Balancing the Edge: Multi-Service Request Allocation in Urban 5G-MEC Environment

Supriya Dilip Tambe
Indian Institute of Technology, Hyderabad, India
cs18resch11002@iith.ac.in

Antony Franklin A.
Indian Institute of Technology, Hyderabad, India
antony.franklin@iith.ac.in

***Abstract*—In urban 5G Multi-access Edge Computing (MEC), overlapping server coverage and dense, heterogeneous user service demands often lead to uneven request distribution, causing resource under-utilization at some edge servers and overloading at others. This imbalance degrades the overall efficiency of the system, making load balancing essential for effective resource utilization and Quality of Service (QoS) assurance during multiple service placement and request allocation. We address this by formulating the Load-aware Service placement and Request allocation (LSR) problem as a multi-objective optimization model and proposing a scalable and efficient two-stage heuristic. The objective is to minimize server utilization variance while reducing service deployments and maximizing user coverage. In the heuristic, the first stage ensures maximum coverage through deterministic request allocation and greedy service placement, all within MEC constraints. The second stage iteratively refines service placement and request allocation to minimize utilization variance, offering reduced computational complexity compared to existing approaches. Experimental results demonstrate that our approach achieves significantly lower variance (65%) in optimal scenarios, reduces service deployments by 15%, increases user coverage by 15%, and establishes itself as an efficient centralized solution for quasi-static multi-service edge deployments in urban 5G-MEC environments.**

***Index Terms*—Load Balancing Strategies, Multi-access Edge Computing, Resource Optimization, Request Allocation, Service Placement.**

I. Introduction

In the era of 5G and beyond networks, the demand for ultra-low-latency applications, such as Augmented Reality (AR) navigation, real-time video analytics, and autonomous vehicles, has increased significantly. These applications/services rely on the Multi-access Edge Computing (MEC) infrastructure, where edge servers are deployed near users in urban environments to ensure localized data processing and minimal latency [1]. However, in such a resource-constrained and distributed environment, efficiently managing service placement and request allocation becomes a critical challenge for service providers [2].

Edge servers deployed at various locations host services such as IoT sensors, vehicular communication, health services, and content services, where traffic demand fluctuates dynamically based on location and time [1], [3]. Urban areas experience a high demand for vehicular communication during peak hours, whereas sensor communication is prevalent near transport hubs for traffic control and signaling purposes. Without efficient load balancing, some servers become overloaded, increasing response time and degrading Quality of Service (QoS); others remain underutilized [4]. Poor service placement leads to high resource utilization in some areas and inefficiencies in other locations. To enhance the coverage of user requests, edge servers operate in overlapping settings, enabling multiple servers to handle the same user request [5]. Unlike centralized cloud computing, MEC servers make real-time distributed placement decisions, adapting to dynamic service demand. High loads can cause failures or poor service quality, especially for smaller edge servers [6]. Effective request redirection ensures efficient resource use and service availability. Heterogeneous server capacities further complicate balancing, and static placement does not adapt to real-time demand shifts, leading to overloaded or idle servers [3].

An adaptive approach to service placement is crucial for balancing workloads in edge networks, where user service demand fluctuates based on location and time. A robust load-balancing mechanism must anticipate these variations and optimize service allocation to prevent overloading and underutilization of the resource. Efficient strategies should minimize service deployments while maximizing user coverage, ensuring optimal resource utilization. Proper planning of service deployment and request allocation increases service availability and reduces costly post-failure adjustments.

Here, we refer to this as **L**oad-balance aware **S**ervice placement and **R**equest Allocation (LSR) problem in urban overlapped 5G-MEC aims to optimize service placement and request allocation while ensuring balanced resource utilization. The optimal and heuristic approaches are strategically developed to minimize server utilization variance, a crucial metric for achieving effective load balancing and optimizing resource utilization in urban overlapped 5G-MEC environments. This objective is pursued in conjunction with maximizing user coverage and minimizing the number of service deployments. Without a well-structured allocation strategy, specific edge servers may become overburdened. In contrast, others remain underutilized, ultimately compromising the overall QoS. The proposed two-stage heuristic method systematically addresses this challenge. In particular, the heuristic's second stage iteratively refines service placement and request allocation decisions based on the deterministic placement determined in the first stage. This process ensures a uniform distribution of workloads across the edge servers, thereby mitigating variance in server utilization, reducing redundant service deployments, and effectively allocating user requests. Thus, proposed approaches offer an effective and efficient solution for maintaining balanced workloads and resource utilization.

