number of ENs	5 - 25
Proactive migration probability threshold	0.5
Deployment period of a VNF	600 sec
User budget	50 - 200 Unit
VNF holding capacity of each EN	250
Weight factor (α)	0.6
Population Size	30
Maximum iteration for LLH (t_{max})	200
Maximum iteration for HLH (T_{max})	100
Sinusoidal nature constant (a_1)	2
Motion coefficient(γ)	0.1

of HPSP is $O(T_{max} \times ((|\mathcal{W}| \times |D|) + (t_{max} \times \mathcal{W}| \times |D|)) + |D|) \approx O(T_{max} \times t_{max} \times |\mathcal{W}| \times |D|)$.

VI. PERFORMANCE EVALUATION

In this part, we compared the performance of the proposed HPSP system with the state-of-art works: DRQNSM [31], POPP [12] and Min-NSR [39]. We have done extensive numerical analysis to assess the efficiency of the developed HPSP system.

A. ENVIRONMENTAL SETUP

We envision an MEC scenario comprised of multiple ENs, where each EN is distributed in a city space of $2000 \times 2500m^2$ following uniform random distribution. When small cells are deployed, each EN is modeled as an LTE eNB with a network range of about 250 meters. Those ENs possess a small server with a storage capability of 4 to 8 terabytes, also the primary RAM is within 2 - 16 GB. Nearly 25 to 30 VMs are housed in each network infrastructure, which are available in a variety of sizes and capacities. Each VM comprises 512 MB to 1024 MB of RAM and a clock speed of 3.30 GHz to 4.15 GHz. These VMs incorporate replicas of the customer's services and are responsible for their operation. To make an accurate prediction of the consumer pathway, we use the path prediction method developed in [32] and in conjunction with the Geo Life project [40], retrieved from Microsoft Research Asia's database. In this collection, there are about 17,621 different trajectories, and the combined range of all of them is close to 1.2 million kilometers. However, to assess HPSP's performance, we only investigate the $2000 \times 2500m^2$ road network in the aforementioned database. We assume, that just one service is linked to a customer at any given time, the customer budget is established with uniform random selection within regions of 50 to 200 Units, and the user arrival pattern is determined by poison distributions. Each plot is constructed via 50 virtual model processes that use randomly generated seed values.

B. PERFORMANCE METRICS

The following mentioned metrics are used to determine the efficacy of our designed HPSP system:

- **Quality-of-Experience (QoE)** refers to the inverse of the average delay of all users for getting the services by deploying the service instances on the ENs that corresponds to the user's trajectory as well as the ENs that have the highest likelihood, which is calculated using Eq. (8). Higher service delay indicates a lower QoE and vice versa. The higher value of QoE denotes that the users can get faster service.
- **Normalized VNF Deployment Cost** is calculated as the average cost for deploying the service instances of the users proactively. The normalized VNF deployment cost is computed by dividing the VNF deployment cost by the user's budget, with a range of [0, 1]. The bigger the value, the lesser the performance of the system.
- **Proactive to Reactive Migration Ratio** is stated as the comparison between the probability of proactive migration and reactive migration of all users. A greater value indicates a higher proactive migration probability than the reactive.
- **Average Migration Time:** It indicates the average time required for migrating the user service instances to the ENs. If the service is deployed proactively on the preferable EN, then there won't be any migration delay. However, if the service is deployed reactively, then the user will experience some migration delay.

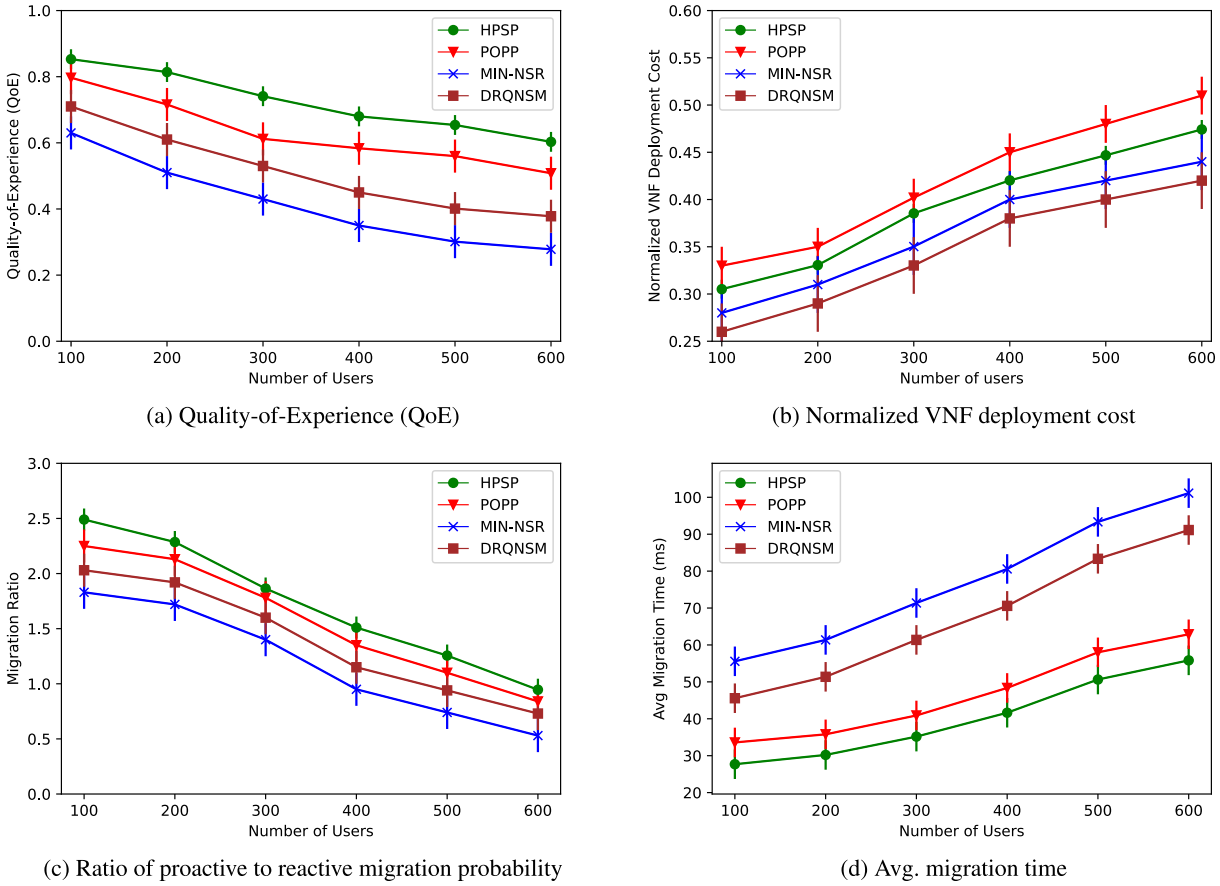
C. RESULT ANALYSIS

The performance of our suggested HPSP solution is addressed here which is assessed by adjusting the number of users, number of ENs of the system, and average instruction size of the tasks.

