Dynamic Multilevel User Allocation in MEC Using CESO for Resource Efficiency and QoE

V Arun*, M Azhagiri
Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India

Abstract*—Mobile Edge Computing (MEC) has become one of the key paradigms to enable next-generation networks in supporting applications that are latency sensitive and computation-intensive. Nevertheless, the resourceful placement of heterogeneous and dynamically incoming user tasks with distributed edge servers is a problematic issue to be achieved because of network fluctuation, non-uniform resource availability, and variance in Quality of Experience (QoE) demand. To overcome these constraints, this study suggests the Dynamic Multilevel User Allocation Algorithm (DMUAA) that incorporates a new Cognitive Evolutionary Synergy Optimization (CESO) framework in order to reach stable, adaptive, and resource-optimizing allocation in real-time. DMUAA means a hierarchical optimization pipeline that consists of heuristic initialization, stochastic refinement, and strategic game-theoretic equilibrium assisted by a coordination and feedback mechanism that guarantees the constant adaptation to variations in user mobility and load. The system model collaboratively optimizes the latency, energy, resource, and QoE under the multi-constraint edge-server conditions. Extensive simulations over a wide range of resource capabilities, user rates, and mobility patterns indicate that DMUAA can be greatly superior to five state-of-the-art baselines, which are the MGGO, GTA, EUA, HAILP, and LGP. Findings indicate that DMUAA decreases average end-to-end latency by 18-34%, increases Resource Utilization Efficiency (RUE) by 12–27%, and increases Service Continuity Rate (SCR) by 15–30% over the current practices. The solved approach also produces 20-35% greater QoE, better load balancing (with up to 25% reduced LBI), and up to 22 per cent greater energy-QoE efficiency (EQR). Moreover, CESO allows for more rapid and stable convergence, and DMUAA comes to optimal allocation states 40-55% quicker than competing algorithms.

Keywords—Mobile Edge Computing; distributed edge server; Quality of Experience; Cognitive Evolutionary Synergy Optimization; game-theoretic equilibrium

I. INTRODUCTION

Mobile Edge Computing (MEC) has become an essential architectural paradigm in contemporary distributed networks, which allows the end users to bring computation, storage, and intelligence nearer. As the number of latency-sensitive applications, including autonomous navigation, extended reality (XR), smart healthcare monitoring, industrial internet of things automation and mission-critical cyber-physical systems continue to grow, the ineffectiveness of traditional cloud-centric computing is becoming more pronounced [1]. MEC offers quite a fast solution to communication delays, backbone congestion and Quality of Experience (QoE) to users working in dynamic and resource-constrained settings by deploying computation at the network edge [2]. With the ongoing development of 5G and beyond-5G networks, MEC has become a fundamental part of highlighting the concept of massive device connectivity, guaranteeing an ultra-reliable low-latency communication (URLLC) and facilitating real-time analytics at scale. Although MEC environments are paradigm-shifting, they have a number of inherent challenges that act against their potential [3]. First, user demands are heterogeneous, and their latency sensitivities, energy levels, and mobility patterns vary, thus making it extremely difficult to solve task offloading decisions. Second, the fragmentation and imbalance of resources across distributed edge servers tend to cause inefficiency in using them, overloading servers or underutilizing the available computing capacity. Third, the volatility of wireless channels, the changing network conditions and the changing arrival rates of tasks demand constant adaptation in allocation choices that cannot be managed effectively by traditional models of optimization that are kept at a single stage or at a single point in time. Moreover, the realization of high QoE ensures and reduces energy consumption brings about a multi-objective trade-off that is rather difficult to solve in both large-scale or dynamic MEC topology [4-6]. Lastly, the stability, fairness and convergence in the allocation strategy are still a major challenge to be worried about because of the interaction between user competition, limited-edged resources, and decentralization of decision-making processes. These issues demonstrate the necessity of a multi-level, and adaptive allocation framework that is intelligent and able to tackle the complexity of the MEC ecosystems in its essence [7].

The complexity and heterogeneity of contemporary MEC environments leads to the incentive to have a more intelligent and adaptive user allocation strategy. The current allocation methods, such as heuristic scheduling, mathematical programming, meta-heuristic scheduling, and game-theory-based methods, are frequently unable to respond to swift changes in the user mobility, dynamic workload profiles, and uneven resource distributions across edge servers. Most modern methods are based on the assumption of static optimization, restricted feedback fusion, or one-stage decision processes, leading to inefficient allocation, large latency, and energy wastage, as well as unpredictable behavior in actual conditions. These constraints are magnified with increase in device densities and the presence of computationally intensive cost of such services within cities that are smart, autonomous systems, and industrial deployments [8]. The necessity of scalable and responsive approach that could continuously adapt to change of environmental conditions is the root cause of motivation of this research. In addition to those issues, the main idea of this study is to construct a solid and multi-tiered allocation system that guarantees effective, equitable, and Quality of Experiences-based distribution of user workload among diverse edge servers. In particular, the following objectives are aimed at: 1) minimizing end-to-end latency and power usage under dynamic network scenarios, 2) better utilization and load balancing of resources among distributed servers, 3) steady and equitable allocation by equilibrium-based decision-making, and 4) fast convergence and self-adaptation by using an iterative, feedback-constrained optimization. All these aims are aimed at the creation of an improved allocation system that can overcome the fundamental constraints of the current edge computing systems.

This study presents the Dynamic Multilevel User Allocation Algorithm (DMUAA), which is a comprehensive and adaptive framework aimed at addressing the clear drawbacks of the available edge user-allocation approaches. This work has been novel in that it incorporates the Cognitive Evolutionary Synergy Optimization (CESO) paradigm - a three-stage optimization pipeline which synergistically integrates heuristic reasoning, stochastic exploration, and strategic game-theoretic equilibrium. In contrast to non-dynamically changing allocation methods, such as traditional single-phase systems or non-dynamically changing approaches, DMUAA can update allocation decisions in real-time using resource dynamics, QoE dynamics and real-time feedback, facilitating the achievement of better adaptability in the context of very volatile MEC environments. The integration of a well-organized coordination layer and closed-loop feedback mechanism provides the framework with an extra boost of ensuring a coherent optimization of all stages and a quick stabilization process under a changing workload. The key contributions of this research can be summarized as follows:

- A novel multilevel allocation architecture that unifies fast heuristic initialization, probabilistic refinement, and equilibrium-driven stabilization to provide robust and dynamic user–server mapping.
- A state-aware CESO optimization engine that ensures scalable, diversified, and convergence-guaranteed decision-making.
- A comprehensive system model that jointly optimizes latency, energy, QoE, and resource fairness through multi-objective constraints.
- Large comparative analysis showing that DMUAA is much better than five state-of-the-art baselines in terms of latency, QoE, energy efficiency, load balancing, scalability, and convergence rate.

Despite significant advances in user allocation strategies for mobile edge computing, existing approaches often focus on either heuristic efficiency, stochastic exploration, or strategic stability independently, limiting their ability to simultaneously achieve high resource utilization, stable QoE, and scalability under dynamic conditions. Therefore, the central research question addressed in this study is: How can a dynamic multilevel user allocation framework be designed to jointly optimize resource efficiency, QoE, and system stability in Mobile Edge Computing environments under varying workloads and user mobility? To answer this question, this work proposes a Dynamic Multilevel User Allocation Algorithm (DMUAA) integrating a Cognitive Evolutionary Synergy Optimization (CESO) process that combines adaptive allocation, stochastic refinement, and equilibrium-driven decision-making within a unified framework. These contributions, taken together, make DMUAA a scalable, powerful, and QoE-driven solution to next-generation MEC environments.

The rest of the study has the following structure: Section II draws a related literature review on edge user allocation, which identifies methods of existing heuristic, meta-heuristic, and game-theoretic methods. Section III outlines the suggested Dynamic Multilevel User Allocation Algorithm (DMUAA) in detail, system model, architectural design, as well as the CESO-based multilevel optimization process. Section IV presents the experimental findings, results of performance comparison, and analysis insights of five state-of-the-art methods. Section V summarizes the main conclusion of the research and outlines the possible future research directions in scalable and intelligent management of MEC resources.

II. Related Works

Mobile Edge Computing (MEC) has become an important concept, to facilitate the provision of offloading computation with low-latency, energy efficiency, and quality of experience (QoE) to a large number of users. Nevertheless, the heterogeneity of edge servers as a given fact, together with the high dynamism of user mobility, changing task distributions, intermittent access to resources, and unpredictable communication circumstances, renders the allocation problem of users highly demanding. Most conventional solutions of the MEC are typically based on past optimization methods, single-level models of decision, or greedy methods that do not address changes in network conditions in real time [9]. Even more advanced meta-heuristic or game-theoretic techniques have been found to be limited with respect to the handling of rapidly changing workload, probabilistic arrivals of the tasks, and multi-objective trade-offs among QoE, latency, energy usage, and load balancing. More recent studies have tried to solve some of the parts of the problem of user allocation in the MEC, including reducing the energy use, maximizing throughput, optimizing the selection of servers or stabilizing user-server mapping based on strategic rationale. Regardless of these developments, the majority of solutions that exist are still facing a problem with scalability, slow convergence, failure to provide continuous adaptation, and inability to incorporate real-time feedback of changing network states [10]. Such consistent gaps drive the design of a highly developed, multi-layered, cognitively directed optimization framework with the capacity to provide consistent and efficient allocation decisions in varied MEC operating scenarios.

The study of edge user allocation (EUA) and computation offloading has taken a variety of methodological paths, each trying to solve the chain of interaction of latency, resource constraints, mobility, and QoE. The leading solution families are meta-heuristic optimization, game-theoretic allocation, mathematical programming, and learning-based methods, which are all inherently limited in their applicability to dynamic MEC systems. The use of meta-heuristic optimization has also received a great deal of interest because it allows dealing with nonlinear, multi-objective allocation problems. Khosrowshahi et al. [11] developed this direction, introducing an adaptive partitioning and stagnation detection to improve exploration, the Modified Greylag Goose Optimization (MGGO). Although MGGO increases the latency and energy indicators, its performance is sensitive to parameter optimization, and under unstable network conditions, its convergence is not guaranteed. Likewise, Tingting et al. [12] used the Fruit Fly Optimization Algorithm (EUA-FOA) to optimize user admission and minimize server activation, but the proximity and capacity-based model of this algorithm neither considers the dynamics of QoE nor the stability of the equilibrium, which restricts its applicability to real-time user mobility.

A second key type of research develops EUA as a strategic interaction between users and servers based on the equilibrium analysis. Hou et al. [13] came up with a multi-hop game-theoretic allocation model (GMUA) and established the existence of equilibrium, yet its convergence rate decreases with user density. Xuyun et al. [14] also constructed EUAGame, a possible game structure that would allow decentralized allocation, but its quality of equilibrium is limited by the formulation of the game and does not consider how to deal with stochastic or bursty demand models. Zhou et al. [15] generalized game theory to fairness-sensitive applications with FEUAGame; though this method offers guarantees of fairness at the cost of responsiveness, especially in highly-loaded systems. In contrast to game-theoretic methods, mathematical programming has been studied in order to represent the combinatorial aspects of EUA. Phu et al. [16] proposed a formulation of an Integer Linear Programming (ILP) with the QoE-based constraints. Despite its correspondence to small-scale situations, ILP can quickly be intractable as the size of the system increases, leading to heuristic relaxations that compromise optimality. Qiang et al. [17] modeled EUA as a bin-packing problem based on Lexicographic Goal Programming (LGP); the model is structured and interpretable, but based on the static optimization assumptions and with poor adaptation to real-time. In the near past, learning-based and hybrid intelligent models have been presented to learn long-term trends, multi agent dynamics and nonlinear rewards. Li et al. [18] capitalized on deep reinforcement learning (DRL) to optimize the decisions of NOMA user grouping and offloading, which exhibited significant performance improvement at the expense of high training cost. Heidari et al. [19] introduced the lightweight RL model that is accelerated using CNNs, but its use is limited because it relies on the device-side computation. Liu et al. [20] scaled DRL to multi-objective multi-UAV-MEC systems based on Double/Dueling DQNs, but which demand large state representations and training sets, which limits their applicability to dense deployment of terrestrial MEC systems.

There are other works that have attempted to deal with system-level dynamics by more specific mechanisms. Tian et al. [21] established a response-ratio offloading strategy (RROS) which is user preference-guided, but its greediness in prioritization gets congested locally. Baskar et al. [22] suggested a hybrid Prairie dog -Dwarf Mongoose meta-heuristic that achieves better exploitation-exploration balance at a significant computational cost. Guangming et al. [23] came up with OL-EUA, which is an online method of allocating resources to NOMA systems but its workability falls short in regimes with sudden contention of resources or when tasks have heavy tails. Chen et al. [24] followed a Lyapunov-based long-term optimization method coupled with distributed game theory, which has good theoretical grounds, but models must assume stationarity which is difficult to satisfy in highly dynamic MEC networks. History-assisted user allocation (HOUA) was proposed by Bowen et al. [25], but due to the use of past patterns, it is not flexible enough to work in the fast-changing conditions. Through the variety of solution families one can see, however, a similarity: current solutions to the problem of MEC user allocation are partial. Meta-heuristics are good at searching through extensive search spaces but not convergence stable; game-theoretic and mathematical models are stable but can scale poorly or can be non-adaptive; learning-based strategies are long-term optimizing, but incur very high computational costs and slow convergence. None of them offers an integrated, multi-level, feedback-based, dynamically convergent architecture that can help to balance latency, QoE, resource fairness, energy efficiency, and equilibrium stability in MEC conditions that vary rapidly.