In summary,

- We define a load-balancing strategy requirement during service placement and request allocation in urban overlapped 5G-MEC environments.
- We formulate the LSR problem as a multi-objective Binary Integer Programming (BIP) problem, incorporating constraints on service availability, computational capacity, coverage, and load balancing. The objective function maximizes the number of served user requests, minimizes the number of deployed services, and minimizes the variance in edge server utilization load to reduce load imbalance.
- To efficiently solve this problem, we develop a two-stage heuristic approach. The first stage focuses on initial deterministic user request allocation and service placement, ensuring maximum coverage while allocating requests to edge servers having maximum capacity. The second stage iteratively minimizes the utilization variance among edge servers by adjusting service and request allocation.
- Experimental results demonstrate that our proposed approach maintains balanced resource utilization with higher user coverage, reduces service deployments, and outperforms benchmark techniques.

II. Related Work

Significant research has been conducted in the edge computing domain, focusing on key areas such as edge user allocation, service placement, and computation offloading. Early work on edge user allocation addressed QoS-aware dynamic user allocation for a single service [5]. Service placement and request allocation are interdependent in cooperative MEC, requiring a joint optimization approach. The Joint Service placement and Request allocation (JSR) problem has been widely studied using iterative methodologies, where service placement is optimized first, followed by request scheduling [2], [7]. However, since these two aspects are closely intertwined, solving them separately may result in suboptimal solutions. Various solution techniques, including heuristics [2], [7], approximation methods [2], [8], Lyapunov optimization [9], and two-stage time-scale models [10], have been explored. Constraints such as computation resources, communication delay, budget, and storage costs have been incorporated into these models. In [2], the JSR problem was analyzing both sharable and non-sharable resources, employing optimal and greedy solutions for homogeneous and heterogeneous environments, without guaranteeing QoS for all services. Authors investigated JSR for data-intensive applications in [10], optimizing user requests under budget constraints while considering service deployment and migration costs with a two-stage time-based approach. In [11], authors presented a detailed survey of JSR, providing a comprehensive overview and classification of various algorithmic approaches for the service placement problem in integrated cloud-fog/edge environments. These JSR works do not consider load balancing, which is crucial for efficient resource utilization and minimizing failures due to request overloading.

From a load-balancing perspective, researchers have explored the utility and communication overhead of multi-hop task offloading [12]. In cloud infrastructure, authors proposed a hybrid optimization approach that combines the strengths of different metaheuristic algorithms to enhance load balancing in cloud environments [13]. The method aimed to minimize response time, optimize resource utilization, and propose security-aware strategies that leverage virtual machines. The authors proposed a hyper-heuristic that used ant-colony optimization to dynamically combine meta-heuristics for 5G MEC resource allocation, minimizing latency and balancing load [14]. It omits the critical factor of minimizing deployed services. In fog computing, traditional task offloading and load balancing problems have been added to reduce overloaded base station load using Particle Swarm Optimization (PSO), to minimize the migration cost after task placement [6]. In IoT-based user load balancing, researchers provide the assignment of moving vehicles leveraging device location and capacity [15].

Meanwhile, multi-agent deep reinforcement learning has been applied to minimize the load of overloaded base stations [16]. Logarithmic utility-based proportional fairness has been explored in vehicular networks to balance the assignment of moving vehicles to base stations [17]. In serverless edge computing, a distributed, multi-objective weighted round-robin approach has been proposed to efficiently balance heterogeneous AI workloads, thereby reducing latency and overload. Each service is deployed to only one server in a non-overlapping manner to minimize overloading while ensuring efficient resource utilization [6]. Authors introduced a distributed service placement strategy to balance workloads between edge and cloud resources, thereby reducing latency and overload. Each service is deployed to only one server in a non-overlapping tree-based edge environment [4]. However, many existing solutions, particularly those based on PSO, involve high computational complexity, as their iterative nature depends on the number of iterations and the use of an initial random solution, making them computationally expensive compared to deterministic approaches.