1) IMPACTS OF VARYING NUMBER OF USERS

Throughout the simulation, we have changed the total number of users while keeping the number of ENs and the average instruction size of the tasks constant at 12 and $140k$, consequently.

Fig 4(a) shows that as the quantity of users grows, the average quality of experience decreases. This is due to the fact that with the increasing number of users, many service instances can not be deployed proactively to the user's trajectory nodes for all users, which in turn reduces the user QoE. The DRQNSM is only concerned with the optimization of delay by processing tasks partially in the local environment and deploying the rest of the service instances in the ENs to reduce energy consumption caused by reactive service migration without considering any user QoE. Also, the Min-NSR system's focus is only on reducing deployment costs without considering any user QoE. On the other hand, POPP and our developed HPSP jointly optimize the user QoE and service deployment cost and exploit the user movement trajectory system for deploying service instances proactively, which in turn reduces the service latency and enhances QoE. In HPSP, it finds the best result from the three LLHs, whereas in POPP, the outcome is determined by a single



(a) Quality-of-Experience (QoE)

(b) Normalized VNF deployment cost

(c) Ratio of proactive to reactive migration probability

(d) Avg. migration time

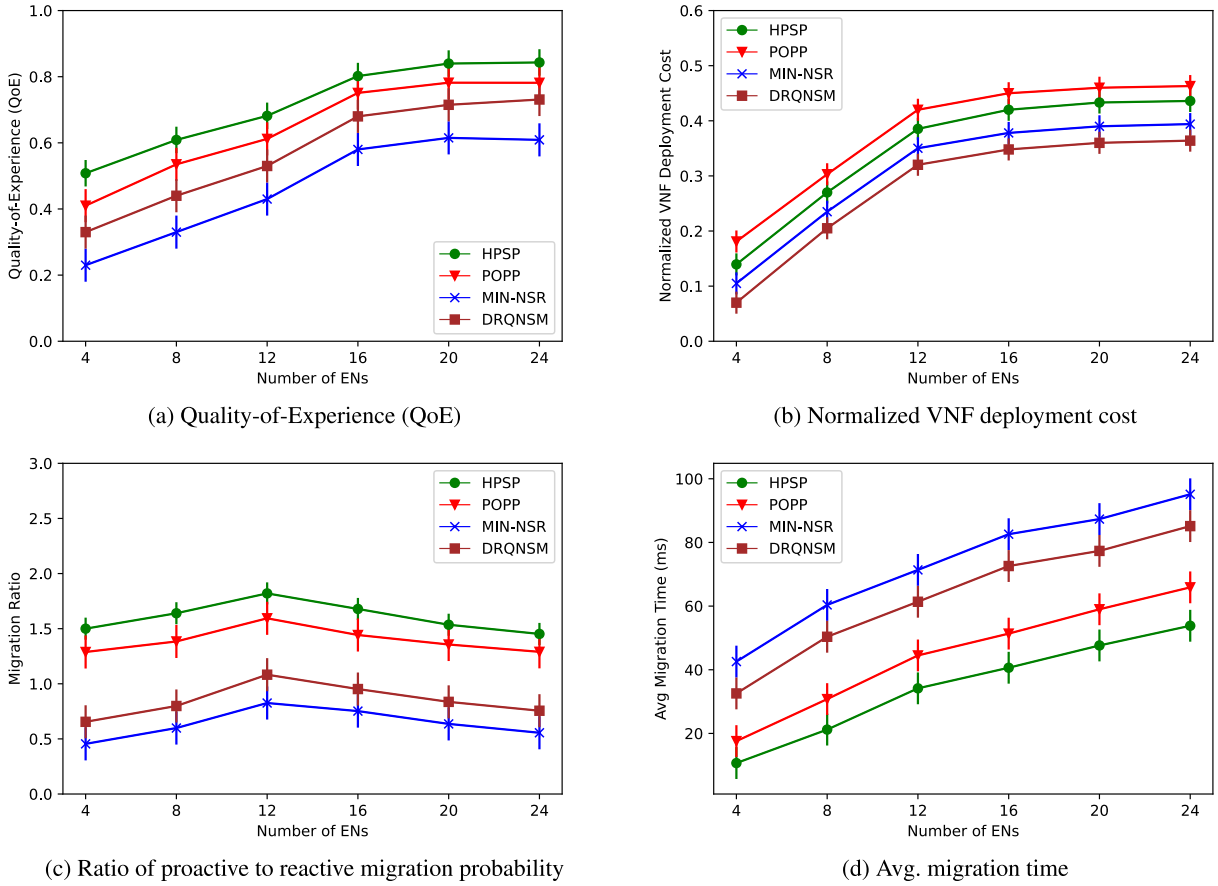
FIGURE 4. **Impacts of varying number of users.**

meta-heuristic, which might not always lead to the best outcome. Therefore, HPSP performs better in terms of user QoE.

Fig.4(b) depicts that the cost of deployment rises as the number of users grows. This is as a result of the system having to place additional service instances with the growing quantity of users. Deployment cost is almost similar in HPSP and POPP which is higher than DRQNSM and Min-NSR since they are primarily concerned with reducing delay, energy consumption, and costs while ignoring QoE. Deployment cost in HPSP is relatively lower than the POPP system because the HPSP finds the best result from the three LLHs, where POPP provides the result using only one meta heuristic. That's why, the HPSP outperforms the state-of-the-art works.