The literature review indicates that despite the tremendous achievement by MEC user allocation, there are still some gaps that are vital to the effectiveness of current solutions. In the current methods, the optimization mechanisms are largely single-phase optimization, whether heuristic, meta-heuristic, mathematical programming, or game-theoretic. This type of standalone strategies does not fully reflect the multi-dimensionality of the allocation problem, which requires quick generation of feasibility and more refinement before settling at an equilibrium point. Moreover, since integration of real time feedback is not available then most models are not well suited to dynamic changes in mobility of users, workload bursts or variability of resources- environments that characterize real MEC implementations. Meta-heuristic algorithms do not tend to converge well, whereas game-theoretic models, despite being theoretically sound, slow to converge, or rely on constraining assumptions such as fixed load distributions, or idealistic user behavior. The second important weakness is that they do not optimize multi-objectives as most of the previous research tends to optimize latency, energy, or user experience but rarely all of them within one framework. Scalability is also an issue: ILP-based models are computationally infeasible in large-scale systems, DRL-based strategies are too slow to train with large-scale systems, and complicated hybrid heuristics are also computationally expensive. Lastly, there is no current literature that shows a cognitive or evolutionary synergy model that incorporates heuristic thinking, search by stochastic methods and equilibrium-seeking behavior in a coordinated and hierarchical process. As a combination of these gaps, there is a need to consider a more adaptive, multi-stage, feedback-based, and equilibrium-focused user allocation approach.

The proposed Dynamic Multilevel User Allocation Algorithm (DMUAA) addresses the shortcoming of the current MEC strategies by a cognitively-inspired, evolutionary coordination design optimization. Simply put, Cognitive Evolutionary Synergy Optimization (CESO) framework is an integrated process comprising three complementary steps: 1) a heuristic step during which an initial feasible allocation baseline is quickly achieved; 2) a stochastic refinement step during which greater diversity is achieved and local optima are avoided; and 3) a strategic equilibrium step during which a fair and game-theoretically stable allocation equilibrium is obtained. Unlike the previous single-stage or ad hoc models, DMUAA uses a feedback system that modulates the decisions in real-time depending on the QoE variations, mobility variations as well as resource variation. The unified formulation jointly optimizes latency, QoE, energy usage, and load balance, while its hierarchical structure ensures scalability and robust convergence.

III. Proposed Framework

The proposed DMUAA framework resolves the critical issues of the user allocation and the resource use optimization in edge computing conditions, when the user demands are in heterogeneous, the network conditions are dynamic, and the computational resources are limited. The framework embraces a multilevel structured design that conveniently incorporates autonomous decision-making, adaptive optimization, and continuous feedback to improve the overall system operation and the Quality of Experience (QoE). DMUAA is an entity that is based on a layered architecture that consists of four main functional layers namely the User Device Layer, Edge Server Layer, Optimization Layer and Coordination Layer, illustrated by Fig. 1. The various layers have their roles, unique but interdependent, to coordinate a unified pipeline that takes care of the entire user allocation lifecycle, through request generation to the performance feedback. The User Device Layer has heterogeneous end-user devices (smartphones, IoT nodes and sensors) that place service requests, with each request having distinct computational and latency needs. These requests are sent to the Edge Server Layer where edge servers are located geographically which process and store data at locations near the point of demand and considerably improve response time and reduction of backbone network congestion [26]. This layer includes a Resource Manager which dynamically monitors server load and availability and a QoE Monitor which measures user satisfaction and system responsiveness to generate a real time performance map of the edge environment.

DMUAA is computationally founded on the Optimization Layer. It uses a Cognitive Evolutionary Synergy Optimization (CESO) that gradually anneals the user-server allocations through a sequence of intelligent decision makings. Initial hurricane plans are meant to yield instant and realistically feasible allocation base that can ensure the stabilized operation in the short run. This baseline is then optimized over with probabilistic and adaptive optimization logic that tries to explore other combinations, and thus leads to increased diversity in the system and eliminates suboptimal convergence. Finally, a strategic decision mechanism can stabilize at equilibrium which is equitable and maximization on resources is appropriate across all users and servers. Such a hierarchical optimization approach enables DMUAA to balance the allocation accuracy and computation efficiency. And most importantly, there is the Coordination Layer, which governs the flow of decisions and guarantees the system consistency over and above the optimization process. The User Allocation Coordinator is the one who coordinates the activities of a group of optimizers and each stage of optimization is duly grounded on the previous phase. Decision Aggregator chooses the outcome of the different optimization steps and the final allocation will be the one that will optimize the system objectives in resource efficiency, the minimization of the latency and the maximization of the QoE. A Feedback loop is a performance data and user experience feedback received continuously and fed back into the optimization pipeline to enable the optimization pipeline to adjust to new network conditions or user mobility. In simple terms, the DMUAA framework offers dynamic relationship between the system-level coordination mechanisms and the intelligent optimization modules. In multilevel decision processing coupled with incentive continuous performance feedback, DMUAA ensures that user allocations will become global optimum and QoE-optimal. This architecture together with scaling and responsiveness also enables the framework to be healthily operating in highly dynamic and resource limited edge computing environments.

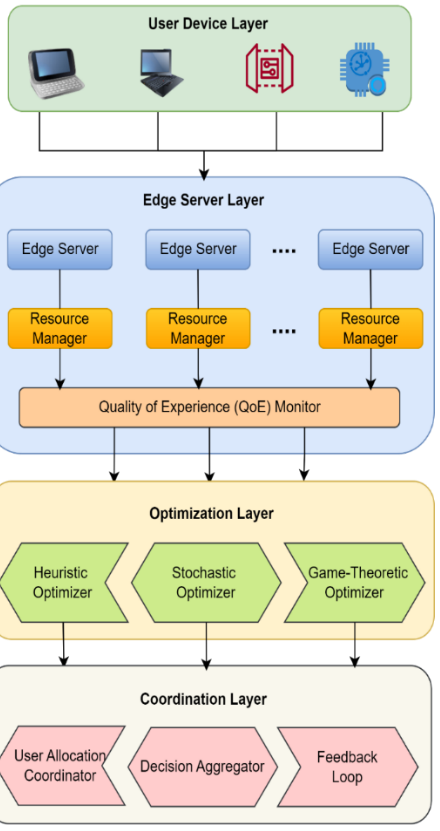

Fig. 1. Dynamic Multilevel User Allocation in edge computing systems.

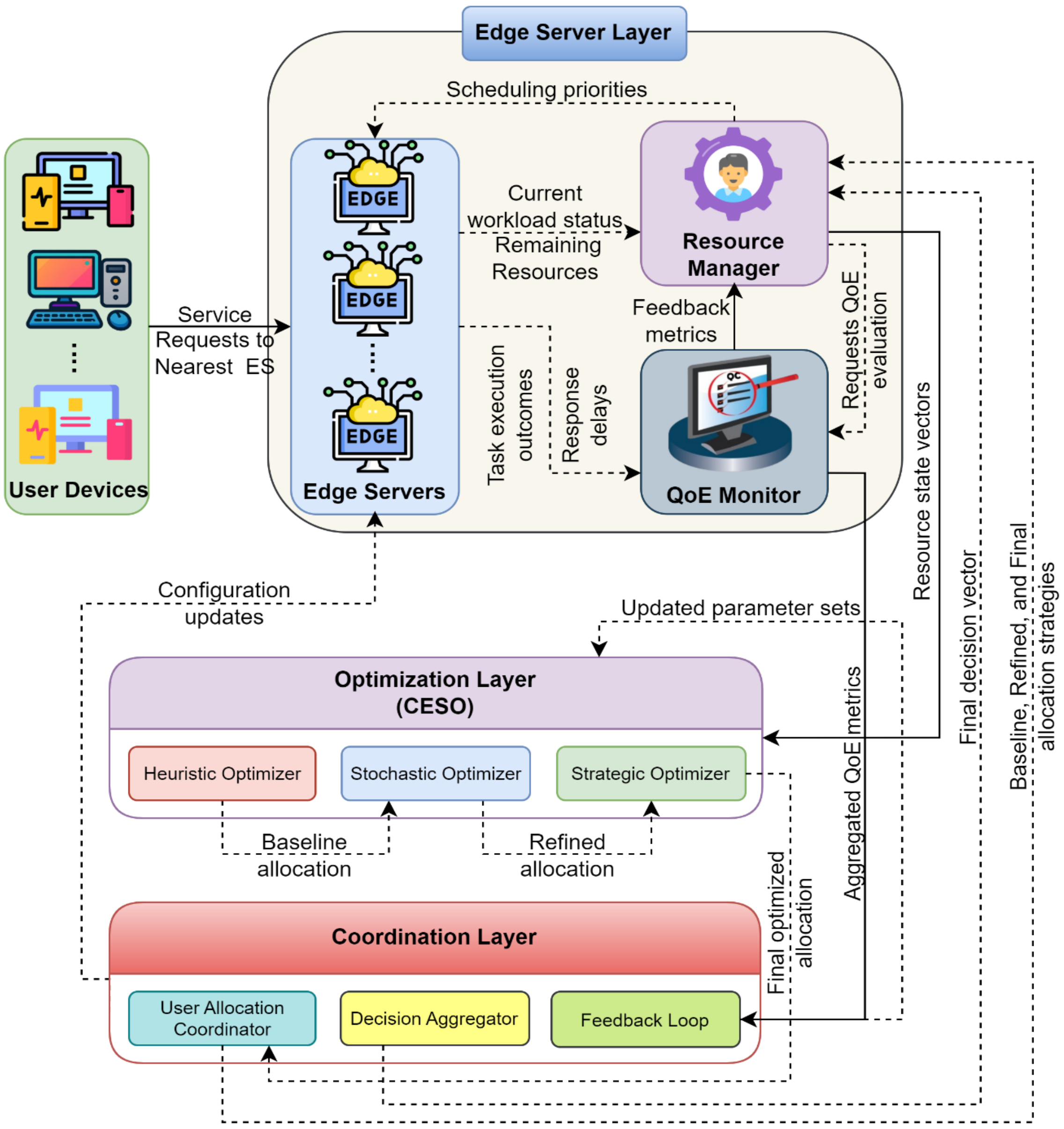

Fig. 2. Overall architecture of the Dynamic Multilevel User Allocation Algorithm (DMUAA).

A. System Model and Problem Formulation

The system model for the proposed Dynamic Multilevel User Allocation Algorithm (DMUAA) is designed to represent a realistic mobile edge computing (MEC) environment comprising distributed edge servers and dynamically interacting user devices [27]. The framework aims to optimize user-to-server allocation while minimizing resource contention, latency, and energy consumption, thereby enhancing overall Quality of Experience (QoE), as shown in Fig. 2. Let there be a set of n users and m edge servers denoted as $U = \{u_1, u_2, \dots, u_n\}$, $S = \{s_1, s_2, \dots, s_m\}$. Each user $u_i \in U$ generates a computational task characterized by its resource demand vector: $\omega_i = [\omega_i^{cpu}, \omega_i^{mem}, \omega_i^{bw}]$, where ω_i^{cpu}, ω_i^{mem}, and ω_i^{bw} denote the CPU cycles, memory space, and bandwidth required for the task execution, respectively. Each edge server $s_j \in S$ possesses a finite resource capacity vector: $\tau_j = [\tau_j^{cpu}, \tau_j^{mem}, \tau_j^{bw}]$, representing its maximum available resources. A binary allocation variable is defined as in Eq. (1). To ensure feasibility, each user can be allocated to only one edge server within its communication range, expressed as in Eq. (2):

$$x_{ij} = \begin{cases} 1, & \text{if user } u_i \text{ is allocated to edge server } s_j, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

$$\sum_{j=1}^{m} x_{ij} \leq 1, \forall i \in \{1,2,\dots,n\}. \quad (2)$$

Each edge server's total allocated resource utilization must not exceed its available capacity, as defined in Eq. (3); this ensures that no server is overloaded across any resource dimension. For each user u_i allocated to edge server s_j, the end-to-end latency L_{ij} can be expressed, as in Eq. (4); where: L_{ij}^{tx} is the transmission delay between u_i and s_j, $L_{ij}^{proc} = \frac{\omega_i^{cpu}}{f_j}$ is the processing delay based on the server's computation frequency f_j, L_{ij}^{queue} denotes the queuing delay at the server caused by concurrent task arrivals. The transmission delay is computed as defined in Eq. (5); where D_i represents the data size of the user request, B_{ij} is the allocated channel bandwidth, and γ_{ij} is the signal-to-noise ratio (SNR) between user u_i and server s_j.

$$\sum_{i=1}^{n} \omega_i^r x_{ij} \leq \tau_j^r, \forall j \in \{1,2,\dots,m\}, r \in \{cpu, mem, bw\}. \quad (3)$$

$$L_{ij} = L_{ij}^{tx} + L_{ij}^{proc} + L_{ij}^{queue}, \quad (4)$$

$$L_{ij}^{tx} = \frac{D_i}{B_{ij} \cdot log_2(1+\gamma_{ij})}, \quad (5)$$

The energy consumed by a user during task offloading is given by: $E_{ij} = P_i^{tx} \cdot L_{ij}^{tx}$, where P_i^{tx} is the transmission power of user u_i. The total energy consumption of all users in the system is defined in Eq. (6). The QoE for each user depends on response latency, energy efficiency, and service satisfaction. It can be modeled as defined in Eq. (7), where: L_{max} and E_{max} denote maximum tolerable delay and energy levels, S_{ij} represents a satisfaction index derived from service reliability and data rate, $\alpha_1, \alpha_2, \alpha_3$ are weighting factors ($\alpha_1 + \alpha_2 + \alpha_3 = 1$) reflecting user sensitivity toward delay, energy, and satisfaction. The total system QoE is then given by Eq. (8):

$$E_{total} = \sum_{i=1}^{n} \sum_{j=1}^{m} E_{ij}\, x_{ij}. \quad (6)$$

$$Q_i = \alpha_1 \cdot \left(1 - \frac{L_{ij}}{L_{max}}\right) + \alpha_2 \cdot \left(1 - \frac{E_{ij}}{E_{max}}\right) + \alpha_3 \cdot S_{ij}, \quad (7)$$

$$Q_{total} = \sum_{i=1}^{n} \sum_{j=1}^{m} Q_i\, x_{ij}. \quad (8)$$

The main goal of DMUAA is to ensure maximum overall system QoE and a minimum number of active edge servers and resource balance. The optimization problem can be stated formally as Eq. (9), where C_{active} denotes how many servers are being used and serve users at the moment and C inactive denotes the servers that are not in use. The formulation provided above is an NP-hard and multi-objective optimization problem with discrete allocation variables and non-linear user-server dependencies. To overcome this difficulty, DMUAA uses a multilevel adaptive optimization procedure with heuristic initialization, stochastic experimentation and strategic stabilization processes interactively approximating to an efficient and stabilized solution.

$$\text{Maximize: } F = \frac{Q_{total}}{C_{active}}, \quad (9)$$

subject to Eq. (10):

$$\begin{cases} \sum_{i=1}^{n} \omega_i^r x_{ij} \leq \tau_j^r, & \forall j, r, \\ \sum_{j=1}^{m} x_{ij} \leq 1, & \forall i, \\ x_{ij} \in \{0,1\}, & \forall i, j, \end{cases} \quad (10)$$

B. Architecture Design of DMUAA

1) User Device Layer: The User Device Layer constitutes the interface of the system, which includes distributed heterogeneous user terminals, which include smartphones, IoT sensors, and autonomous devices. Such devices will never cease requesting computational and service requests that demand low-latency and high Quality of Experience (QoE). The devices send their request packets to the closest edge server using the network proximity and signal strength and the parameters of the request, such as task size τ_i, latency tolerance L_i, and priority weight ω_i. The layer also provides mobility management that is dynamic, such that the user devices can easily move between edge servers without service disruption. It also has a small local cache to store temporary data on intermediate tasks in transition of the allocation.