Novel Contribution: Unlike prior studies that mainly address balancing during user or task allocation, our work explicitly integrates load balancing into service placement, addressing the critical need for balanced resource utilization in overlapped and capacity-constrained 5G-MEC environments. While traditional service placement focuses on meeting coverage or latency, our approach jointly optimizes service distribution and server load variance, ensuring both feasibility and fairness in resource usage. While PSO is a well-established optimization method, it has mainly been applied under quasi-static assumptions at the request allocation stage, without addressing load balancing in service placement. We intend to highlight this gap rather than the generality of PSOs. The

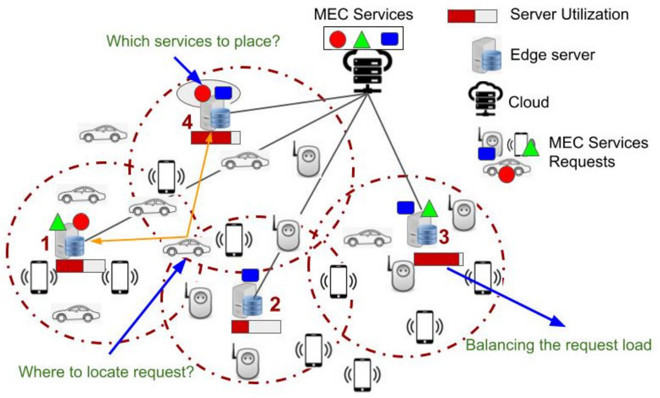


Fig. 1: Motivating Example.

TABLE I: Key Notations.

Notation	Description
$\mathcal{E}$	Set of edge servers.
$\mathcal{S}$	Set of available MEC services.
$\mathcal{U}$	Set of users.
A_e	Available service deployment capacity of edge server e.
C_e	Capacity of edge server e.
U_e	Server utilization of edge server e.
$edgereme_e$	Remaining capacity of edge server e.
$d_{e,u}$	Distance between edge server e and user request u.
R_e	Coverage radius of edge server e.
D_u	List of service demand requests of user u.
L_s	Load of service user request s.
V	Maximum acceptable variance.
$x_{e,s}$	Binary decision variable for Service deployment decisions.
$y_{e,u}$	Binary decision variable containing primary server$e$ for user u.
z_u	Binary decision variable for user u allocation to cloud.

proposed two-stage model first maximizes user coverage as a baseline, then iteratively refines service placement to minimize utilization variance, reducing iteration complexity compared to stochastic methods like PSO. This makes our solution more practical, computationally efficient, and scalable for quasi-static urban deployments.

III. System Model

In a 5G-MEC environment, edge servers deployed in overlapping coverage zones process user-generated MEC service requests while constrained by finite computational resources. As depicted in Fig. 1, multiple edge servers provide overlapping coverage, allowing flexibility in request allocation and introducing the risk of uneven server utilization while placing the services as per user request demand in a small-scale. Some servers may become overloaded while others remain underutilized, leading to degraded QoS.

The system model for the LSR problem considers a set of E edge servers, represented by $\mathcal{E} = 1, 2, \cdots, E$, deployed in an overlapped 5G MEC environment. Each edge server e has a finite computing capacity denoted by C_e, a service deployment capacity A_e, and a coverage radius R_e. A set of S MEC services, denoted by $\mathcal{S} = 1, 2, \cdots, S$, can be deployed on these servers, where each service s has a specific computational requirement denoted by L_s. The system also consists of U user requests, represented by $\mathcal{U} = 1, 2, \cdots, U$, dynamically generating service requests with demand D_u. The distance $d_{e,u}$ between an edge server and a user request is computed based on their locations. The service provider aims to deploy services while allocating user requests, ensuring efficient resource utilization, and distributing server utilization evenly. Table I summarizes all key parameters used in the model.

IV. Optimal Approach

We design a multi-objective optimization model to develop an optimal load-balanced service placement and request allocation (LSR-O) approach in a quasi-static scenario. The objective function is designed to jointly minimize the variance of edge server utilization, minimize the number of service deployments, and maximize user coverage. Minimizing variance ensures a balanced distribution of services and user requests across edge servers, promoting fair resource utilization. Reducing redundant service deployments leads to efficient use of computing resources and lowers operational costs. Maximizing user coverage ensures that most users are served within acceptable latency limits, crucial for supporting various edge applications.

We formulated the solution for this stage using BIP as follows: Let $x_{e,s}$ represent a binary variable if service s is deployed at edge server e. Let the binary variable $y_{e,u}$ indicate if the user requests u is assigned to edge server e, and z_u represents the user request u is served at the cloud.