The change in the ratio of the probability of proactive to reactive deployment can be seen in Fig.4(c). As the number of users increases, the frequency of proactive deployment decreases and reactive deployment increases. The fundamental reason for this is that each EN's limited capabilities limit each user's proactive placement of service instances on their chosen ENs. The POPP and HPSP systems both aim to distribute service instances as far as feasible on the ENs within the budget while reducing the chance of reactive migration. However, the performance of HPSP is a little bit better than POPP because HPSP finds the best LLH from the three LLHs of the heuristic pool, which provides the best fitness value. The migration ratio is lower in DRQNSM, because, it only focuses on reducing reactive migration to minimize service delay without placing any service instances proactively. Similarly, Min-NSR operates poorly since it is focused on reducing the possibility of reactive migration.

As the user's quantity grows, the average migration time also increases which is depicted in Fig.4(d). The limit of ENs capacity causes an increase in reactive migration with the increasing number of users as a result the average migration also increases. Here, the DRQNSM and Min-NSR performed worst because they only focused on reducing the energy consumption and the deployment cost by increasing the reactive migration which is the result of higher migration time. On the other hand, the HPSP and POPP both exhibit decreased migration times because they reduce reactive migration as much as possible by leveraging the user movement trajectory to increase the QoE of users. Although both HPSP and POPP perform near to each other, HPSP exhibits an improved result due to adopting a hyper-heuristic strategy.



(a) Quality-of-Experience (QoE)

(b) Normalized VNF deployment cost

(c) Ratio of proactive to reactive migration probability

(d) Avg. migration time

FIGURE 5. **Impact of varying number of edge servers.**

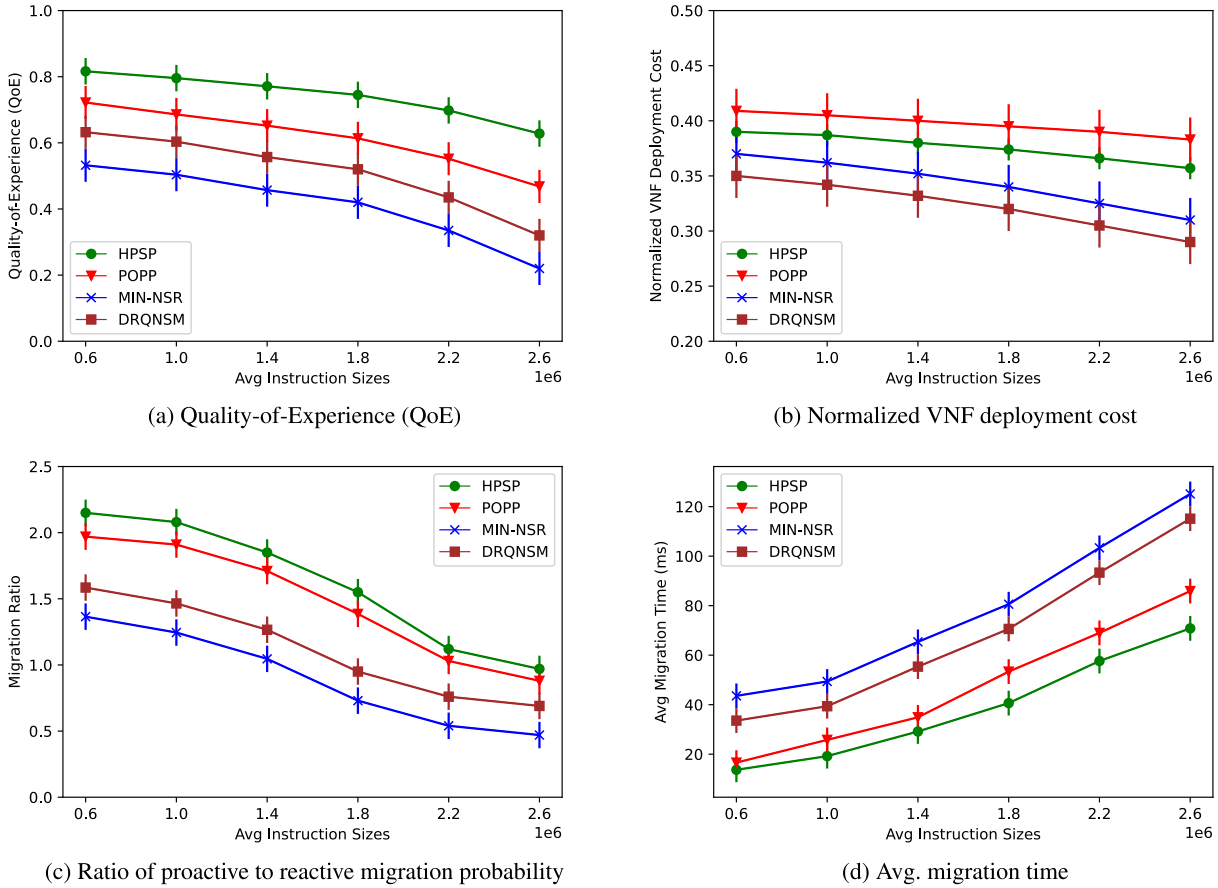
2) IMPACTS OF VARYING NUMBER OF EDGE SERVERS

During this test, we altered the number of ENs in the entire environment by retaining the number of users, and average instruction size of the tasks at 300 and 140k, consequently.

Fig.5(a) indicates that raising the ENs from 4 to 12 enhances QoE, but, increasing the ENs from 12 to 24 does not significantly increase QoE; instead, it approaches a saturation point. Raising ENs from 12 to 24, QoE is not improved. The reason is that the system can deploy the expected number of service copies for users. For that reason, QoE is not elevated in this range. Comparing QoE among DRQNSM, Min-NSR, POPP, and HPSP, the QoE of DRQNSM and Min-NSR are lower because they solely seek to reduce service delay, and deploy service instances reactively. Moreover, the performance of Min-NSR is lower because it solely seeks to reduce the deployment costs without taking into consideration user QoE. In contrast, the POPP and HPSP deploy service instances proactively by taking into account the user trajectory ENs and deploying the service instances on those ENs where there is a higher likelihood that a user will be connected, which improves user QoE. However, HPSP enhances the results of POPP by utilizing the hyper-heuristic Strategy. HPSP selects the LLH among the three meta-heuristics from the heuristic pool which gives the higher QoE. As a result, HPSP outperforms POPP in terms of QoE.