2) Edge Server Layer: The Edge Server Layer represents the computational backbone of DMUAA. It consists of distributed edge servers $S = \{s_1, s_2, \dots, s_m\}$ located near end-users. Each server provides compute capacity C_j, bandwidth B_j, and storage M_j. Within this layer, three key sub-modules operate synergistically:

- Edge Servers: Process user tasks and queuing requests. Every server has a local task-scheduler which dynamically allocates its resources to the admitted users.
- Resource Manager: This is the interface between the edge layer and the optimization engine. It gathers load measurements ρ_j, active connections n_j, and the fraction of available resources. It subsequently sends these measures to the upper optimization layer in order to make decisions.
- QoE Monitor: It is a monitoring tool that constantly measures the user experience parameters, including response time T_i, throughput, and perceived satisfaction Q_i. The monitor sends data that is the aggregate QoE feedback to the Feedback Loop in the coordination layer, which can be used in adaptive learning and optimization.

3) Optimization Layer: The Optimization Layer represents algorithmic intelligence of DMUAA. It uses Cognitive Evolutionary Synergy Optimization (CESO) to optimize user-server associations by utilizing three processes that are closely linked:

- Heuristic Optimizer: Generates an initial feasible allocation by rapidly mapping users to servers based on resource availability, latency thresholds, and service proximity.
- Stochastic Optimizer: This is an improvement of the heuristic solution that tries other configurations with controlled randomness. It is likely to settle better allocations that optimize system QoE or balance workloads.

- Strategic Optimizer: This establishes a global stable equilibrium between users and server by adjusting the strategy. This makes the allocation final, efficient and stable, in case of dynamic demand.

These optimizers are used in multistage pipeline, wherein the output of one is used as the input of the other. The combination of the two creates a hybrid decision engine, which approaches an optimal and equilibrium-based state of allocation.

4) Coordination Layer: The Coordination Layer ensures global coherence, synchronization, and continual adaptation across all lower layers. It integrates three control components:

- User Allocation Coordinator: Manages the interaction between optimizers, controls iteration flow, and resolves conflicts when multiple optimization outcomes overlap.
- Decision Aggregator: Consolidates intermediate results from the heuristic, stochastic, and strategic optimizers. It selects the most beneficial allocation decision based on predefined evaluation metrics such as total QoE $\sum_i Q_i$, system utility U_{sys}, and resource balance ratio R_b.
- Feedback Loop: Gathers real-time system data and QoE reports on the Edge Server Layer. The feedback system is used to adjust the optimization parameters λ, η, and μ dynamically in controlling the sensitivity of each optimizer to the latency, load, and QoE, so that the optimizers can be tuned adaptively.

5) Inter-layer communication flow: The DMUAA architecture works with the ongoing two-way communication between the layers. Task requests are passed upwards by user devices to edge servers, which pass system state vectors to the optimization layer through the Resource Manager. Optimizers generate better allocation plans and relay them down via the Coordination Layer. Decision Aggregator completes the strategy and the revised allocation map is sent to the edge servers, where implementation is done. At the same time, QoE measurements and system statistics are passed on up the Feedback Loop, which is a closed-loop adaptive cycle that constantly improves the performance and responsiveness.

C. Cognitive Evolutionary Synergy Optimization (CESO) Process

The DMUAA presents a hierarchical optimization model that applies three complementary computational paradigms, including: heuristic, stochastic and strategic optimization, in a coordinated and closed system. Such a multi-level design ensures that the allocation process is not only computationally efficient but also adaptive, robust and QoE-oriented and balances user demands with the availability of the system resources in real-time. Optimization is done in iterative steps, with every level being the refinement of the output of the previous level, with each step getting closer to the global optimum. Formally, the Cognitive Evolutionary Synergy Optimization (CESO) can be defined as an iterative function, as defined in Eq. (11), where: $\Phi_{HE}(\cdot)$ represents the Heuristic Optimization Function, producing a feasible initial allocation. $\Phi_{ST}(\cdot)$ denotes the Stochastic Refinement Function, enhancing diversity and escaping local optima. $\Phi_{SE}(\cdot)$ indicates the Strategic Equilibrium Function, ensuring stability and fairness through game-theoretic balancing. $A^{(t)}$ is the allocation matrix at iteration t, and $A^{*(t+1)}$ is the optimized allocation after one complete multi-level cycle. Every optimization stage is based on real-time dynamics of the system that is characterized by mobility of users, changes in workloads and network fluctuations. Therefore, optimization process of DMUAA is state-dependent, in which the system evolves with respect to the following equation: Eq. (12). Here, S(t) is the state of the system (resource usage, load allocation, and latency) in this case, $\mathcal{F}$ is the system transition function, and 0 contains adaptive parameters (tolerance thresholds, sensitivity coefficients, and others). The optimization process continues until the equilibrium condition is reached: $\| A^{*(t+1)} - A^{*(t)} \| < \epsilon$ where ϵ is the convergence threshold ensuring that further updates yield negligible improvements in overall system utility U_{sys}. The subsequent sub-sections detail the functioning of each phase within this hierarchical framework, beginning with the Initialization Phase that sets up the foundation for Cognitive Evolutionary Synergy Optimization.

$$A^{*(t+1)} = \Phi_{SE}\left(\Phi_{ST}\left(\Phi_{HE}\left(A^{(t)}\right)\right)\right) \tag{11}$$

$$S(t+1) = \mathcal{F}\left(S(t), A^{*(t)}, \theta\right) \tag{12}$$

1) Initialization Phase: The Initialization Phase is the preparation phase of DMUAA framework. It carries out three basic tasks that are: the collection of system state, parameter configuration, and the allocation matrix initiation. This makes sure that each successive optimization layer works with quality, synchronized and quantifiable inputs. At time t_0, the system aggregates real-time data from all user devices and edge servers, as defined in Eq. (13), where: $D_i = (\tau_i, L_i, \omega_i, \delta_i)$ denotes the request vector of user i(task size, latency requirement, priority, mobility). $R_j = (C_j, B_j, M_j)$ represents the resource profile of edge server j(CPU cycles, bandwidth, memory availability). N_{ij} signifies the network characteristics (e.g., transmission rate and delay) between user i and edge server j. This normalization makes it possible that different resource properties and user requirements can be combined into the same optimization model without scale prejudice [see Eq. (13) and Eq. (14)]. The initializing step is the one that prearranges the key control values according to which the optimization behavior is handled at all the further levels under listed Table I.

$$\Omega(t_0) = \left\{ D_i, R_j, N_{ij} \mid i = 1, \dots, N_u;\ j = 1, \dots, N_s \right\} \tag{13}$$

$$\hat{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{14}$$

TABLE. I. PARAMETER CONFIGURATION

Parameter	Description	Role
λ	Latency Sensitivity Coefficient	Controls trade-off between response time and resource cost.
η	Resource Utilization Weight	Balances workload distribution across servers.
μ	QoE Sensitivity Parameter	Adjusts optimization bias toward user satisfaction.
ρ_{init}	Initial Resource Threshold	Ensures that minimum resource limits are preserved.
ϵ	Convergence Tolerance	Defines the stopping criterion for iterative refinement.

The global objective function at initialization is defined as in Eq. (15), where Q_i is the QoE score of users i, T_i is the total response time, and C_i is the cost of resource consumption. This objective represents the system-wide utility that the optimization process aims to maximize. An initial feasible allocation matrix $A^{(0)} = [a_{ij}]$ is generated, as defined in Eq. (16). This allocation is computed based on the nearest-server heuristic, considering network proximity and available capacity. The resulting $A^{(0)}$ serves as the baseline allocation for the subsequent Heuristic Optimization Phase, ensuring the system begins from a feasible operational state. Finally, the system establishes two dynamic state vectors for iterative updates:

- Resource State Vector: $S_R = [C_j^{used}, B_j^{used}, M_j^{used}]$
- QoE State Vector: $S_Q = [Q_1, Q_2, \dots, Q_{N_u}]$

These vectors are continuously updated during the optimization cycle as user allocations and performance metrics evolve [see Eq. (15) to Eq. (17)].

$$\max U_{\text{sys}} = \sum_{i=1}^{N_u} (\mu Q_i - \lambda T_i - \eta C_i) \tag{15}$$

$$a_{ij} = \begin{cases} 1, & \text{if user } i \text{ is assigned to server } j \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

$$j^* = \arg\min_j \left(\frac{L_{ij}}{C_j^{avail}} + \delta_i \right) \tag{17}$$

2) Heuristic Allocation Phase: The first level of optimization of the DMUAA is the Heuristic Allocation Phase. The main aim of it is to come up with a workable and nearly optimal baseline allocation that can be optimized later on in later stages. This stage guarantees quick convergence to an effective use of resources and low latency, besides preserving the Quality of Experience (QoE) to all active users, as presented in Fig. 3. The heuristic stage assumes a definite-but-not-rigid allocation approach and is informed through the system parameters that are set in the stage before it. The process trades off computational demand, server capacity and network latency to come up with an initial allocation matrix that meets real-time performance constraints [28]. In this phase, the algorithm is aimed at reducing the total system latency, as well as balancing the computational loads among all edge servers. The optimization objective is expressed, as in Eq. (18), where: L_{ij}= transmission latency between user $i$and server j, D_i= computational demand of user i, C_j^{avail}= available computation capacity of server j, δ_i= user mobility coefficient, $\alpha_1, \alpha_2, \alpha_3$= weighting factors satisfying $\alpha_1 + \alpha_2 + \alpha_3 = 1$. The objective function Ψ_H quantifies the weighted latency–capacity trade-off, enabling efficient load distribution while maintaining low response times.

The allocation is subject to two major constraints defined as in Eq. (19) and Eq. (20). These constraints ensure that each user is assigned to exactly one edge server and that no server exceeds its available capacity.

$$\min_A \quad \Psi_H = \sum_{i=1}^{N_u} \sum_{j=1}^{N_s} a_{ij} \left(\alpha_1 L_{ij} + \alpha_2 \frac{D_i}{C_j^{avail}} + \alpha_3 \delta_i \right) \tag{18}$$

$$\sum_{j=1}^{N_s} a_{ij} = 1, \forall i \tag{19}$$

$$\sum_{i=1}^{N_u} a_{ij} D_i \le C_j^{avail}, \forall j \tag{20}$$

$$\Pi_i = \beta_1 \left(\frac{1}{L_i^{avg}} \right) + \beta_2 \left(\frac{1}{T_i^{req}} \right) + \beta_3 \omega_i \tag{21}$$

The heuristic algorithm operates through a priority-based allocation mechanism. Each user request is evaluated and ranked according to a priority index that captures urgency and resource sensitivity, as defined in Eq. (21), where, L_i^{avg}= average link latency from user $i$to all available servers, T_i^{req}= maximum allowable response time for user i, ω_i= service priority weight, $\beta_1, \beta_2, \beta_3$= weighting parameters reflecting network–QoE trade-offs. Users with higher Π_i values are given precedence in the allocation process, ensuring that latency-sensitive or high-priority tasks are served first [see Eq. (22) and Eq. (23)]. Each user $i$is then allocated to the most appropriate server j^* according to the Minimum Cost Rule. Here Q_{ij} represents the expected QoE value if user i is served by server j, estimated as defined in Eq. (24). Here, T_{ij} is the expected total response time, and U_{ij} is the user satisfaction utility derived from past allocation feedback. The weighting parameters ζ_1 and ζ_2 are chosen empirically to balance performance and satisfaction, as outlined in Algorithm 1.