As edge servers are resource-constrained, a limited number of services are deployed on each server. The services deployed at an edge server should not exceed this capacity and are specified in Eq. 1.

$$\sum_{s=1}^{|\mathcal{S}|} x_{e,s} \leq A_e, \quad \forall e \in \mathcal{E}. \tag{1}$$

A user request u can only be assigned to an edge server e if the required service s_u is deployed at that edge server, defined as a service availability constraint:

$$y_{e,u} \leq x_{e,s_u}, \forall e \in \mathcal{E}, \quad \forall u \in \mathcal{U}. \tag{2}$$

Each user request u must connect to exactly one edge server e, unless they are not served by cloud $z[u]$, which is 1 if the user is unserved.

$$\sum_{e=1}^{|\mathcal{E}|} y_{e,u} + z_u = 1, \quad \forall u \in \mathcal{U}. \tag{3}$$

A user request u can only be assigned to an edge server e within the coverage radius R_e. Here, $d_{e,u}$ represents the distance between user request u and edge server e. The coverage constraint represents this:

$$d_{e,u} \cdot y_{e,u} \leq R_e, \quad \forall u \in \mathcal{U}, \forall e \in \mathcal{E}. \tag{4}$$

The total computing load of all assigned users at edge server e should not exceed its available computational capacity C_e.

The load demand for each user service request u is represented by L_{D_u}.

$$\sum_{u=1}^{|\mathcal{U}|} L_{D_u} \cdot y_{e,u} \leq C_e, \quad \forall e \in \mathcal{E}. \tag{5}$$

To balance the load among available edge servers, we compute the normalized utilization of edge server e, based on the allocated user workloads.

$$U_e = \frac{C_e - \sum_{u=1}^{|\mathcal{U}|} L_{D_u - 1} \cdot y_{e,u}}{C_e}, \quad \forall e \in \mathcal{E}. \tag{6}$$

The following Eq. 7 and Eq. 8 measure the absolute difference in utilization between edge servers e and k. The Eq. 9 enforces an upper limit of allowable variance V on the imbalance to prevent excessive load deviation between edge servers.

$$\delta_{e,k} \geq U_e - U_k, \quad \forall e, k \in \mathcal{E}, e \neq k. \tag{7}$$

$$\delta_{e,k} \geq U_k - U_e, \quad \forall e, k \in \mathcal{E}, e \neq k. \tag{8}$$

$$\delta_{e,k} \leq \mathrm{V}, \quad \forall e, k \in \mathcal{E}, e \neq k. \tag{9}$$

By considering the constraints as mentioned earlier, the objective function aims to maximize the number of user requests served, minimize the number of deployed services, and reduce load imbalance across edge servers by minimizing variance, as defined below:

$$\text{Maximize} \quad \sum_{e=1}^{|\mathcal{E}|}\sum_{u=1}^{|\mathcal{U}|} y_{e,u} - \sum_{e=1}^{|\mathcal{E}|}\sum_{s=1}^{|\mathcal{S}|} x_{e,s} - \frac{1}{|\mathcal{E}|} \cdot \sum_{e=1}^{|\mathcal{E}|}\sum_{k=1}^{|\mathcal{E}|} (U_e - U_k)^2 \tag{10}$$

subject to: 1, 2, 3, 4, 5, 6, 7, 8, 9.

A. Motivation behind the Design of Load-Balance Aware System

To emphasize the necessity of load balancing in developing service placement and request allocation strategies, we analyze an experimental scenario illustrating its impact. Fig. 2 and Fig. 3 highlight the server utilization when 30 edge servers handle service requests from 500 users for 3 distinct services. Due to the spatial clustering of users and uneven service requirements, demand tends to concentrate on a subset of nearby servers, leaving others underutilized. The results in Fig. 2 indicate that approximately 5 servers experience full utilization, while 14 servers remain idle, leading to inefficient resource usage [3]. In contrast, after implementing an optimized load-balancing strategy, server utilization is distributed more evenly, ranging from 20% to 60%, ensuring a balanced allocation of service and request demand. Fig. 3 demonstrates the effectiveness of this approach by showcasing uniform utilization across all available servers.

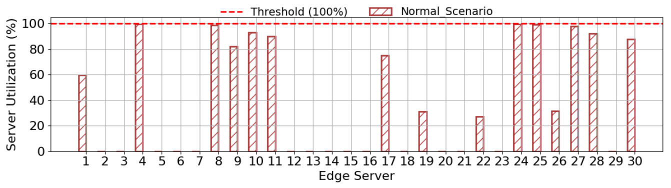


Fig. 2: Each Server's Utilization After Running The Optimal Solution Without Considering Load Balancing Constraints.