The deployment cost rises as the number of ENs increases, as seen in Fig.5(b). This is because expanding the number of ENs necessitates expanding the deployment of service replicas to ENs. From 4 to 12 ENs the deployment cost significantly rises because user budget and edge capability do not impose any restrictions on the provision of service copies. However, from 12 to 24 ENs, the deployment cost does not change significantly and also maintains almost a fixed range because the systems do not need to deploy additional service replicas to improve user QoE. Comparing deployment costs among the studied systems, the DRQNSM and Min-NSR outperform the developed HPSP solution since they only aim to reduce deployment costs without considering QoE by deploying service instances proactively. The HPSP and POPP systems deploy the service replicas on the predicted path nodes under the constraints of the user budget. However, due to the utilization of a hyper-heuristic strategy in the developed HPSP system, the HPSP system performs better than POPP.

In Fig.5(c), we can observe the change in the ratio of the probability of proactive to reactive deployment due to a change in the number of ENs. This graph demonstrates that by increasing the number of ENs from 4 to 12, the migration



(a) Quality-of-Experience (QoE)

(b) Normalized VNF deployment cost

(c) Ratio of proactive to reactive migration probability

(d) Avg. migration time

FIGURE 6. **Impact of varying instruction sizes of tasks.**

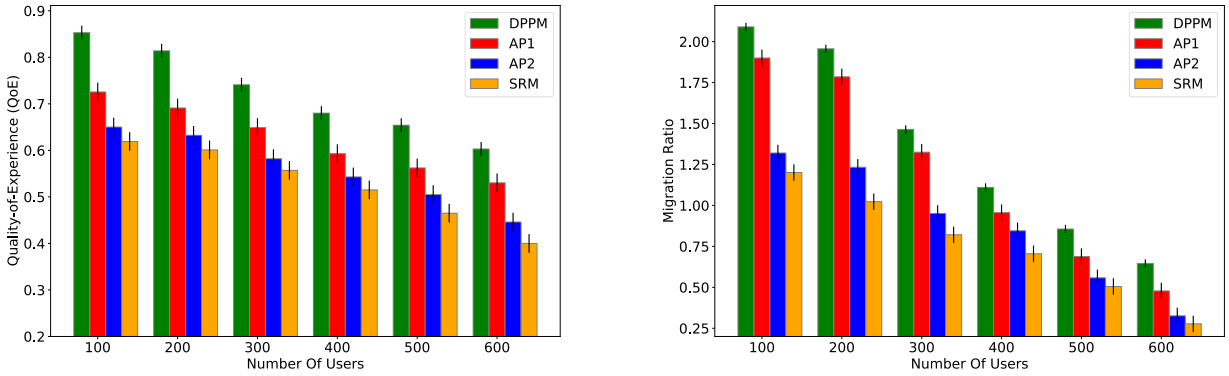
ratio is enhanced. However, this ratio is decremented by raising the ENs from 12 to 24. Because of budget limits, some service replicas cannot be deployed proactively in all desirable ENs. Therefore, the possibility of reactive migration is increased than proactive migration in those scenarios. Here, the DRQNSM and Min-NSR perform badly since they simply evaluate lowering migration costs without addressing diminishing reactive migration likelihood. Although the POPP and HPSP give a close result, however, the HPSP performs slightly better than the POPP. This is because we have selected that lower level heuristic among the LLH which gives the best fitness value.

Form Fig.5(d), it can be seen that with the increase of network size, migration time also increases. The reason is, with the increasing number of edge servers, the number of reactive deployments is increased, which in turn enhances migration time. The system reaches saturation for 300 users when there are 12 ENs in the system, at which point very few reactive migrations are required. We do not need to deploy additional service replicas after increasing the ENs from 12 to 24; therefore, the migration time has not increased as much as before. The DRQNSM and Min-NSR have not deployed the service instances proactively and reduced the reactive migration probability by concentrating on minimizing delay and the deployment cost which is the reason for their large amount of migration time. The POPP and HPSP systems are able to shorten the migration time by deploying service instances proactively by considering the user's trajectory. However, HPSP reduces the migration time further by utilizing the Hyper-heuristic strategy which selects the best heuristic from LLHs depending on the scenario. Therefore, the HPSP outperforms the state-of-the-art works.

3) IMPACTS OF VARYING SIZES OF TASKS

In this experiment, we have altered the instruction size of tasks in the entire system by retaining the number of users, and the number of ENs at 300 and 12, respectively. Also with the increasing sizes of tasks, we have increased the deadline and unit cost for the tasks and budget of users.

From Fig. 6(a), we can observe that QoE experienced by the users is reduced with the increase in instruction size of the tasks. The reason is that, when the instruction size of the tasks is increasing, the deadline of the task is also increasing, but, due to the limitation of computational capabilities of the ENs, services instances of all tasks can not be deployed to the preferred ENs, which in turns degrades QoE. Here, the DRQNSM and Min-NSR system performs the worst because it is solely concerned with lowering delay and deployment costs while neglecting user QoE and have not considered the



(a) Quality-of-Experience (QoE)

(b) Ratio of proactive to reactive migration probability

FIGURE 7. **Impact of various mobility models.**

user path prediction model to deploy the service instances proactively. On the other side, the POPP and HPSP both exploit the user movement trajectory and jointly optimize user QoE and service deployment costs while deploying the service instances. The HPSP exhibits improvement over POPP in terms of QoE due to exploiting the Hyper-Heuristic strategy for allocating the service instances.

The deployment cost decreases as the instruction size of tasks increases, which is shown in Fig.6(b). Here, both the unit cost for tasks and the budget of the user for the tasks are also increased with the increasing size of the tasks. However, due to the ENs' limited processing power and the increasing instruction size of tasks, service instances of some tasks cannot be deployed to the preferred trajectory nodes. Because of this, the cost of deployment decreases as task size grows. The DRQNSM performs best in this case as it is only concerned with the optimization of delay by processing tasks partially in the local environment and deploying the rest of the service instances in the ENs to reduce energy consumption. Similarly, the Min-NSR shows better results than the developed HPSP system in this case as it is focused on reducing cost by reducing proactive deployment. On the contrary, the POPP and HPSP have higher deployment costs than Min-NSR and the results of the POPP and HPSP are nearly identical.