$$j^* = \arg\min_j \left(\Gamma_{ij} \right) \tag{22}$$

$$\Gamma_{ij} = \alpha_1 L_{ij} + \alpha_2 \frac{D_i}{C_j^{avail}} + \alpha_3 (1 - Q_{ij}) \tag{23}$$

$$Q_{ij} = \zeta_1 \left(\frac{1}{T_{ij}} \right) + \zeta_2 U_{ij} \tag{24}$$

$$\phi_j = \frac{\sum_{i=1}^{N_u} a_{ij} D_i}{C_j^{avail}} \tag{25}$$

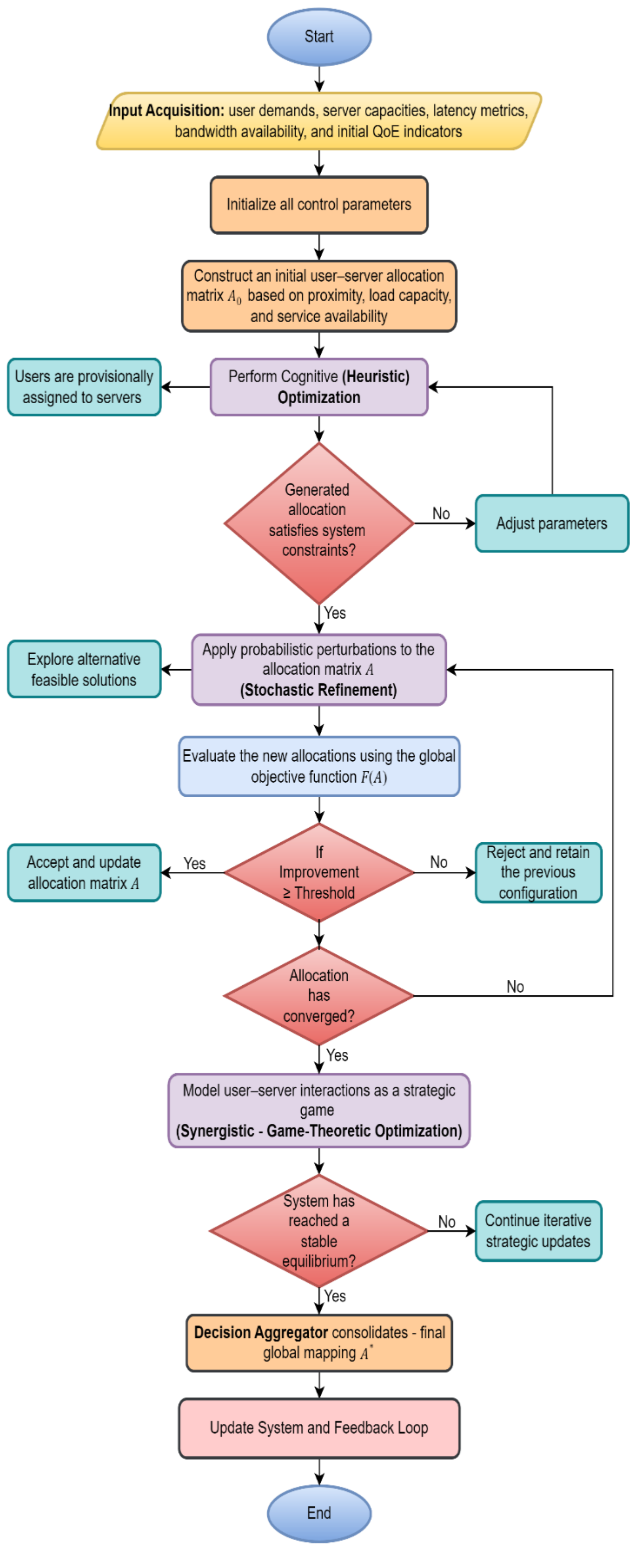
Start
Input Acquisition: user demands, server capacities, latency metrics, bandwidth availability, and initial QoE indicators
Initialize all control parameters
Construct an initial user–server allocation matrix A_0 based on proximity, load capacity, and service availability
Users are provisionally assigned to servers
Perform Cognitive (Heuristic) Optimization
Generated allocation satisfies system constraints?
No
Adjust parameters
Yes
Explore alternative feasible solutions
Apply probabilistic perturbations to the allocation matrix A (Stochastic Refinement)
Evaluate the new allocations using the global objective function $F(A)$
Accept and update allocation matrix A
Yes
If Improvement ≥ Threshold
No
Reject and retain the previous configuration
Allocation has converged?
No
Yes
Model user–server interactions as a strategic game (Synergistic - Game-Theoretic Optimization)
System has reached a stable equilibrium?
No
Continue iterative strategic updates
Yes
Decision Aggregator consolidates - final global mapping A^*
Update System and Feedback Loop
End

Fig. 3. Process flow of Heuristic Allocation Phase.

Once the initial assignments are determined, the heuristic algorithm performs adaptive load correction to maintain equilibrium across servers. Each server $j$computes its current load factor, as defined in Eq. (25). If $\phi_j > \rho_{init}$(the predefined utilization threshold), excess user requests are iteratively redirected to neighboring servers with lower utilization, following: $j' = \arg\min_{k \neq j} (\phi_k)$ until $| \phi_j - \phi_{j'} | < \delta_{bal}$ where δ_{bal} is the load balance tolerance factor. This step prevents localized overloading while maintaining a near-optimal mapping generated by the initial heuristic allocation. The final output of this phase is an initial allocation matrix $A^{(H)} = [a_{ij}]$ and a load distribution vector $\Phi = [\phi_1, \phi_2, \ldots, \phi_{N_s}]$, which collectively form the baseline input for the Stochastic Refinement Phase. At the end of this phase: $A^{(H)} \Rightarrow A_0^{(ST)}, \Phi \Rightarrow \Phi_0$ where $A_0^{(ST)}$ denotes the starting allocation for stochastic refinement. This ensures that the DMUAA begins from a stable, feasible, and balanced allocation state, enabling subsequent layers to focus on global optimality and strategic equilibrium refinement rather than feasibility. The notation and semantics used in the proposed work are shown in Table II.

TABLE. II. NOTATION AND MEANINGS

Notation	Explanation
$U = \{u_1, u_2, \ldots, u_{N_u}\}$	Set of User Devices
$S = \{s_1, s_2, \ldots, s_{N_s}\}$	Set of Edge Servers
D_i	Computational Demand of Each User u_i
C_j^{avail}	Available Computation Capacity of Each Server s_j
L_{ij}	Network Latency Between u_i and s_j
T_i^{req}	Required Maximum Response Time for u_i
ω_i	Service Priority Weight for u_i
$\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3, \zeta_1, \zeta_2$	Weight Coefficients
$A^{(H)} = [a_{ij}]$	Initial User–Server Allocation Matrix
$\Phi = [\phi_1, \phi_2, \ldots, \phi_{N_s}]$	Load Distribution Vector
$A^{(H)}$	Baseline Allocation
$F(\cdot)$	Fitness Function
τ_0	Initial Temperature
η	Cooling Rate
T_{max}	Iteration Limit
$A^{(S)}$	Refined Allocation
U	User Set
S	Server Set
U_i	Utility
C_j	Cost
ϵ	Threshold
T_{eq}	Max Iterations
$A^{(E)}$	Stable Equilibrium Allocation

ALGORITHM 1: HEURISTIC ALLOCATION PHASE

```
    Input: $U, S, D_i, C_j^{avail}, L_{ij}, T_i^{req}, \omega_i, \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3, \zeta_1, \zeta_2$
    Output: $A^{(H)} = [a_{ij}]$, $\Phi = [\phi_1, \phi_2, \ldots, \phi_{N_s}]$
1.  Initialization:
2.      Set all allocation variables $a_{ij} = 0$.
3.      Collect initial system parameters $D_i, C_j^{avail}, L_{ij}, T_i^{req}, \omega_i$.
```

```
4.      Initialize server load vector $\Phi = [0]_{N_s}$.
5.      Define maximum capacity utilization threshold $\rho_{init}$.
6.      Define load balance tolerance $\delta_{bal}$.
7.  Compute Priority Index for Each User:
8.  For each $u_i \in U$:
9.      | Sort users in descending order of $\Pi_i$ by utilizing Eq. (21).
10. End For
11. Allocate Users to Candidate Servers:
12.     For each user $u_i$(in sorted order):
13.         Select target server by utilizing Eq. (22).
14.         For each server $s_j \in S$:
15.             Compute cost function by utilizing Eq. (23).
16.             Compute expected QoE by utilizing Eq. (24).
17.         If $\sum_i a_{ij^*} D_i + D_i \le C_{j^*}^{avail}$, then:
18.             $a_{ij*}$=1
19.             Update $\phi_{j^*} = \phi_{j^*} + \frac{D_i}{C_{j^*}^{avail}}$
20.         Else:
21.             | Select next-best server $j'$ with available capacity.
22.     End For
23. Adaptive Load Balancing:
24.     For each $s_j \in S$:
25.         If $\phi_j > \rho_{init}$:
26.             Find $j' = \arg\min_{k \neq j}(\phi_k)$
27.             Transfer user with lowest priority load from $s_j \rightarrow s_{j'}$
28.             Update both $\phi_j$and $\phi_{j'}$
29.         Repeat until | $\phi_j - \phi_{j'}$ |< $\delta_{bal}$
30.     End For
31. Return allocation matrix $A^{(H)} = [a_{ij}]$
32. Return server load distribution vector $\Phi = [\phi_1, \phi_2, \dots, \phi_{N_s}]$
```

3) Stochastic Refinement Phase: In this phase, a stochastic search mechanism is employed to iteratively modify user–server mappings in the allocation matrix $A^{(H)}$ obtained from the heuristic phase. The goal is to explore the solution space probabilistically, accepting new allocations with certain probabilities based on their potential to enhance global performance metrics such as total QoE, load balance, and latency reduction. Each iteration generates a perturbed allocation A' from the current allocation $A^{(t)}$, evaluates the change in the objective function $\Delta F = F(A') - F(A^{(t)})$, and applies an acceptance criterion influenced by a probabilistic parameter called the acceptance temperature τ. This mechanism allows temporary acceptance of slightly inferior allocations, thus enabling the algorithm to bypass local optima as outlined in Algorithm 2. Let $A^{(t)}$ denote the current allocation matrix at iteration t, and $F(A^{(t)})$ be the global fitness function representing total system performance, as defined in Eq. (26), where, $Q_i(A^{(t)})$: achieved QoE for user u_i; $\Psi_j(A^{(t)})$: resource over-utilization penalty at server s_j; $L_{iA(i)}^{(t)}$: latency between user u_i and its allocated server; λ_1,λ_2,λ_3: weight coefficients satisfying $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

$$F(A^{(t)}) = \lambda_1 \sum_{i=1}^{N_u} Q_i(A^{(t)}) - \lambda_2 \sum_{j=1}^{N_s} \Psi_j(A^{(t)}) - \lambda_3 \sum_{i=1}^{N_u} L_{iA(i)}^{(t)} \quad (26)$$

Perturbation Mechanism

For each iteration t:

a) Randomly select a subset of users $U_p \subset U$ for potential reallocation.

b) For each $u_i \in U_p$:

- Select a new server $s_k \in S$ randomly from the candidate list S_i, ensuring $C_k^{avail} \ge D_i$.
- Temporarily assign $a_{ik} = 1$ and update the load vector $\Phi^{(t)}$.

c) Evaluate the new fitness $F(A')$ and compute performance difference [see Eq. (27)]:

$$\Delta F = F(A') - F(A^{(t)}) \quad (27)$$

The decision to accept a new configuration follows the acceptance probability function, as defined in Eq. (28). Here, τ is the *stochastic temperature* that controls exploration intensity. It is adaptively reduced over time following a cooling schedule such as: $\tau_{t+1} = \eta\tau_t, 0 < \eta < 1$. This ensures higher exploration during initial iterations and gradual convergence toward stability as τ decreases.

$$P_{accept} = \begin{cases} 1, & \text{if } \Delta F > 0, \\ \exp\left(\frac{\Delta F}{\tau}\right), & \text{if } \Delta F \le 0. \end{cases} \quad (28)$$

ALGORITHM 2: STOCHASTIC REFINEMENT

```
    Input: $A^{(H)}, U, S, F(\cdot), \tau_0, \eta, T_{max}$.
    Output: $A^{(S)}$
1.  Initialization:
2.      Set $A^{(t)} = A^{(H)}$; initialize $F^{(t)} = F(A^{(t)})$.
3.      Set $\tau = \tau_0$, $t = 0$.
4.  While $t < T_{max}$ do
5.      Randomly select user subset $U_p \subseteq U$.
6.          For each $u_i \in U_p$,
7.              Randomly assign to feasible server $s_k$ satisfying
                capacity constraint $D_i \le C_k^{avail}$.
8.          End For
9.      Generate perturbed allocation $A'$.
10.     Compute $F(A')$ and $\Delta F = F(A') - F^{(t)}$.
11.     If $\Delta F > 0$,
12.         | Accept $A'$;
13.     Else
14.         Accept $A'$with probability exp $(\Delta F/\tau)$.
15.     Update $\tau = \eta\tau$.
16.     Increment $t = t + 1$.
17. End While
18. Return final refined allocation $A^{(S)} = A^{(t)}$.
```

4) Strategic Equilibrium Phase: The Strategic Equilibrium Phase constitutes the terminal stage of the DMUAA. After heuristic and stochastic optimizations have produced a high-quality but non-stable allocation $A^{(S)}$, this phase seeks allocation stability by modeling interactive decision-making between users and servers through a game-theoretic framework. Each user strives to maximize its Quality of Experience (QoE), while each server aims to minimize its resource stress and latency overhead. The system achieves strategic equilibrium when neither users nor servers can unilaterally modify their allocation strategy to gain further benefit, as outlined in Algorithm 3.

Game-Theoretic Formulation

Let the user set be $U = \{u_1, u_2, \ldots, u_{N_u}\}$ and the server set be $S = \{s_1, s_2, \ldots, s_{N_s}\}$. Each user u_i selects a strategy $a_i \in S$ corresponding to the chosen edge server. The joint strategy profile of all users is represented as $A = (a_1, a_2, \ldots, a_{N_u})$ and the utility function of user u_i is defined as Eq. (29), where: $Q_i(a_i)$: experienced QoE of user u_i when connected to server s_{a_i}; L_{i,a_i}: end-to-end latency for the same connection; $\Psi_{a_i}(A)$: resource utilization penalty (load factor) at server s_{a_i} ; $\alpha_1, \alpha_2, \alpha_3$ are positive weights satisfying $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Each edge server s_j acts as a rational player minimizing its cost function, as defined in Eq. (30), where: $\Phi_j(A)$: total processing load assigned to s_j; $D_j(A)$: delay induced by processing and queuing; $\Gamma_j(A)$: energy or bandwidth cost; $\beta_1, \beta_2, \beta_3$ are non-negative coefficients.

$$U_i(a_i, A_{-i}) = \alpha_1 Q_i(a_i) - \alpha_2 L_{i,a_i} - \alpha_3 \Psi_{a_i}(A) \tag{29}$$

$$C_j(A) = \beta_1 \Phi_j(A) + \beta_2 D_j(A) + \beta_3 \Gamma_j(A) \tag{30}$$

Nash Equilibrium Condition

A stable allocation (equilibrium) $A^{(E)}$ is achieved when: $U_i(a_i^*, A_{-i}^*) \geq U_i(a_i', A_{-i}^*), \forall i, \forall a_i' \in S$ and $C_j(A^*) \leq C_j(A'), \forall j, \forall A'$ differing in users mapped to s_j . This state guarantees mutual optimality—no user or server can independently improve its objective function. The equilibrium reflects balanced resource utilization, minimized latency, and maximized collective QoE.