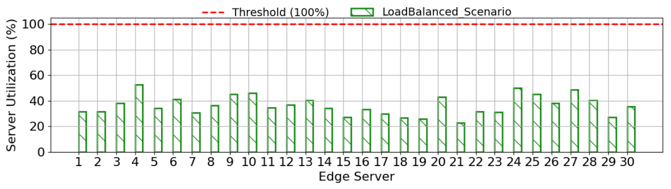


Fig. 3: Each Server's Utilization After The Optimal Load Balancing.

B. NP-Hardness of the LSR Problem

The JSR problem is already established as an NP-hard problem that effectively generalizes classical combinatorial problems such as the knapsack problem due to its multiple packing constraints [3]. This is due to its inclusion of numerous packing constraints, such as those related to storage, computation, and communication capacities. The incorporation of load-balancing objectives further amplifies its computational complexity. In particular, the goal of minimizing server utilization variance and achieving equitable workload distribution across edge servers introduces additional layers of complexity, transforming the problem into a multi-objective optimization task. This requires considering service placement, request allocation, and heterogeneous resource capacity constraints. The increased intricacy of these interdependent factors, especially within overlapped 5G-MEC environments, makes pursuing an optimal solution computationally impractical for large-scale instances.

V. Efficient Approach

We propose the heuristic LSR (LSR-H) approach to address scalability and adaptability, which reduces computation time and enables efficient on-demand load balancing in a large-scale urban 5G-MEC environment, where executing an optimal solution is not feasible and efficient. The LSR-H approach is a centralized, two-stage heuristic load balancing method that addresses these challenges effectively by focusing on service placement and request allocation, while minimizing variance in resource utilization. The need for a two-stage heuristic algorithm arises from the complexity of load balancing during service placement and request allocation in urban overlapped 5G-MEC environments. Balancing computational load without first ensuring deterministic service placement, considering the current request demand, leads to more service deployments and poor resource utilization. A one-stage approach fails to adapt to demand variations over time, resulting in inefficient placements requiring frequent reconfiguration and additional deployment costs. Here, the first stage efficiently maximizes user coverage. These placed services consider capacity-based load balancing and meet the constraints mentioned. The second stage iteratively refines the allocation of services and requests by adjusting based on variance, resulting in improved load distribution. Compared to existing approaches, such as PSO [6], the proposed heuristic is designed explicitly for MEC constraints, ensuring faster convergence while mitigating the initial randomness of PSO by prioritizing user coverage and deterministic service placement in the initial stage.

Algorithm 1 Stage 1: Initial Request and Service Allocation in Urban 5G-Edge Environment

```
1: Input: $\mathcal{U}$, $\mathcal{E}$, $\mathcal{S}$, $d$, $R$, $C$, $L$, $D$, $A$
2: Output: $x$, $y$, $z$
3: Initialize $x_{e,s}$ as 0 to store the service placement decision.
4: Initialize $y_{e,u}$ as 0 to store the user allocation decision.
5: Initialize $z_u$ as 0 to store user connected to cloud.
6: Update $usercoverage_u$ as edge servers in coverage for each user $u$.
7: Update $edgecovered_e$ as for users covered by each edge $e$.
8: Sort $usercoverage$ by the number of edges covering each user in ascending order.
9: Extract sorted user indices into $sorted\_indices$.
10: for $u \in sorted\_indices$ do
11:     if $|usercoverage_u| = 0$ then
12:         Add $u$ to $z$.
13:     else if $|usercoverage_u| = 1$ then
14:         Assign the user to the only available edge if capacity permits.
15:     else
16:         if service is already present then
17:             Select the best edge based on an available capacity.
18:             Allocate user request to the selected edge if feasible.
19:         else if service is not present then
20:             Select the best edge based on available capacity and services.
21:             Allocate service and user request to the selected edge if feasible.
22:         end if
23:     end if
24: end for
25: Return $x$, $y$, and $z$.
```

Stage 1: The Algorithm 1 aims to efficiently allocate users and services in an urban 5G edge environment. First, it initializes the matrices to store the placement of the service $x_{e,s}$, the allocation of user requests $y_{e,u}$, and the connected user requests to the cloud z_u. Then, it determines edge server coverage for each user and sorts users based on coverage constraints. Users with no edge server coverage are assigned to the cloud. In contrast, those with a single edge server are assigned if capacity allows. The algorithm checks if the required service is already deployed for users covered by multiple edges. If present, it selects the best edge based on available capacity; otherwise, it deploys the service and assigns users accordingly. The final decisions are made on the services' placement and the users' allocation. The stage-1 algorithm has a worst-case time complexity of $\mathcal{O}(|\mathcal{E}| \cdot |\mathcal{U}| + |\mathcal{U}| \cdot \log |\mathcal{U}|)$.