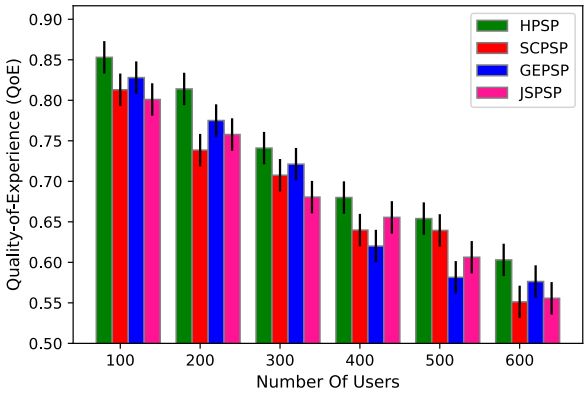
Fig. 6(c) indicates the ratio of the probability of proactive to reactive service placement decreases while increasing the instruction size of tasks. Due to the computational capacity limitation of the ENs, with the increasing number of task sizes the proactive service placement decreases, and the probability of reactive service placement increases. The HPSP and POPP both try to proactively deploy the service instances by considering user trajectory. Therefore, the ratio of proactive to reactive placement in POPP and HPSP systems is almost similar, which is better than DRQNSM and Min-NSR. The result of Min-NSR is the worst because it only focuses on reducing the deployment cost without considering the minimization of reactive deployment. The result of DRQNSM is almost similar to Min-NSR because it only focuses on reducing the delay and deployment cost without concentrating on proactive service deployment.

With the increasing instruction size of the task, the migration time also increases which can be observed from Fig.6(d). For the task with 60k to 140k instruction size, the increments in migration time are small. But for the tasks with 140k to 260k instruction size, the increments in migration time are huge. The inability of tasks with high instruction sizes to be deployed proactively in all required ENs due to the restricted computational capabilities of the ENs increases the likelihood of reactive migration and lengthens the migration time for tasks with large instructions. Migration time in the DRQNSM and Min-NSR is larger than HPSP and POPP because they only consider reactive migration of service instances to minimize deployment costs. Even though the HPSP and POPP show almost the same migration time, HPSP can further decrease migration time than POPP due to exploiting the hyper-heuristic approach, which selects the best LLH from the heuristic pool. Therefore, HPSP outperforms in terms of average migration time.

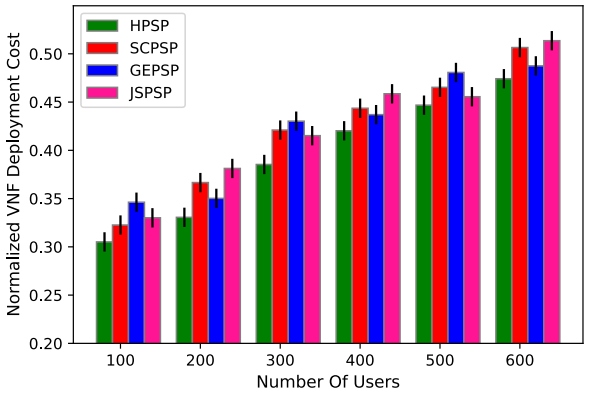
4) IMPACTS OF VARYING THE MOBILITY MODEL

In this experiment, we have changed the mobility model to analyze the changes in the QoE and ratio of proactive to reactive migration probability by varying the total number of users while keeping the number of ENs and the average instruction size of the tasks constant at 12 and 140k. We have applied the mobility prediction model provided in [32], [41], [42], and [43] referred to as Smooth Random Mobility (SRM) Model, AP1, AP2, and DPPM on the HPSP system to evaluate the effectiveness.

In Fig. 7(a), as the number of users increases, the QoE is decreased. Analyzing this graph, we can notice that from all of the above methods, the DPPM performs better. This may be explained by the fact that DPPM retains both the users' past movement traces and contextual information about the users. Moreover, DPPM also considers the time of the day and the day's type (weekday, weekend, occasional, etc.) while



(a) Quality-of-Experience (QoE)



(b) Normalized VNF deployment cost

FIGURE 8. Impact of various heuristics.

filtering the data. In addition to these, DPPM is aware of user destinations, unlike AP1 and AP2. When there are no historical mobility traces of the users, DPPM accuracy is roughly 38% while AP1 and AP2 accuracy is 0% [32]. This is because without having historical mobility traces, these methods cannot make any predictions accurately. Except for these, to reach the destination, DPPM makes proficient use of direction. On the other hand, the SRM model selects the next position based on the random processes of speed and direction control of the current position, which can not make any prediction of the destination route. As a consequence, by employing the DPPM mobility model, the HPSP system near-optimally deploys service replicas, hence improving user QoE. For a similar reason, the proactive to reactive migration ratio is very much higher in our DPPM-based HPSP method shown in Fig 7(b).

5) IMPACTS OF VARYING THE HEURISTIC

In this experiment, we have analyzed the performance of the developed system by using different meta-heuristic algorithms to analyze the changes in the QoE and deployment cost. For that reason, we have varied the total number of users while keeping the number of ENs and the average instruction size of the tasks constant at 12 and 140k. We have applied individual heuristic HPSP, SCPSP, JSPSP, and GEPSP to evaluate the effectiveness.

In Fig. 8(a), as the number of users increases, the QoE is decreased and in Fig 8(b) the deployment cost is increased. The graph analysis reveals that HPSP outperforms all of the aforementioned heuristics in terms of performance. This can be attributed to the fact that HPSP leverages the best efficiency and incorporates multiple levels of heuristics, allowing for optimal heuristic selection within the LLH framework. The difficulties of learning in a constantly changing MEC environment and the constrained capabilities of a single heuristic when compared to a multi-level heuristic in search of optimal solutions might be responsible for the inferior performance of LLH (SCPSP, GEPSP, and JSPSP) compared to HPSP.

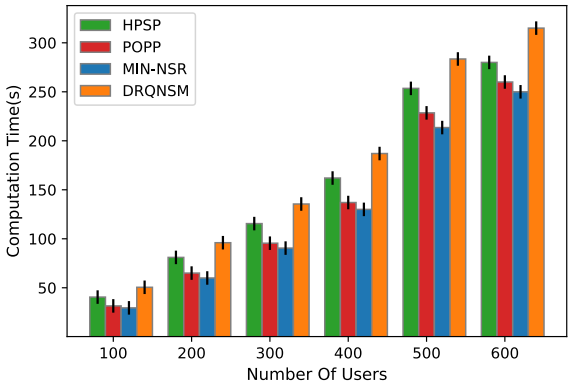


FIGURE 9. Impact of computation time.