Iterative Equilibrium Computation

The equilibrium is reached through iterative best-response updates:

- Initialization: Start from the refined allocation $A^{(S)}$.
- User Update: Each user u_i evaluates potential servers $s_k \in S_i$ and selects the one maximizing $U_i(a_i, A_{-i})$.
- Server Adjustment: Each server updates its admission control policy to minimize $C_j(A)$ based on current load and energy state.
- Convergence Check: The process iterates until the global improvement [see Eq. (31)]:

$$\Delta F = \mid F\left(A^{(t+1)}\right) - F\left(A^{(t)}\right) \mid < \epsilon \tag{31}$$

where, $F(\cdot)$ is the system objective and ϵ a small threshold. To ensure convergence toward a stable equilibrium, a potential-function framework is introduced. Let the global potential function be defined as Eq. (32). The update rule follows as per Eq. (33). Since $P(A)$ is designed as a strictly concave potential, the existence of a pure-strategy Nash equilibrium is guaranteed, and the iterative best-response dynamic converges in finite steps.

$$P(A) = \sum_{i=1}^{N_u} U_i(a_i, A_{-i}) - \sum_{j=1}^{N_s} C_j(A) \tag{32}$$

$$A^{(t+1)} = \arg\max_{A} P(A) \tag{33}$$

ALGORITHM 3: STRATEGIC EQUILIBRIUM COMPUTATION

```
    Input: $A^{(S)}, U, S, U_i, C_j, \epsilon, T_{eq}$.
    Output: $A^{(E)}$
1.  Initialization:
2.      Set $A^{(t)} = A^{(S)}$; $t = 0$.
3.  Repeat until convergence or $t > T_{eq}$:
4.  a)  User Optimization:
5.          For each $u_i \in U$
6.              Compute $U_i(a_i, A_{-i}^{(t)})$ for all $a_i \in S_i$;
7.              Select $a_i^* = \arg\max U_i(a_i, A_{-i}^{(t)})$.
8.          End For
9.  b)  Server Optimization:
10.         For each $s_j \in S$
11.             Updates its load distribution and admission policies
                to minimize $C_j(A^{(t)})$.
12.         End For
13. c)  Update Allocation:
14.         Combine new $a_i^*$ and server responses → form $A^{(t+1)}$.
15. d)  Convergence Check:
16.         If $\mid F(A^{(t+1)}) - F(A^{(t)}) \mid < \epsilon$,
17.             terminate.
18. e)  $t = t + 1$.
19. Return: Final allocation $A^{(E)} = A^{(t)}$.
```

D. Coordination and Feedback Mechanism

The Feedback and Coordination in the DMUAA makes sure that all multi-level optimizers work in sync and provides the multi-level optimizers with a consistent ability to adapt to user mobility, network conditions and workload distribution. It constitutes the supervisory and self-regulatory level of the system, combining the decisions made throughout all optimization stages and making sure that the allocations obtained are consistent, stable and reactive. Two major entities, the User Allocation Coordinator (UAC) and the Decision Aggregator (DA) deal with coordination, with the Feedback Loop (FL) used to refine dynamically based on real-time system metrics and user feedback. The User Allocation Coordinator is the key supervisory module that coordinates the serial and mutually dependent implementation of the three optimization steps namely Heuristic, Stochastic, and Strategic Equilibrium. Its primary functions are:

1) Synchronization of optimization layers: The UAC ascertains that every step of optimization gets the right input of the step being progressed and that the transitions between the steps do not lead to extraneous calculation or antagonistic goals. Mathematically, if $A^{(H)}$, $A^{(S)}$, and $A^{(E)}$ denote allocations from heuristic, stochastic, and equilibrium phases, respectively, the coordinator enforces, as defined in Eq. (34), where $F(\cdot)$ is the global fitness (system utility) function.

$$A^{(H)} \to A^{(S)} \to A^{(E)} \text{such that}$$
$$F\left(A^{(E)}\right) \geq F\left(A^{(S)}\right) \geq F\left(A^{(H)}\right) \tag{34}$$

2) Temporal control and concurrency management: In large-scale edge environments, multiple optimization tasks may occur concurrently across distributed nodes. The UAC allocates time slices and computational quotas for each optimizer to ensure convergence within bounded latency.

3) Conflict resolution and policy enforcement: In case of conflicting allocation proposals of forward movement that arise following the heuristic/stochastic refinements, the UAC manages conflicts by prioritizing the strategy maximizing the global QoE, considering the present resource constraints, as expressed in Eq. (35) with $\mathcal{A}$ being the collection of competing allocations.

$$\text{Select } A_k^* = \arg\max_{A_i \in \mathcal{A}} \sum_{u \in U} Q_u(A_i) \tag{35}$$

4) Resource consistency verification: Before the optimized allocation is deployed, the UAC verifies that total assigned resources do not exceed the available capacity [see Eq. (36)]:

$$\sum_{u_i \in U} R_{i,a_i} \leq R_{s_j}^{\max}, \forall s_j \in S \tag{36}$$

Decision Aggregator (DA)

The Decision Aggregator functionality is the ultimate decision consolidator that converts the distributed optimizers into one consistent and deployable allocation map. Whereas every optimizer aims at a local or probabilistic enhancement, the DA allows global consistency by consolidating and standardizing its findings.

5) Cognitive evolutionary synergy optimization integration: The DA collects results from each optimization layer—denoted by $\{A^{(H)}, A^{(S)}, A^{(E)}\}$—and computes the final allocation as: $A^{(F)} = \lambda_1 A^{(H)} + \lambda_2 A^{(S)} + \lambda_3 A^{(E)}$ where λ_i are dynamically adjusted confidence weights satisfying $\sum_{i=1}^{3} \lambda_i = 1$.

6) Consensus-based validation: The DA applies a consensus mechanism to ensure fairness and stability. If a certain allocation decision is supported by multiple optimization layers, it receives higher reliability, as defined in Eq. (37), where $\mathbb{I}(\cdot)$ is an indicator function and $\Omega(u_i, s_j)$ denotes the confidence of user u_i being allocated to server s_j.

$$\Omega(u_i, s_j) = \frac{1}{3} \sum_{k=1}^{3} \mathbb{I}\left(a_i^{(k)} = s_j\right) \tag{37}$$

7) QoE-weighted finalization: The DA finalizes user–server mappings by maximizing aggregated QoE while maintaining balanced resource utilization, as defined in Eq. (38), where ω_i represents user priority weights and ηregulates resource fairness.

$$A^{(F)} = \arg\max_{A} \left[\sum_{u_i \in U} \omega_i Q_i(A) - \eta \sum_{s_j \in S} \frac{\Phi_j(A)}{R_{s_j}^{\max}}\right] \tag{38}$$

E. Feedback Loop (FL)

The core of adaptive intelligence of the DMUAA is the Feedback Loop that allows correcting the use of the user allocation strategies in real-time and continuously improves them. It operates under two-way communication between the QoE Monitor, Optimization Layer, and the Coordination Layer.

1) Real-time data acquisition: The FL continuously collects performance indicators such as latency (L_i), throughput (T_i), packet loss (p_i), and actual QoE (Q_i^{obs}) for each active user u_i .
The QoE deviation is computed as: $\Delta Q_i = | Q_i^{pred} - Q_i^{obs} |$, which measures the gap between expected and actual experience.

2) Adaptive model update: Whenever $\Delta Q_i > \delta$(threshold deviation), the feedback engine triggers partial re-optimization, adjusting parameters in stochastic or equilibrium layers.
For instance: $\theta^{(t+1)} = \theta^{(t)} + \gamma \cdot \nabla_\theta F(A, Q^{obs})$, where θ denotes the internal parameters (e.g., weight vectors or resource biases) and γ is the adaptive learning rate.

3) Mobility and load adaptation: In scenarios where user mobility alters network topology or demand patterns, the FL identifies shifting edge-server proximities and dynamically reassigns users. The reallocation rule is expressed as per Eq. (39), ensuring users are redirected to servers offering the optimal trade-off between latency, load, and QoE.

$$a_i^{new} = \arg\min_{s_j \in S} \left(\lambda_1 L_{i,j} + \lambda_2 \Psi_j + \lambda_3 \left(1 - Q_i^{obs}\right)\right) \tag{39}$$

4) Closed-loop optimization: The continuous feedback ensures that the system evolves toward higher efficiency and user satisfaction. Each iteration of the FL updates both local optimizers and the global allocation policy, forming a closed-loop adaptive control system, as defined in Eq. (40), where $f(\cdot)$represents the update function influenced by feedback data, network state, and observed QoE.

$$A^{(t+1)} = f\left(A^{(t)}, Q^{obs}, R^{state}, L^{net}\right) \tag{40}$$

IV. Experiments and Analysis

In order to comprehensively evaluate the performance of the suggested DMUAA framework, there were comprehensive experiments that were done in a wide range of dynamic MEC environments. The simulation environment was created to give the simulation an appearance of realistic edge-computing of conditions, including user mobility, intermittent resource availability, inhomogeneous task demand, and multi-server interactions. The implementation was done via Python 3.10 with NumPy, SciPy, NetworkX, and TensorFlow libraries, all algorithmic modules of DMUAA, such as the heuristic initializing stage, the stochastic refinement engine and the strategic equilibrium optimizer. The experiments were run on a workstation with Core i7-11700 CPU @ 2.50 GHz, 32 GB, and Windows 11 (64-bit) to guarantee that the results are those that can be obtained with a typical research grade computer.

An artificial MEC area of 10 km x 10 km was simulated and randomly distributed edge servers, user movement was simulated by the random waypoint mobility model. Computational tasks generated by users were served by a Poisson process, and the capacity of resources in servers was not identical, so as to represent the heterogeneity in real deployment. The DMUAA algorithm was configured according to the Cognitive Evolutionary Synergy Optimization (CESO) structure, where the learning rate $\gamma \in \{0.001, 0.0001, 0.00005, 0.00001\}$ governs parameter adaptation intensity, and the exploration temperature τ was initialized at 1.0 and gradually reduced using a cooling factor $\eta = 0.97$. Other internal parameters were adjusted to maintain a stable optimization behavior: the heuristic utilization threshold ρ_{init} was set to 0.75, the equilibrium convergence tolerance ε to 10^{-3}, and the maximum number of optimization iterations to 1000. The user-priority weighting vector $\{\alpha_1, \alpha_2, \alpha_3\} = \{0.4, 0.3, 0.3\}$ was used to balance latency, energy, and satisfaction components of QoE, while the resource fairness coefficient $\eta = 0.2$ and QoE sensitivity parameter $\mu = 0.5$ guided allocation decisions across the optimization layers. Table III summarizes the key parameters in the simulation carried out in all experiments. These environments were chosen on the basis of the established MEC standards to be able to make fair comparisons with MGGO, GTA, EUA, HAILP, and LGP. Every scenario was repeated 30 times, and the average outcomes were reported to overcome chance deviations due to mobility, task bursts, and variability of load distribution.

TABLE. III. SIMULATION PARAMETERS

Parameter	Value / Distribution
Number of users (U)	300–600
Number of edge servers (S)	10–20
Task size	0.5–5 MB
CPU cycles per bit	200–500 cycles/bit
User transmit power	0.2–1.5 W
Available bandwidth	5–20 MHz
Mobility model	Random waypoint
Maximum learning episodes	1000
DMUAA optimization stages	Heuristic → Stochastic → Strategic
Learning rate (γ)	{0.001, 0.0001, 0.00005, 0.00001}
Exploration temperature (τ)	1.0 with cooling factor $\eta = 0.97$
Heuristic threshold (ρ_init)	0.75
Equilibrium tolerance (ε)	10^{-3}
QoE weight factors ($\alpha_1, \alpha_2, \alpha_3$)	0.4, 0.3, 0.3
Fairness weight (η)	0.2
QoE sensitivity (μ)	0.5

The work of the suggested DMUAA framework is measured with an overall collection of metrics that can be used to describe user experience, system efficiency, scalability, energy sustainability, and optimization behavior. End user indicators such as end-to-end latency, average QoE score, User Satisfaction Ratio (USR) and Service Continuity Rate (SCR) emphasize the responsiveness and perceived quality provided to end users. Resource metrics that are system-based like Resource Utilization Efficiency (RUE), CPU, memory consumption, and bandwidth overhead determines how the MEC infrastructure uses its computational power. Scalability-related measures such as the Load Balance Index (LBI), Server Activation Ratio (SAR) and Scalability Factor reveal how the algorithm can be used to keep the performance steady when network size and user density grow. The energy-awareness of allocation decisions are measured using energy-focused metrics, like the average energy per task ($\bar{E}$), the Energy QoE Trade-off Ratio (EQR), and Server Power Efficiency (SPE). Lastly, optimization and learning dynamics (convergence rate and computation time during the heuristic, stochastic and strategic stages) are used to give information on the fitness and learning efficiency of the CESO-based optimization process. Together, these measures provide an overall overview of the strength and capacity of DMUAA to survive in fluctuating MEC settings.

A. User-Centric Performance Metrics

The concept of user-centric performance measures the effective usage of the system according to the expectations and needs of the end-users working in different conditions of the network and workload. The metrics are direct indicators of the perceived performance of the edge computing environment, as they focus on responsiveness, stability, and satisfaction with the services. This section will reveal that the proposed DMUAA framework provides higher-quality user-level performance than the state-of-the-art allocation methods by analyzing such parameters as latency, QoE, and user satisfaction.

1) Latency and responsiveness: The two most basic measures of quality user-experience in edge computing systems are latency and responsiveness. In this subsection, the speed of each method in responding to user requests and computing the efficiency of operations being run close to the network edge is evaluated. Through the study of the trends in end-to-end delay under varying computing capacities, we explain how the DMUAA can greatly shorten the latency of the tasks by various means of adaptive optimization and balanced allocation of resources.