Algorithm 2 Stage 2: Load Balancing with Minimal Service Deployments

```
1: Input: $x$, $y$, $edgerem$, $C$, $D$, $L$, $A$, $V$.
2: Output: $x$, $y$ and $variance$.
3: Initialize: $prev\_variance \leftarrow \infty$
4: while True do
5:     Compute utilization using Eq. 6
6:     Compute $avg\_util \leftarrow \frac{\sum U}{|E|}$
7:     Compute $current\_variance \leftarrow$ variance($U$)
8:     if $|prev\_variance - current\_variance| < \epsilon$ then
9:         break                    ▷ Stop if variance is negligible
10:    end if
11:    $prev\_variance \leftarrow current\_variance$
12:    Identify $over\_utilized$ and $under\_utilized$ servers:
13:    if $over\_utilized = \emptyset$ or $under\_utilized = \emptyset$ then
14:        break                    ▷ No need for further adjustments
15:    end if
16:    for each $hv \in over\_utilized$ do
17:        $assigned\_users \leftarrow \{u \in U \mid y_{hv,u} = 1\}$
18:        for each $u \in assigned\_users$ do
19:            Identify the service required by the user and its load.
20:            Identify feasible target edge servers as per coverage and capacity.
21:            $existing$ list which service is already present.
22:            $deployable$ list which service is not present.
23:            for each $lv \in existing \cup deployable$ do
24:                if $x_{lv,s} = 0$ then
25:                    Deploy service and update $A_{lv}$,
26:                end if
27:                Reallocate user and balance the capacity,
28:                Compute latest $current\_variance$.
29:                if $current\_variance < prev\_variance$ then
30:                    break                ▷ Keep reallocation.
31:                else
32:                    Revert reallocation.
33:                end if
34:            end for
35:        end for
36:    end for
37: end while
38: Return $final\_util$, variance($final\_util$)
```

Stage 2: The algorithm in Algorithm 2 focuses on load balancing by minimizing utilization variance through iterative rearrangements of service deployments and request allocation in a 5GMEC environment. It begins by computing edge server utilization and variance, identifying overutilized and underutilized servers. If the variance change is negligible, the process stops. Otherwise, users are reallocated from overloaded servers to underloaded ones, prioritizing edges that already host the required service to minimize the need for new deployments. If necessary, a service is deployed on an underutilized edge only if it has deployment capacity. After each reallocation, the variance is recomputed, and changes are kept only if they improve load balancing; otherwise, they are reverted. The process iterates until no further variance improvement is possible. This approach ensures an even workload distribution, reduces excessive service deployments, and maintains efficient resource utilization. The stage-2 algorithm has a time complexity of $\mathcal{O}(|\mathcal{U}| \cdot |\mathcal{E}|)$.

VI. Experimental Analysis

In this section, we performed various experiments to evaluate the effectiveness of the proposed approach on a real-world EUA dataset [18]. The proposed approach is adaptable to any dataset that provides the locations of edge servers and user requests, enabling broader applicability beyond the EUA dataset. An Intel Core i7-6820HQ CPU, operating at 2.70GHz, is used as the platform for the experiments. The optimal LSR-O approach is solved using the IBM Cplex optimizer tool [19], and the heuristic LSR-H approach is developed using Python. To evaluate the performance of the proposed methods, we use the following performance metrics:

- **Average server utilization variance:** Average server utilization variance quantifies the spread of utilization levels across all edge servers, measuring the imbalance in resource usage to evaluate the effectiveness of load balance. The smaller, the better.
- **User coverage rate:** It is defined by the ratio of the edge server allocated to user requests and the total demand; the higher, the better.
- **Number of services deployed:** the total number of services deployed to serve the user service demand; The lower, the better.
- **Efficiency:** It is defined as the time required to run the required approach to get the final results; The lower, the better.
- **Number of iterations:** LSR-H and PSO are iterative approaches; therefore, their performance can be evaluated by the number of iterations required to reach convergence; The lower, the better.

To evaluate the performance of the proposed approach, it is compared with the following state-of-the-art approaches:

- **JSR approach [3]:** This is the optimal approach for service placement and request allocation without considering load balancing constraints and server utilization variance in the objective function.
- **PSO approach [6]:** This is the PSO approach used for load balancing while multi-service placement and request allocation considering required objectives as a fitness function. The experiment parameters used for PSO analysis are summarized in Table III.