6) IMPACTS OF COMPUTATION TIME

In this experiment, we have analyzed the changes in the computation time by varying the total number of users while keeping the number of ENs and the average instruction size of the tasks constant at 12 and 140k.

In Fig. 9, as the quantity of users grows, the computation time of the system increases. This is because, with the increasing number of users, the deployment of additional service instances is needed to handle the load, which in turn increases the program execution time for finding near-optimal solutions. The Min-NSR system takes minimum computation time because it uses a greedy-based algorithm and can get stuck in suboptimal solutions. Here, the POPP system uses a metaheuristic and on the contrary, the developed HPSP system uses a hyper-heuristic algorithm (expanded version of metaheuristic). Meta-heuristics algorithms strive to find good solutions while also exploring the search space for potentially better options. The computation time of the hyper-heuristic-based HPSP solution is a little bit higher than the POPP system due to the automated selection or generation of suitable meta-heuristics, and adaptive Learning for the decision-making process. The DRQNSM system takes the maximum computation time because it uses the DRQN algorithm, which requires a training phase where they

learn from a large amount of data to build a model for making decisions. This training process can be computationally expensive and time-consuming. Therefore, the computational time of DRQNSM is highest among all systems.

VII. CONCLUSION

This work developed a framework for proactive deployment of service replicas in the 5G mobile edge computing environment. Service instances were deployed on route ENs by predicting the users' tracks, which improves user QoE. However, the cost of service deployment was also increased to achieve a better user QoE. As a result, our work developed an optimal deployment approach that traded off between enhancing QoE and decreasing deployment cost. For the NP-hardness of the developed optimal formulation of the service deployment problem for a large network, a hyper-heuristic strategy-based HPSP solution was devised, which maximized user QoE and minimized the deployment cost. The numerical results carried out in Python 3.8.0 revealed that the HPSP system outperformed the DRQNSM, Min-NSR, and POPP systems by up to 40%, 50% and 20%, respectively, in terms of user QoE. Furthermore, it reduced deployment costs by 18% over POPP. Future research has a lot of promise when it comes to examining how to increase VNF deployment effectiveness in a situation where user arrival rates are continually changing.

DECLARATION OF INTERESTS

The authors state that they do not have any known competing financial interests or personal ties that may seem to have influenced the work disclosed in this study.

REFERENCES

[1] S. Painuly, S. Sharma, and P. Matta, "Future trends and challenges in next generation smart application of 5G-IoT," in *Proc. 5th Int. Conf. Comput. Methodologies Commun. (ICCMC)*, Apr. 2021, pp. 354–357.

[2] L. Nadeem, M. A. Azam, Y. Amin, M. A. Al-Ghamdi, K. K. Chai, M. F. N. Khan, and M. A. Khan, "Integration of D2D, network slicing, and MEC in 5G cellular networks: Survey and challenges," *IEEE Access*, vol. 9, pp. 37590–37612, 2021.

[3] P. Ranaweera, A. Jurcut, and M. Liyanage, "MEC-enabled 5G use cases: A survey on security vulnerabilities and countermeasures," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–37, Dec. 2022.

[4] M. Shahjalal, N. Farhana, P. Roy, M. A. Razzaque, K. Kaur, and M. M. Hassan, "A binary gray wolf optimization algorithm for deployment of virtual network functions in 5G hybrid cloud," *Comput. Commun.*, vol. 193, pp. 63–74, Sep. 2022.

[5] G. Kang, H. Liu, and K. Li, "Analysis on the new progress of spectrum planning of IMT-2020(5G)," *J. Phys., Conf. Ser.*, vol. 1437, no. 1, Jan. 2020, Art. no. 012016.

[6] A. A. Barakabitze and R. Walshe, "SDN and NFV for QoE-driven multimedia services delivery: The road towards 6G and beyond networks," *Comput. Netw.*, vol. 214, Sep. 2022, Art. no. 109133.

[7] A. Abdulghaffar, A. Mahmoud, M. Abu-Amara, and T. Sheltami, "Modeling and evaluation of software defined networking based 5G core network architecture," *IEEE Access*, vol. 9, pp. 10179–10198, 2021.

[8] H.-W. Kao and E. H. Wu, "QoE sustainability on 5G and beyond 5G networks," *IEEE Wireless Commun.*, vol. 30, no. 1, pp. 118–125, Feb. 2023.

[9] S. Velrajan and V. C. Sharmila, "QoS-aware service migration in multi-access edge compute using closed-loop adaptive particle swarm optimization algorithm," *J. Netw. Syst. Manage.*, vol. 31, no. 1, p. 17, Jan. 2023.

[10] P. Roy, A. Tahsin, S. Sarker, T. Adhikary, M. A. Razzaque, and M. M. Hassan, "User mobility and quality-of-experience aware placement of virtual network functions in 5G," *Comput. Commun.*, vol. 150, pp. 367–377, Jan. 2020.

[11] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 196–210, Jan. 2022.

[12] P. Roy, S. Sarker, M. A. Razzaque, M. M. Hassan, S. A. AlQahtani, G. Aloi, and G. Fortino, "AI-enabled mobile multimedia service instance placement scheme in mobile edge computing," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107573.

[13] M. Blocho, "Heuristics, metaheuristics, and hyperheuristics for rich vehicle routing problems," in *Smart Delivery Systems* (Intelligent Data-Centric Systems), J. Nalepa, Ed. Amsterdam, The Netherlands: Elsevier, 2020, pp. 101–156.

[14] Y. Ma, M. Dai, S. Shao, Y. Xia, F. Li, Y. Shen, J. Li, Y. Li, and H. Peng, "A performance and reliability-guaranteed predictive approach to service migration path selection in mobile computing," *IEEE Internet Things J.*, vol. 10, no. 20, pp. 17977–17987, Oct. 2023.

[15] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, and H. Flinck, "Fast service migration in 5G trends and scenarios," *IEEE Netw.*, vol. 34, no. 2, pp. 92–98, Mar. 2020.

[16] M.-L. Chiang, H.-C. Hsieh, T.-Y. Chang, T.-L. Lin, and H.-W. Chen, "An adaptive replica configuration mechanism based on predictive file popularity and queue balance in mobile edge computing environment," *Soft Comput.*, vol. 27, no. 1, pp. 107–129, Jan. 2023.