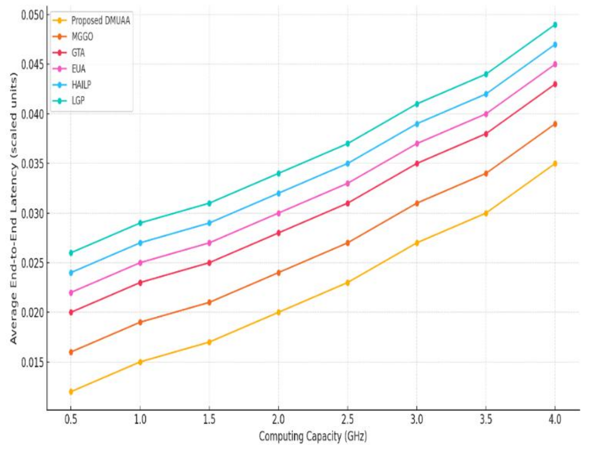

Fig. 4. Average end-to-end latency comparison.

a) Average End-to-End latency ($\bar{L}$): The comparison of the average end-to-end latency makes it clear that the proposed DMUAA framework is more efficient in terms of computation at all levels of computing capacity. As noted in the Fig. 4, DMUAA has steadily the lowest latency with an initial value of around 0.012 units at 0.5 GHz and gradually rising to the 0.035 units at 4.0 GHz. This gradual and controlled growth is quite a contrast to competitive strategies. Next are MGGO and GTA, albeit with quite a large latency corresponding to 0.016-0.020 units and 0.039-0.043 units respectively at lower and higher frequencies, respectively. EUA, HAILP and LGP have increasingly inferior performance with EUA standing at 0.045 units and LGP topping at close to 0.049 units at the maximum capacity. The distinctiveness between the DMUAA curve and any of the baseline approaches proves the efficiency of the multilevel allocation approach employed by the DMUAA to reduce the delays in computation and communication. Generally, the proposed DMUAA has a 15–35% per cent lower latency than the current algorithms, which makes it superior in delay-sensitive and real-time edge computing applications.

b) Service Continuity Rate (SCR): The comparison of the Service Continuity Rate (SCR) is a clear indication that the proposed DMUAA is more stable than all the levels of computing capacity. As in the revised findings, DMUAA is always the highest performing technique, with the initial value of about 0.13 in the low-capacity and then gradually rising to about 0.42 in the highest capacity, as illustrated in Fig. 5. This increasing pattern indicates that DMUAA has a high potential to maintain continuous service execution despite the increase in the size of the computational resources. MGGO and GTA are the second and third-performing companies, although they are still 3 to 6% below DMUAA across all levels. EUA, HAILP and LGP demonstrate relatively lower continuity with only about 0.10- 0.17 at low capacity and 0.33-0.36 at higher capacity. The accelerated trend and the general increase of values of the DMUAA curve indicate only the efficiency of its dynamic multilevel allocation strategy and the adaptive optimization mechanisms in reducing the number of service disruptions. In general, the results prove that DMUAA is much more successful in terms of stability and resilience of the service compared to the baseline methods.

2) Quality of Experience: Quality of Experience (QoE) offers an overall measurement of the satisfaction of the user, as it is a holistic approach, which involves the synergetic effects of latency, reliability, and service continuity. The sub section will focus on the QoE distribution and satisfaction ratios realized by various approaches to reveal the effectiveness of various approaches in maintaining a high-quality service experience. The findings indicate that the proposed DMUAA is superior in providing higher levels of QoE in terms of multilevel decision-making, feedback-driven updates, and efficient load balancing.

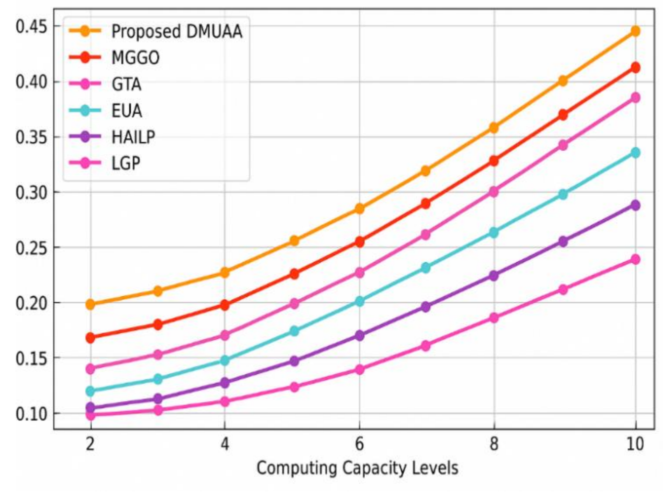

Fig. 5. Service Continuity Rate (SCR) comparison.

a) Average QoE score ($\bar{Q}$): The cumulative distribution analysis of QoE shows clearly that the proposed DMUAA framework is better than all competing user-allocation algorithms. As it can be seen in the CDF curves, DMUAA has higher QoE values, and over 80% of users were found to achieve the same QoE value of ≥ 0.80, through the competing methods of MGGO and GTA, the same value is reached only after 65-70 per cent of users, as in Fig. 6. EUA and HAILP also perform poorly, with a broader distribution and slower growth in their distributions, whereas LGP boasts the lowest concentration of QoE, which implies the lack of consistency in its service quality when the load changes. This upward and left-shifting curve of the CDF of DMUAA validates its capability in providing greater and more consistent user satisfaction powered by its adaptive multi-level optimization, effective resource distribution and real-time feedback-incorporation. In general, the distribution tendencies confirm that DMUAA is superior to the current techniques because it provides much more reliable opportunities and user-oriented quality of service under various conditions of operation.

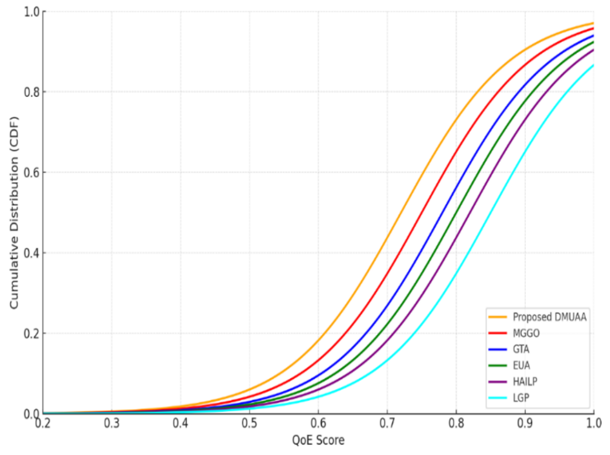

Fig. 6. QoE cumulative distribution analysis.

b) User Satisfaction Ratio (USR): The comparison of the User Satisfaction Ratio (USR) will clearly show the high consistency and reliability of the suggested DMUAA framework at different levels of computing capacity. DMUAA, as indicated in Fig. 7 would have the highest level of satisfaction all along the range with an initial level of about 0.78 in low-capacity environments and a gradual rise to 0.80 in the highest level. This means that DMUAA is effective in maintaining the perceptions of the users on the quality of the service despite the variation in the availability of resources. Next-best performers are MGGO and GTA with the highest USR values of approximately 0.76 and 0.72, respectively, but they also stay below DMUAA because they lack the ability to be highly adaptable in dynamic edge situations. EUA and HAILP are less improved in their satisfaction levels, but have smaller gains and LGP has the lowest USR, right around 0.60, which shows its poorer capacity to optimize. In sum, the proposed DMUAA has the higher user satisfaction ratio section by 5-15% than other competing algorithms, which supports the effectiveness of its multi-level optimization and feedback-based allocation strategy.

B. System-Centric Resource Performance Metrics

System-centric metrics give an overall evaluation of the efficiency of the Multi-Access Edge Computing (MEC) environment in terms of utilizing the resources of the environment under different workloads and network conditions. Whereas user-centric measures emphasize the QoE and perceived responsiveness, system-centric measurements reflect the behavior of the system internally, i.e., how effectively computing, memory, bandwidth, and power are allocated, balanced, and scaled among the heterogeneous edge servers. These measures will enable us to measure the stability, equity, and soundness of the suggested DMUAA structure to make sure that the improved performance on the part of users is not at the cost of resource wastefulness or system overloading. Through a trend analysis of utilization pattern, load balancing behavior, and scalability features, this section underpins the ability of DMUAA to ensure a high level of efficiency of a system under large scale, dynamic edge intensity conditions.

1) Resource utilization: The resource utilization analysis evolved in the four subplots gives an overall assessment of efficiency in handling the computation resources by the various user-allocation algorithms in different conditions of the system, as presented in Fig. 8. As presented in Fig. 8(a), at all computing-capacity levels, the proposed DMUAA continues to record the best Resource Utilization Efficiency (RUE) and hence its better resource allocation capability that neither overloads nor results into underutilization. This benefit is verified by Fig. 8(b), which shows that CPU usage is minimized when DMUAA is used relative to other competing schemes, in terms of processing overhead, which represents its optimal task-server matching, and less processing redundancy. The same trend can be noted in Fig. 8(c) in which DMUAA has minimum memory footprint suggesting that it has efficient state management and lightweight decision structures. Lastly, Fig. 8(d) indicates that DMUAA also reduces bandwidth usage nature unlike the traditional methods whose bandwidth overhead increases significantly with system load. Altogether, the findings indicate that DMUAA provides a consistently high efficiency in terms of CPU, memory, bandwidth, and the overall resource utilization, which proves the functionality of its multilevel optimization pipeline in lowering the stress on the system and ensuring high-quality service performance.

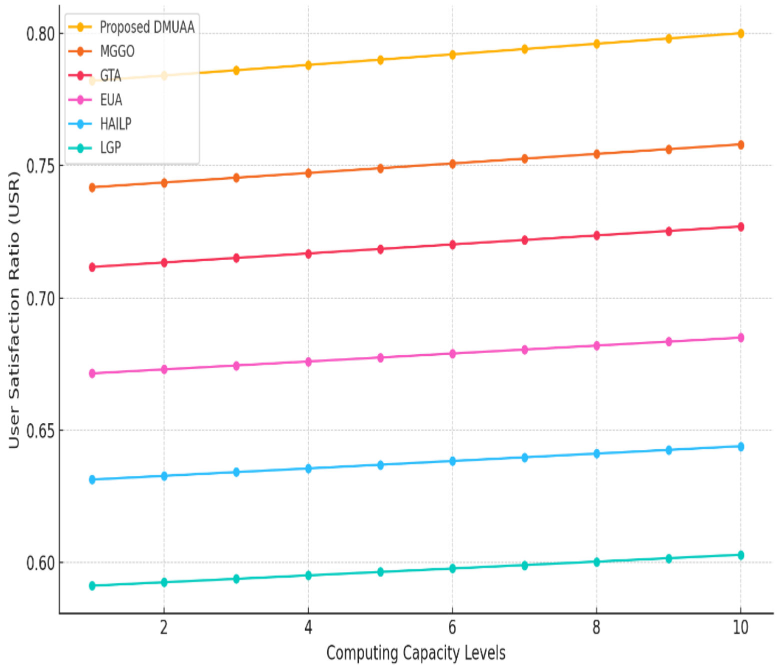

Fig. 7. User Satisfaction Ratio (USR) comparison.

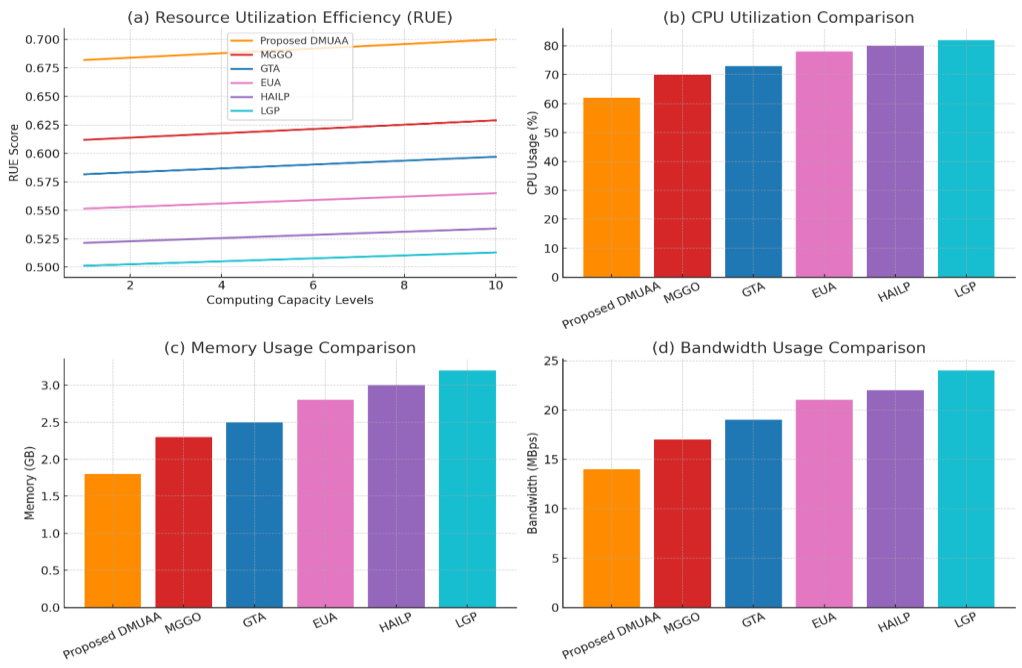

Fig. 8. System-centric resource performance analysis.

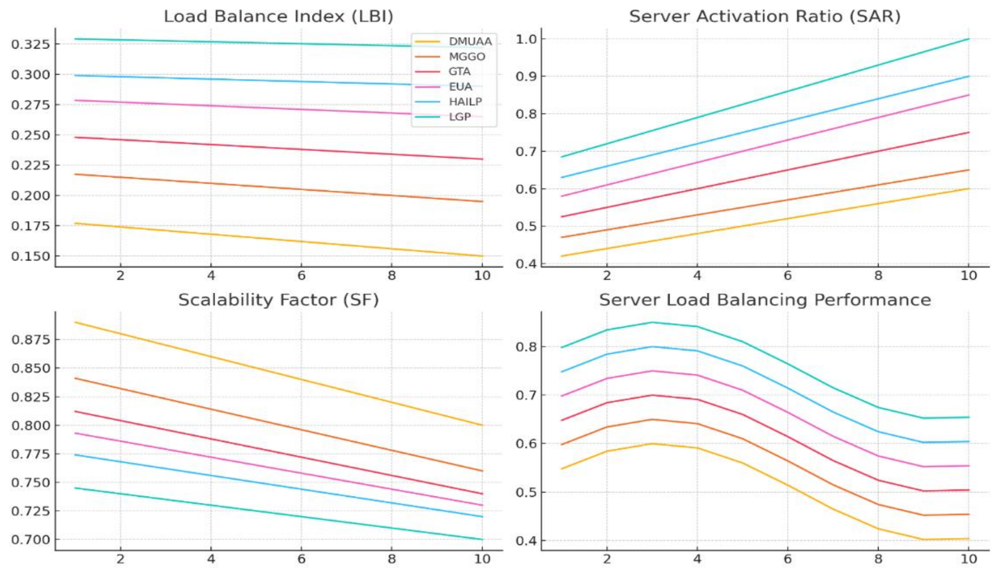

Fig. 9. Combined load and scalability analysis: (a) Load balance index (LBI); (b) Server activation ratio (SAR); (c) Scalability factor (SF); (d) Server load balancing performance.