TABLE II: Experiment Sets.

Set	No. of Edge Servers	No. of User Requests	No. of Services	Max Variance
Set #1	30	400	$1, 3, \cdots, 9$	0.3
Set #2	$10, 20, \cdots, 50$	500	4	0.3
Set #3	30	$100, \cdots, 500$	3	0.3
Set #4	30	400	3	$0.1, 0.3, 0.5, 0.7$

TABLE III: PSO Experiment Settings.

Number of Particles	Number of Iterations	Inertia Weight	Convergence
25	100	0.7	0.1

A. Experimental Settings and Real World Dataset

We used a real-world EUA dataset containing the locations of 125 base stations/edge servers, and 816 users [18]. Table II summarizes all the experiment settings used for analysis. Each edge server has a coverage radius of 250- 400 meters. Additionally, we randomly selected the service deployment capacity of the edge server from the total available services, and the processing capacity of any edge server is 100-300. Each service chooses a random computing load between 1-15. In addition, all experiments are performed 15 times on the experimental parameters selected for each setup, and the results are averaged.

B. Impact of Number of Services

To analyze the effect of the number of services on service deployments and server variance, we use the experiment settings mentioned in Set #1 in Table II. Fig. 4a illustrates that service deployments increase across all approaches as the number of services increases. LSR-O consistently achieves the highest number of deployments due to its optimized, load-balanced placement strategy, deploying significantly more services than JSR. PSO initially deploys more services than LSR-H, which is attributed to its use of random service placement. As the number of services increases, JSR exhibits the highest average utilization variance, highlighting a severe load imbalance and motivating the need for load-balancing strategies, as shown in Fig. 4b. The average utilization variance of LSR-O is approximately 65–78% lower than that of JSR. While PSO achieves the lowest variance due to its random service placement, it deploys significantly more services, making it impractical for real-world edge environments with limited resources. In contrast, LSR-O and LSR-H maintain balanced and moderate variance levels while efficiently deploying services, offering a more realistic and practical solution.

C. Impact of Number of Edge Servers

We analyze user coverage and edge server utilization in a 5G MEC environment using the experimental settings of Set #2 in Table II. Fig. 5a shows that as the number of

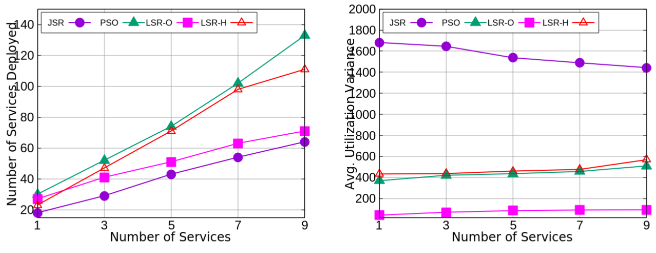

(a) Total Services Deployed. (b) Avg. Utilization Variance.

Fig. 4: Impact of Number of Services (Set #1).

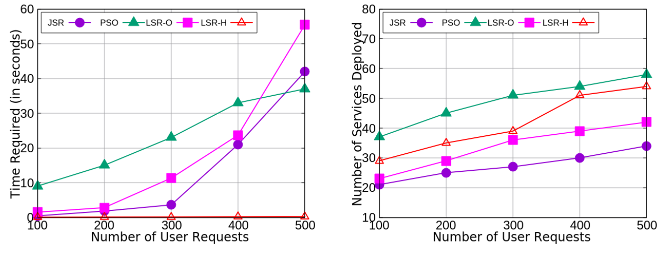

(a) Time Required to Run. (b) Total Services Deployed.

Fig. 6: Impact of Number of User Requests (Set #3).

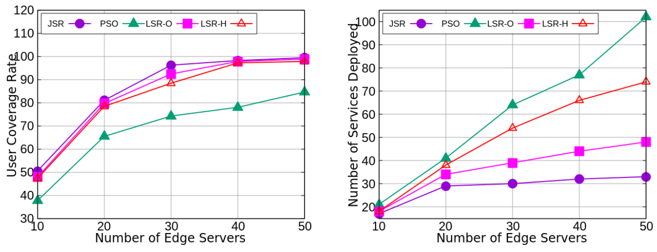

(a) User Coverage Rate. (b) Total Services Deployed.

Fig. 5: Impact of Number of Edge Servers (Set #2).