[17] S. D. A. Shah, M. A. Gregory, and S. Li, "Cloud-native network slicing using software defined networking based multi-access edge computing: A survey," *IEEE Access*, vol. 9, pp. 10903–10924, 2021.

[18] T. V. Doan, G. T. Nguyen, M. Reisslein, and F. H. P. Fitzek, "FAST: Flexible and low-latency state transfer in mobile edge computing," *IEEE Access*, vol. 9, pp. 115315–115334, 2021.

[19] Y. Mansouri and M. A. Babar, "A review of edge computing: Features and resource virtualization," *J. Parallel Distrib. Comput.*, vol. 150, pp. 155–183, Apr. 2021.

[20] V. M. Varier, D. K. Rajamani, N. Goldfarb, F. Tavakkolmoghaddam, A. Munawar, and G. S. Fischer, "Collaborative suturing: A reinforcement learning approach to automate hand-off task in suturing for surgical robots," in *Proc. 29th IEEE Int. Conf. Robot Human Interact. Commun. (RO-MAN)*, Aug. 2020, pp. 1380–1386.

[21] S. A. Mohamed, S. Sorour, and H. S. Hassanein, "Group-delay aware task offloading with service replication for scalable mobile edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[22] X. Gu, C. Ji, and G. Zhang, "Energy-optimal latency-constrained application offloading in mobile-edge computing," *Sensors*, vol. 20, no. 11, p. 3064, May 2020.

[23] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "On enabling 5G automotive systems using follow me edge-cloud concept," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5302–5316, Jun. 2018.

[24] Q. Cao, Q. Wu, B. Liu, S. Zhang, and Y. Zhang, "An optimization method for mobile edge service migration in cyberphysical power system," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–12, Feb. 2021.

[25] H. Nashaat, E. Ahmed, and R. Rizk, "IoT application placement algorithm based on multi-dimensional QoE prioritization model in fog computing environment," *IEEE Access*, vol. 8, pp. 111253–111264, 2020.

[26] A. Zhou, S. Wang, S. Wan, and L. Qi, "LMM: Latency-aware micro-service mashup in mobile edge computing environment," *Neural Comput. Appl.*, vol. 32, no. 19, pp. 15411–15425, Oct. 2020.

[27] X. Wang, Z. Xu, and S. He, "Low latency-oriented reliable slicing for URLLC services over TDM-PON based mobile edge computing enabled cloud radio access network," in *Proc. 19th Int. Conf. Opt. Commun. Netw. (ICOCN)*, Aug. 2021, pp. 1–3.

[28] S. D. A. Shah, M. A. Gregory, S. Li, and R. D. R. Fontes, "SDN enhanced multi-access edge computing (MEC) for E2E mobility and QoS management," *IEEE Access*, vol. 8, pp. 77459–77469, 2020.

[29] H. Hu, W. Zhang, L. Xu, and P. Qi, "A mobility-aware service function chain migration strategy based on deep reinforcement learning," *J. Netw. Syst. Manage.*, vol. 31, no. 1, p. 21, Jan. 2023.

[30] F. Carpio, W. Bziuk, and A. Jukan, "Scaling migrations and replications of virtual network functions based on network traffic forecasting," *Comput. Netw.*, vol. 203, Feb. 2022, Art. no. 108582.

[31] W. Chen, Y. Chen, and J. Liu, "Service migration for mobile edge computing based on partially observable Markov decision processes," *Comput. Electr. Eng.*, vol. 106, Mar. 2023, Art. no. 108552.
[32] A. Nadembega, A. Hafid, and T. Taleb, "A destination and mobility path prediction scheme for mobile networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 6, pp. 2577–2590, Jun. 2015.
[33] M. Hung and J. Fisk, "An algorithm for 0-1 multiple knapsack problems," *Nav. Res. Logistics Quart.*, vol. 25, no. 3, pp. 571–579, 1978.
[34] G. L. Pappa, G. Ochoa, M. R. Hyde, A. A. Freitas, J. Woodward, and J. Swan, "Contrasting meta-learning and hyper-heuristic research: The role of evolutionary algorithms," *Genetic Program. Evolvable Mach.*, vol. 15, no. 1, pp. 3–35, Mar. 2014.
[35] R. Chelouah and P. Siarry, "Tabu search applied to global optimization," *Eur. J. Oper. Res.*, vol. 123, no. 2, pp. 256–270, Jun. 2000.
[36] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.
[37] J.-S. Chou and D.-N. Truong, "A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean," *Appl. Math. Comput.*, vol. 389, Jan. 2021, Art. no. 125535.
[38] A. Mohammadi-Balani, M. Dehghan Nayeri, A. Azar, and M. Taghizadeh-Yazdi, "Golden eagle optimizer: A nature-inspired metaheuristic algorithm," *Comput. Ind. Eng.*, vol. 152, Feb. 2021, Art. no. 107050.
[39] I. Farris, T. Taleb, M. Bagaa, and H. Flick, "Optimizing service replication for mobile delay-sensitive applications in 5G edge network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.
[40] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma, "Understanding transportation modes based on GPS data for web applications," *ACM Trans. Web*, vol. 4, no. 1, pp. 1–36, Jan. 2010.
[41] C. Bettstetter, "Smooth is better than sharp: A random mobility model for simulation of wireless networks," in *Proc. 4th ACM Int. Workshop Modelling, Anal. Simulation Wireless Mobile Syst.*, Jul. 2001, pp. 19–27.
[42] H. Abu-Ghazaleh and A. S. Alfa, "Application of mobility prediction in wireless networks using Markov renewal theory," *IEEE Trans. Veh. Technol.*, vol. 59, no. 2, pp. 788–802, Feb. 2010.
[43] T. Anagnostopoulos, C. Anagnostopoulos, and S. Hadjiefthymiades, "Efficient location prediction in mobile cellular networks," *Int. J. Wireless Inf. Netw.*, vol. 19, no. 2, pp. 97–111, Jun. 2012.

SAFIQUL ISLAM received the B.Sc. degree from the Department of Computer Science and Engineering, Green University of Bangladesh, Dhaka, Bangladesh, in 2022. He is currently a member of the Green Computing and Communication (GCC) Research Group, Department of Computer Science and Engineering, Green University of Bangladesh. His research interests include optimization, artificial intelligence, mobile device cloud, network function virtualization, and mobile edge computing in the field of networking.