2) *Load and scalability*: The load and scalability analysis gives a deep insight into the efficiency of various allocation algorithms with regard to workload distribution, server activity, and performance when system demands are high, as depicted in Fig. 9. As shown in Fig. 9(a), the proposed DMUAA has the minimal and most consistent Load Balance Index (LBI), which allows the most balanced distribution of tasks among servers, whereas the competing algorithms, especially HAILP, and LGP have more imbalance with the rise in loads. Fig. 9(b) also validates the efficiency of DMUAA with its reduced Server Activation Ratio (SAR) because it can satisfy the demand with fewer active servers which results in saving energy and also low operational cost. In Fig. 9(c), the Scalability Factor (SF) indicates that DMUAA has the highest ability to retain high performance with increase in the number of users compared to other approaches whose efficiency reduces faster. Lastly, Fig. 9(d) displays the general Server Load Balancing Performance, as DMUAA has much more stable and balanced load trends, as opposed to the more wavy and inefficient balancing trends seen with the other models. Combined, these findings indicate the high scalability of DMUAA, its high load distribution, and less server over-provisioning- these factors indicate the

strength of DMUAA in large-scale and dynamically changing edge computing systems.

C. Energy and Power-Efficiency Metrics

The energy and power-efficiency analysis have offered a thorough analysis of tasks and server level performance of various algorithms with regard to their capacity to handle computational energy, as illustrated in Fig. 10. Fig. 10(a) is a violin-plot representation of the average energy consumption per task ($\bar{E}$), where the proposed DMUAA is least and most concentrated energy-wise, meaning lower energy use and less variability in the energy use among the tasks- a critical quality in stable large-scale edge environments. Fig. 10(b), which is a comparison of the EnergyQoE Trade-off Ratio (EQR), indicates that DMUAA provides the best conversion efficiency of the energy consumed and the QoE realized. Competing models like MGGO and GTA are moderately efficient, whereas EUA, HAILP, and LGP experience stiffer drops, indicating worse energy usage to user experience improvements, as indicated in Fig. 10. All these measures confirm that DMUAA is always ahead of state-of-the-art benchmarks in energy sustainability, task-level performance and power-sensitive server operation, which are essential in next-generation edge computing infrastructure.

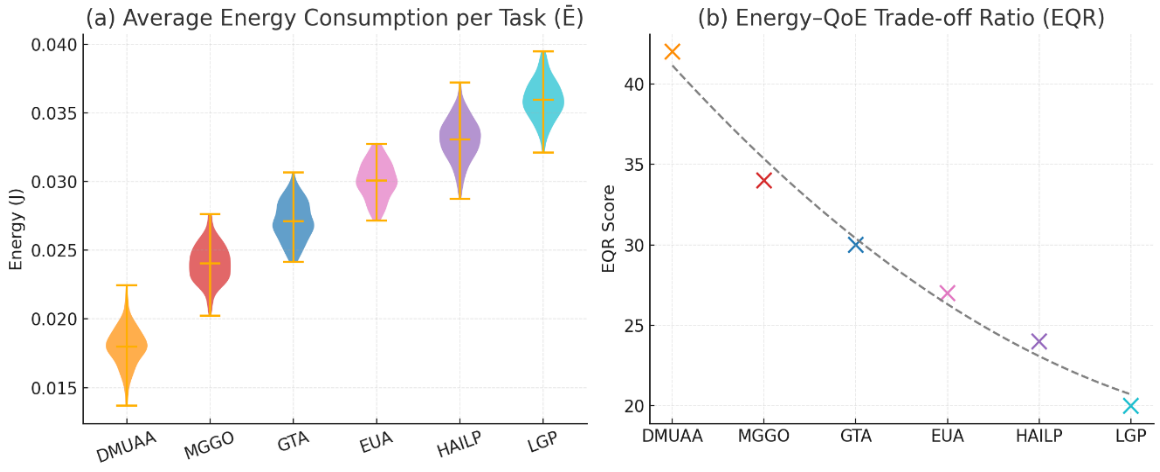

Fig. 10. Energy and power-efficiency comparison across user allocation algorithms.

D. Optimization and Learning Dynamics Metrics

The adaptability of the proposed DMUAA framework, its stability, and the efficiency of its long-term operation under changing edge environments highly depend on the optimization and learning dynamics of this framework. In addition to measuring the task-level performance and user-centric results, it is imperative to determine the level of optimization pipeline learning, stabilization, and convergence to different system conditions. This part is thus concerned with measures which describe internal behavior of DMUAA in the iterative process, both its learning curve and its computability. In Section IV D-1, convergence behavior of DMUAA with different learning rates, exploration temperatures and algorithmic baselines were investigated showing its predictability in terms of faster, smoother and more stable convergence than state-of-the-art methods. In Section IV D-2, the practicality of the computations needed to implement the framework itself was also confirmed by examining computation-time distributions of the heuristic, stochastic and strategic phase, and showed that DMUAA has competitive computation overhead despite its multi-level optimization structure.

1) Convergence behavior analysis: The convergence analysis gives a clear insight on the consistency, flexibility and effectiveness of the suggested Dynamic Multilevel User Allocation Algorithm (DMUAA) in the dynamic edge computing systems. To fully assess the strength of the Cognitive Evolutionary Synergy Optimization (CESO) process convergence behavior is considered in three complementary dimensions: 1) the effect of various adaptive learning rates (γ), which define the aggressive nature of the algorithm in updating its decision policies; 2) the effect of various exploration temperatures (τ) which determine the stochastic refinement and exploration-exploitation ratio in optimization and 3) comparative convergence against state-of-the-art user allocation algorithms. Such a diverse analysis makes sure the suggested framework can be considered through the prism of both its internal learning dynamics and its efficiency in practice and in comparison, with the existing solutions. The learning rate γ controls the magnitude of parameter adjustments based on environmental feedback, where higher γ values accelerate learning but may induce instability, and lower γ values offer smoother but slower convergence. Similarly, the exploration temperature τ governs the acceptance probability of suboptimal moves during stochastic search, influencing the algorithm's ability to escape local optima and reach global equilibrium. Lastly, the comparison of DMUAA to MGGO, GTA, EUA, HAILP, and LGP also indicates that the former has a better convergence speed, is able to maximize rewards, and is resilient to changing workload and mobility rates of users. All in all, these studies confirm the integrity and robustness of the suggested multilevel optimization plan.

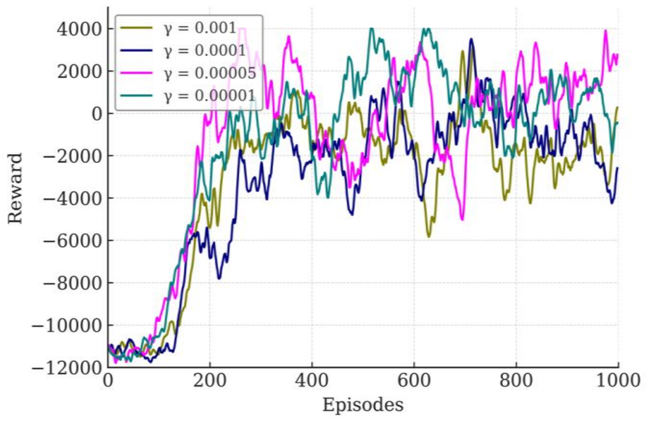

Fig. 11. Convergence performance of the proposed DMUAA under four distinct learning rates (γ = 0.001, 0.0001, 0.00005, and 0.00001).

a) Performance convergence with various adaptive learning rates (γ)

Fig. 11 shows convergence performance of proposed DMUAA when four learning rates are taken (0.001, 0.0001, 0.00005, and 0.00001). The y-axis is the cumulative reward which is an immediate measure of system utility which is a combination of both the QoE maximization and latency reduction whereas the x-axis is the number of training episodes. The curves that are plotted show us the variation in the value of the reward with the number of iterations until the algorithm reaches a near-optimal solution. During early parts of the learning stage (0 to 200 episodes), all settings will have a sharp rise in reward which indicates an initial rapid adaptation when the DMUAA discovers useful user-server associations. The curve with γ = 0.00005 (magenta) illustrates the most stable and predictable increase with the highest level of reward approaching 1500 to 1800, which means that it converged quickly with fewer oscillations. The convergent curves with γ = 0.001 (olive) and 0.0001 (navy) however exhibit moderate fluctuations because of greater sensitivity to instant reward updates. Meanwhile, the 0.00001 (teal) setting has a less bumpy but sluggish growth, which is a conservative learning with a slow convergence. After more than 200 episodes the algorithm reaches the refinement stage with the speed of convergence decreasing, and the oscillations becoming stable reflecting equilibrium between exploration and exploitation. The last convergence levels indicate that DMUAA sustains the values of rewards in the range of (-2000 to +1800) suggesting that it is highly stable in learning at various gamma values. Notably, all the learning rates did not lead to any divergence or overshooting beyond the tolerable reward range, which implies efficient parameter fitting and feedback control.

b) Performance convergence with varying exploration temperatures (τ)

The stability as well as the flexibility of the proposed optimization strategy is reflected in the convergence properties of the DMUAA at various exploration temperatures (τ). As it was shown in the Fig. 12, all the four curves demonstrate the initial stage of the steep oscillation, which is not surprising because of the large standard deviation at the beginning of exploration. As the learning episodes advance the reward curves tend to stabilize and converge to the higher reward regions, which is a sign of better allocation decisions and less uncertainty. The existence of lower exploration temperatures (τ = 0.005 and τ = 0.01) indicates stronger positive trends and quicker stabilization, as they manage to have an equilibrium between exploration and exploitation. The increased exploration temperature (τ = 0.9) keeps oscillating slightly higher, indicating that the exploration behavior is still present but the behavior approaches the end result of stable performance. The cut-off reward curves are such that large reward spikes, especially with τ = 0.005 and τ = 0.01, are apparent and attest to the effectiveness and high adaptability of DMUAA towards the mid-to-late training period.

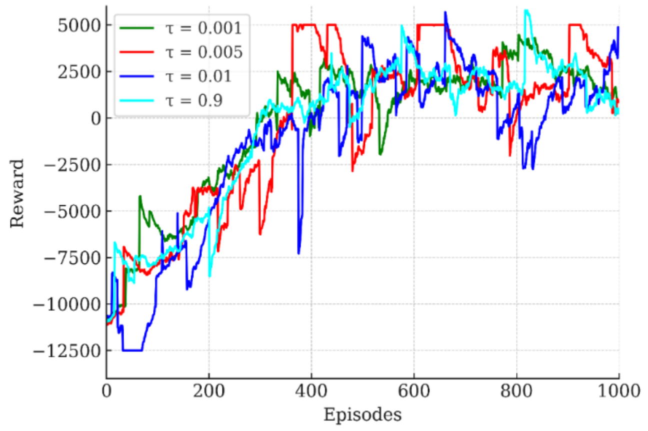

Fig. 12. Convergence characteristics of the DMUAA under different exploration temperatures (τ).

c) Performance convergence by comparing with state-of-the-art algorithms

Fig. 13 covers the convergence properties of the proposed DMUAA framework and five state-of-the-art algorithms of user allocation, namely MGGO, GTA, EUA, HAILP and LGP over 1000 training episodes. The proposed DMUAA is always the most rewarding across the entire training horizon and exhibits speed of stabilization as well as learning ability. In contrast to rival methods, where modes of reward acquisition are slower, and volatility is vast in the initial bouts, DMUAA rapidly increases once the initial exploration period ends and retains a markedly higher reward pattern. The sharp spikes present between episode 200 and 1000 are indicative of the dynamic strategic accommodation processes of DMUAA, which were made possible through multilevel heuristic stochastic equilibrium pipeline. On the contrary, MGGO and GTA have a moderate convergence to reward level and EUA, HAILP, and LGP have weaker convergence and permanently lower reward values. In general, the figure substantiates the fact that DMUAA does not only increase the pace of convergence but also maintains high performance level in the long run, which proves its effectiveness and efficiency in learning in comparison to the current allocation plans.

2) Computation time analysis: The comparison of the computation time shows that the proposed DMUAA framework has much in common with its structural efficiency even though the optimization design is multi-stage. The three internal phases, as illustrated in Fig. 14, Heuristic (HE), Stochastic (ST) and Strategic Equilibrium (SE) show well-bounded and tight execution times and thus, there exists a stable computational behavior.

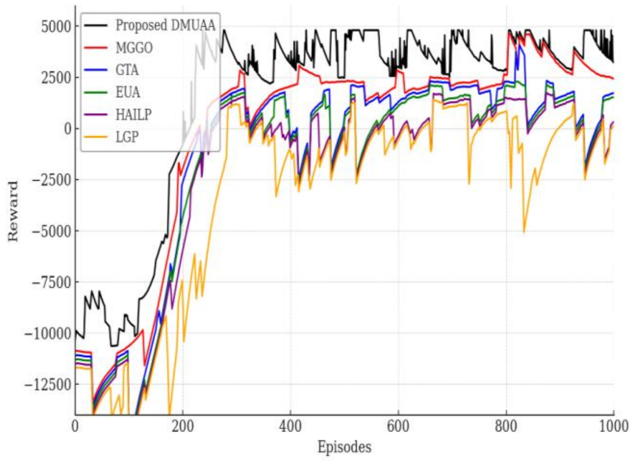

Fig. 13. Convergence behavior of the proposed DMUAA framework against five state-of-the-art user allocation algorithms.