MAHADI AHAMMED received the B.Sc. degree from the Department of Computer Science and Engineering, Green University of Bangladesh, Dhaka, Bangladesh, in 2022. He is currently a member of the Green Computing and Communication (GCC) Research Group, Department of Computer Science and Engineering, Green University of Bangladesh. His research interests include optimization, artificial intelligence, mobile device cloud, network function virtualization, and mobile edge computing in the field of networking.

NURA ALAM SIDDIQUE received the B.Sc. degree from the Department of Computer Science and Engineering, Green University of Bangladesh, Dhaka, Bangladesh, in 2022. He is currently a member of the Green Computing and Communication (GCC) Research Group, Department of Computer Science and Engineering, Green University of Bangladesh. His research interests include optimization, artificial intelligence, mobile device cloud, network function virtualization, and mobile edge computing in the field of networking.

PALASH ROY received the B.Sc. and M.Sc. degrees from the Department of Computer Science and Engineering, University of Dhaka, in 2019 and 2022, respectively. He is currently a Lecturer with the Department of Computer Science and Engineering (CSE), Green University of Bangladesh. He is also a member of the Green Networking Research Group (GNR), Department of CSE, University of Dhaka. His research interests include mobile device cloud, network function virtualization, mobile edge computing, and reinforcement learning. He is an active member of the IEEE Computer Society.

MD. ABDUR RAZZAQUE (Senior Member, IEEE) received the B.S. degree in applied physics and electronics and the M.S. degree in computer science from the University of Dhaka, Bangladesh, in 1997 and 1999, respectively, and the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in August 2009. From 2010 to 2011, he was a Research Professor with the College of Electronics and Information, Kyung Hee University, South Korea. From 2016 to 2021, he was with the Green University of Bangladesh at different periods as a Pro Vice-Chancellor, the Dean of the Faculty of Science and Engineering, and the Chairperson of the Department of Computer Science and Engineering (CSE). He was a Visiting Professor with Stratford University, Falls Church, VA, USA, in 2017. He is currently a Professor with the Department of CSE, University of Dhaka. He is also the Director of the Green Networking Research Group (http://cse.du.ac.bd/gnr), Department of CSE, University of Dhaka. He has published more than 140 research papers in international conferences and journals. His research interests include modeling, analysis, and optimization of wireless networking protocols and architectures, mobile crowdsourcing, sensor data clouds, the Internet of Things, and edge computing. He is a TPC Member of IEEE HPCC, ICOIN, SCALCOM, SKIMA, ICIEV, ADM, NSysS, and ICACCI. He is a member of the IEEE Computer Society. He is the General Chair of STI 2021–2019 and the TPC Chair of ICIET 2019–2018. He was chairing the 2021 Executive Committee for the IEEE Computer Society Bangladesh Chapter. He is an Associate Editor of IEEE Access. He is an Editorial Board Member of the *Journal of Networks and Applications*.

MOHAMMAD MEHEDI HASSAN (Senior Member, IEEE) received the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in February 2011. He is currently a Full Professor with the Department of Information Systems, College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Saudi Arabia. He has authored or coauthored more than 180 publications, including refereed IEEE/ACM/Springer/Elsevier journals, conference papers, books, and book chapters. He has authored and coauthored more than 365 publications including refereed journals (333 SCI/ISI-Indexed Journal papers, 42 conference papers, one book, and two book chapters. His research interests include cloud computing, edge computing, the Internet of Things, body sensor networks, big data, deep learning, mobile cloud, smart computing, wireless sensor networks, 5G networks, and social networks. He has served as the Chair and a Technical Program Committee Member in numerous reputed international conferences/workshops, such as IEEE CCNC, ACM BodyNets, and IEEE HPCC. He was a recipient of several awards, including the 2021 Outstanding Editors Award from *Future Generation Computer Systems* journal, the Distinguished Research Award from the College of Computer and Information Sciences, KSU, in 2020, the Best Conference Paper Award from the IEEE International Conference on Sustainable Technologies for Industry 4.0 (STI) 2020, the Best Journal Paper Award from IEEE Systems Journal in 2018, the Best Conference Paper Award from CloudComp, in 2014 Conference and the Excellence in Research Award from the College of Computer and Information Sciences, KSU, in 2015 to 2016. He is one of the top 2% Scientists in the world in the networking and telecommunication field. He is one of the top computer scientists in Saudi Arabia as well. Recently, his six publications have been recognized as the ESI Highly Cited Papers.

KASHIF SALEEM received the B.Sc. degree in computer science from Allama Iqbal Open University, Islamabad, Pakistan, in 2002, the P.G.D. degree in computer technology and communication from Government College University, Lahore, Pakistan, in 2004, and the M.E. degree in electrical engineering electronics and telecommunication, and the Ph.D. degree in electrical engineering from the University of Technology Malaysia, in 2007 and 2011, respectively. Since 2012, he has been engaged with the Center of Excellence in Information Assurance (CoEIA), King Saud University, Saudi Arabia, as an Assistant Professor, and was promoted to Associate Professor in 2018. Recently, he joined the School of IT & Engineering at Melbourne Institute of Technology (MIT) in February 2024 as a Full Time Associate Professor of Networking. He is professionally certified by the Massachusetts Institute of Technology (MIT) in cybersecurity, the University of the Aegean in information and communication security, IBM in security intelligence analysis, and Microsoft and Cisco in computer networks. He acquired several research grants in Saudi Arabia, the EU, and other parts of the world. He has authored or coauthored more than 130 papers in refereed journals and international conferences. His research interests include ubiquitous computing, mobile computing, the Internet of Things (IoT), machine-to-machine (M2M) communication, wireless mesh networks (WMNs), wireless sensor networks (WSNs), and mobile ad hoc networks (MANETs), intelligent autonomous systems, information security, and bioinformatics. He served as a technical program committee member and organized numerous international workshops and conferences. He is providing services as an Associate Editor mainly to *Alexandria Engineering Journal*, *Journal of Multimedia Information System* (JMIS), IEEE Access, *International Journal of E-Health and Medical Communications* (IJEHMC), and *International Journal of Cyber-Security and Digital Forensics* (IJCSDF).