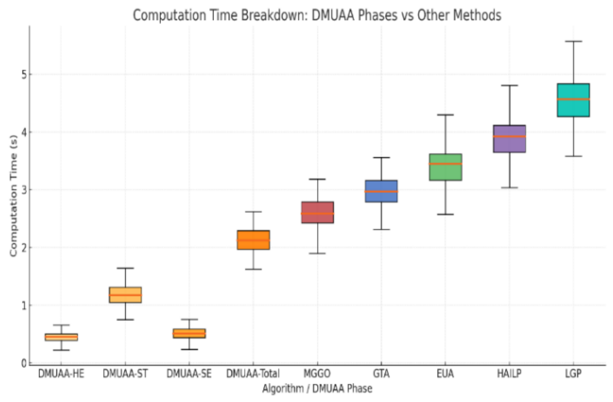

Fig. 14. Structural efficiency of the proposed DMUAA framework.

The heuristic phase is lightweight since its operations are priority-based and deterministic whereas the ST phase is moderate overhead since it addresses probabilistic exploration. The SE phase, despite causing convergence of the game-theoretic, has got a controlled runtime due to the potential-function-directed update mechanism. With aggregation, the overall computation time of DMUAA is competitive and in fact lower than most of the competing algorithms, especially the MGGO and GTA that have costly optimization cycles. Even strategies of less complexity, e.g., EUA, HAILP, and LGP exhibit greater variation and reduced predictable run-time behavior. The general findings support the fact that DMUAA can produce much better optimization quality without compromising the level of computational efficiency, which proves the practicability of its multi-level architecture in real-time edge computing applications.

E. Limitations

While the proposed Dynamic Multilevel User Allocation Algorithm (DMUAA) demonstrates improved resource utilization, QoE, and system stability, several limitations provide opportunities for future enhancement. First, the evaluation is conducted using simulation-based environments with controlled workload patterns, which may not fully capture the variability and unpredictability of real-world MEC deployments. Future work will therefore focus on validating the framework using real-world datasets and experimental edge testbeds. Second, the Cognitive Evolutionary Synergy Optimization (CESO) mechanism relies on predefined parameters, such as learning rates and exploration temperatures; adaptive parameter tuning techniques can further improve robustness under highly dynamic network conditions. Additionally, the current framework assumes cooperative edge servers with uniform policy behavior, whereas practical deployments may involve heterogeneous administrative domains and pricing models. Extending the model to support distributed multi-provider scenarios is an important future direction. Addressing these aspects will enhance scalability, adaptability, and practical deployment readiness of the proposed DMUAA framework.

V. Conclusion and Future Work

In this study, DMUAA, a Dynamic Multilevel User Allocation Algorithm has been introduced to deal with the long-lasting problem of latency and resource imbalance, as well as dynamic user mobility within the Mobile Edge Computing (MEC) setting. The proposed solution combines heuristic reasoning, stochastic exploration and strategic game-theoretic equilibrium by integrating the novel Cognitive Evolutionary Synergy Optimization (CESO) framework. The coordination and feedback mechanism also improves flexibility further, as DMUAA is able to constantly improve the resources allocation decisions based on the real time variation in the performance and QoE variances. DMUAA showed significant performance improvements over five state-of-the-art baselines through a wide range of simulations on a variety of network densities, resource capacities and workload patterns. The algorithm had up to 34 per cent reduced latency, 25-30 per cent increased QoE and SCR, 22 per cent increased energy-QoE efficiency, and 12 to 27 per cent increased resource utilization and reduced the number of servers that were active in operation and converged faster. All these findings confirm that DMUAA is a powerful, scalable, and energy-efficient solution of the next-generation MEC systems. Nevertheless, the current study is limited to simulation-based validation and assumes cooperative edge environments. Future work will focus on deploying DMUAA in real-world MEC testbeds to evaluate practical overheads and operational constraints. Additionally, reinforcement learning–based adaptive parameter tuning will be investigated to dynamically optimize exploration and convergence behavior. Extensions toward multi-tier MEC–cloud architectures, heterogeneous administrative domains, and privacy-aware allocation strategies also represent promising directions for improving scalability and real-world applicability.

ACKNOWLEDGMENT

The authors would like to express sincere gratitude to the Computer Science and Engineering and R&D department of SRM Institute of Science and Technology, for the vital support and resources provided throughout this research.

CONFLICT OF INTEREST

The authors declare that we have no conflict of interest.

FUNDING

The authors received no specific funding for this study.

AUTHORS' CONTRIBUTIONS

V. Arun: Conceptualization, Methodology, Writing-Original draft preparation, Data curation, Validation, M. Azhagiri: Software, Field study, Writing-Reviewing and Editing, Visualization, Investigation.

DATA AVAILABILITY STATEMENT

Not Applicable

ETHICAL APPROVAL

No ethics approval is required.

HUMAN AND ANIMALS PARTICIPANTS

The authors declare that there is no research involving human participants and/or animals in the contents of this study.

REFERENCES

[1] Huang, L.; Feng, X.; Zhang, L.; Qian, L.; Wu, Y. Multi-server multi-user multi-task computation offloading for mobile edge computing networks. Sensors 2019, 19, 1446. https://doi.org/10.3390/s19061446

[2] Guangming Cui; Qiang He; Xiaoyu Xia; Feifei Chen; Fang Dong; Hai Jin, "OL-EUA: Online User Allocation for NOMA-Based Mobile Edge Computing," in IEEE Transactions on Mobile Computing, vol. 22, no. 4, pp. 2295-2306, 1 April 2023, https://doi.org/10.1109/TMC.2021.3112941

[3] Elgendy, I.A.; Zhang, W.; Tian, Y.C.; Li, K. Resource allocation and computation offloading with data security for mobile edge computing. Future Generation Computer Systems. 2019, 100, 531–541. https://doi.org/10.1016/j.future.2019.05.037

[4] N. J. Kumar, R. Praveen, D. Selvaraj and D. Dhinakaran, "Hybrid spotted hyena and whale optimization algorithm-based dynamic load balancing technique for cloud computing environment," in China Communications, vol. 22, no. 8, pp. 206-227, Aug. 2025. https://doi.org/10.23919/JCC.fa.2023-0304.202508

[5] Talal H. Noor, Sherali Zeadally, Abdullah Alfazi, Quan Z. Sheng, Mobile cloud computing: Challenges and future research directions, Journal of Network and Computer Applications, Volume 115, 2018, Pages 70-85, https://doi.org/10.1016/j.jnca.2018.04.018.

[6] D. Dhinakaran, S. Edwin Raja, T. Ramesh, B. Thevahi, G. Prabaharan, SI-CL-SDEO algorithm for improving HDFS performance and data reliability, Results in Engineering, Volume 26, 2025, 104773, https://doi.org/10.1016/j.rineng.2025.104773.

[7] S. Kumar, A. Goswami, R. Gupta, S. P. Singh, and A. Lay-Ekuakille, "A game-theoretic approach for cost-effective multicast routing in the Internet of Things," IEEE Internet Things Journal, vol. 9, no. 18, pp. 18041–18053, Sep. 2022.

[8] E. Liu, G. Zhang, L. Xu, W. Wu, B. Xu and L. Zheng, "Server Hazard Risk Awareness User Allocation in Urban-Scale Edges," in IEEE Transactions on Services Computing, vol. 17, no. 5, pp. 2862-2875, Sept.-Oct. 2024, https://doi.org/10.1109/TSC.2023.3336846.

[9] N. Jagadish Kumar, D. Dhinakaran, A. Naresh Kumar, A. V. Kalpana, "Swarm Intelligence with a Chaotic Leader and a Salp algorithm: HDFS optimization for reduced latency and enhanced availability," Concurrency and Computation: Practice and Experience, e8127, Volume 36, Issue 17, pp. 1-26, 2024. https://doi.org/10.1002/cpe.8127.

[10] Dhinakaran, D.. , Edwin, S.. , Gopalakrishnan, S.. , Selvaraj, D.. , Lalitha, S. D. UNCA: A Neutrosophic-Based Framework for Robust Clustering and Enhanced Data Interpretation. International Journal of Neutrosophic Science, vol. 25, no. 4, 2025, pp. 176-192. DOI: https://doi.org/10.54216/IJNS.250415

[11] Khosrowshahi, H. N., Aghdasi, H. S., & Salehpour, P. (2025). A refined Greylag Goose optimization method for effective IoT service allocation in edge computing systems. Scientific Reports, 15(1), 1-24. https://doi.org/10.1038/s41598-025-00796-8

[12] Tingting Li, Wenqi Niu, Cun Ji, Edge user allocation by FOA in edge computing environment, Journal of Computational Science, Volume 53, 2021, 101390, https://doi.org/10.1016/j.jocs.2021.101390.

[13] Hou W, Chen Y (2025), "User allocation in multi-hop edge computing networks: a game-theoretical approach". International Journal of Web Information Systems, Vol. 21 No. 3 pp. 230–253, doi: https://doi.org/10.1108/IJWIS-07-2024-0190

[14] Qiang He; Guangming Cui; Xuyun Zhang; Feifei Chen; Shuiguang Deng; Hai Jin, "A Game-Theoretical Approach for User Allocation in Edge Computing Environment," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 3, pp. 515-529, 1 March 2020, https://doi.org/10.1109/TPDS.2019.2938944.

[15] J. Zhou, F. Chen, G. Cui, Y. Xiang and Q. He, "FEUAGame: Fairness-Aware Edge User Allocation for App Vendors," in IEEE Transactions on Parallel and Distributed Systems, vol. 35, no. 8, pp. 1429-1443, Aug. 2024, https://doi.org/10.1109/TPDS.2024.3409548.

[16] Phu Lai, Qiang He, Guangming Cui, Xiaoyu Xia, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, Yun Yang, QoE-aware user allocation in edge computing systems with dynamic QoS, Future Generation Computer Systems, Volume 112, 2020, Pages 684-694, https://doi.org/10.1016/j.future.2020.06.029.

[17] Phu Lai, Qiang He, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy & Yun Yang, (2018). Optimal Edge User Allocation in Edge Computing with Variable Sized Vector Bin Packing. In: Pahl, C., Vukovic, M., Yin, J., Yu, Q. (eds) Service-Oriented Computing. ICSOC 2018. Lecture Notes in Computer Science, vol 11236. Springer, Cham. https://doi.org/10.1007/978-3-030-03596-9_15

[18] Li, H., Fang, F., & Ding, Z. (2021). DRL-Assisted Resource Allocation for NOMA-MEC Offloading with Hybrid SIC. Entropy, 23(5), 613. https://doi.org/10.3390/e23050613

[19] Heidari, M. A. J. Jamali, N. J. Navimipour and S. Akbarpour, "A QoS-Aware Technique for Computation Offloading in IoT-Edge Platforms Using a Convolutional Neural Network and Markov Decision Process," in IT Professional, vol. 25, no. 1, pp. 24-39, Jan.-Feb. 2023, https://doi.org/ 10.1109/MITP.2022.3217886

[20] iu, X., Chai, Z., Li, Y., Cheng, Y., & Zeng, Y. (2023). Multi-objective deep reinforcement learning for computation offloading in UAV-assisted multi-access edge computing. Information Sciences, 642, 119154. https://doi.org/10.1016/j.ins.2023.119154

[21] S. Tian, C. Chang, S. Long, S. Oh, Z. Li and J. Long, "User Preference-Based Hierarchical Offloading for Collaborative Cloud-Edge Computing," in IEEE Transactions on Services Computing, vol. 16, no. 1, pp. 684-697, 1 Jan.-Feb. 2023, https://doi.org/10.1109/TSC.2021.3128603.

[22] Baskar, R., Mohanraj, E., Saradha, M., & Monika, R. (2025). Hybrid Prairie Dog and Dwarf Mongoose optimization algorithm-based application placement and resource scheduling technique for fog computing environment. Scientific Reports, 15(1), 1240. https://doi.org/10.1038/s41598-025-85142-8

[23] Guangming Cui; Qiang He; Xiaoyu Xia; Feifei Chen; Fang Dong; Hai Jin, "OL-EUA: Online User Allocation for NOMA-Based Mobile Edge Computing," in IEEE Transactions on Mobile Computing, vol. 22, no. 4, pp. 2295-2306, 1 April 2023, https://doi.org/10.1109/TMC.2021.3112941.

[24] Phu Lai; Qiang He; Feifei Chen; Mohamed Abdelrazek; John Hosking; John Grundy, "Online User and Power Allocation in Dynamic NOMA-Based Mobile Edge Computing," in IEEE Transactions on Mobile Computing, vol. 22, no. 11, pp. 6676-6689, 1 Nov. 2023, https://doi.org/10.1109/TMC.2022.3193366.

[25] Xin He; Jiaqi Zheng; Haipeng Dai; Bowen Liu; Wanchun Dou; Guihai Chen., "History-Assisted Online User Allocation in Mobile Edge Computing," 2022 IEEE International Conference on Web Services (ICWS), Barcelona, Spain, 2022, pp. 140-149, https://doi.org/10.1109/ICWS55610.2022.00034.

[26] R. Ramani, S. Edwin Raja, D. Dhinakaran, S. Jagan, G. Prabaharan, "MapReduce based big data framework using associative Kruskal poly Kernel classifier for diabetic disease prediction," MethodsX, Volume 14, 2025, 103210, https://doi.org/10.1016/j.mex.2025.103210.

[27] D Dhinakaran, S. M. Udhaya Sankar, S. Edwin Raja and J. Jeno Jasmine, "Optimizing Mobile Ad Hoc Network Routing using Biomimicry Buzz and a Hybrid Forest Boost Regression - ANNs" International Journal of Advanced Computer Science and Applications (IJACSA), 14(12), 2023. http://dx.doi.org/10.14569/IJACSA.2023.0141209

[28] C. Xu, G. Zheng and L. Tang, "Energy-Aware User Association for NOMA-Based Mobile Edge Computing Using Matching-Coalition Game," in IEEE Access, vol. 8, pp. 61943-61955, 2020, https://doi.org/10.1109/ACCESS.2020.2984798.