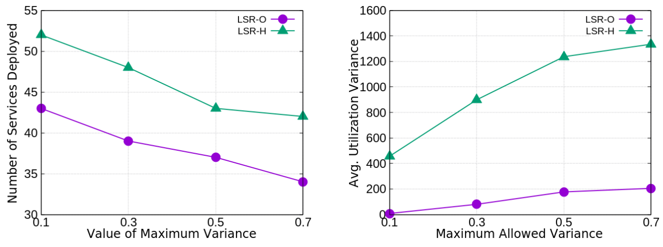

(a) Total Services Deployed. (b) Avg Utilization Variance.

Fig. 7: Impact of Maximum Variance Allowed (Set #4).

edge servers increases, the user coverage rate improves for all approaches. JSR achieves a higher user coverage rate as it does not consider load balancing, focusing solely on maximizing coverage. In contrast, LSR-O optimally determines user associations while maintaining a lower coverage rate by incorporating load balancing constraints with an allowable variance. LSR-H achieves a near-optimal user coverage rate by deterministically deploying services while maximizing user coverage, resulting in approximately 16% higher coverage than the iterative PSO approach. As shown in Fig. 5b, service deployments increase across all approaches as the number of edge servers rises to serve 500 user requests. PSO achieves higher service deployments due to its random initial placement strategy. JSR results in fewer deployments due to a lack of load-balancing considerations. With effective load balancing, LSR-O deploys services near optimally. Notably, LSR-H deploys approximately 15% fewer services than PSO on average across the observed range.

D. Impact of Number of User Requests

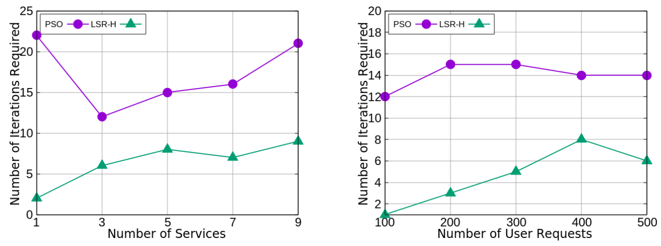
To assess the impact of the number of user requests on time efficiency and service deployments in a 5G MEC environment, we use the experimental settings of Set #3 in the Table. II. Fig. 6a shows that as the number of user requests increases, the time required by all approaches grows. The exponential growth in time required for JSR and LSR-O highlights the NP-hardness of the problem. At the same time, PSO incurs high computation time due to multiple iterations and particle evaluations for suboptimal solutions. At 500 user requests, LSR-H demonstrates remarkable computational efficiency, completing service placement in approximately 0.5 seconds, compared to 37 seconds for PSO. Fig. 6b confirms that as user requests increase, all approaches show a rising trend in service deployments. PSO deploys the most services due to its iterative nature, without requiring strict global optimization or an initial random service deployment. LSR-O, while optimal, deploys more services than JSR, as it enforces load-balancing constraints. LSR-H deploys near-optimal services by balancing efficiency and load variance through heuristics.

E. Impact of Maximum Variance (V)

To analyze the impact of the maximum allowable variance (V) on service deployments and utilization variance, we consider the experimental settings of Set #4, as outlined in Table II. This analysis compares the performance of LSR-O and LSR-H. As shown in Fig. 7a, increasing the allowable variance relaxes load balancing, enabling uneven request distribution and reducing redundant deployments, thereby lowering the total number of services deployed. Conversely, Fig. 7b shows that a higher allowable variance permits uneven load distribution across edge servers, thereby increasing the variance in average utilization. In urban 5G MEC environments, LSR-O proves to be the more efficient approach for minimizing service deployments, deploying approximately 16–18% fewer services than LSR-H across various variance levels, with a minimum deployment of 35 services at a variance of 0.7. Furthermore, LSR-O achieves significantly lower utilization variance, ranging from 27.59% to 50% of that observed in LSR-H, representing a reduction of 50% to 72.41%.

F. Impact on the Convergence

To analyze the effect on convergence (number of iterations) by varying the number of services and user requests, we use the experiment settings mentioned in Sets #1 and #3 in Table II. Fig. 8 demonstrates that evaluating convergence performance in service placement and request allocation reveals that



(a) By Varying Services. (b) By Varying User Requests.

Fig. 8: Impact on the Convergence (Set #1, Set #3).

the proposed LSR-H heuristic significantly outperforms PSO in terms of iterations. As the number of services increases, LSR-H maintains a consistently low and stable iteration count, demonstrating its scalability and effectiveness in handling service diversity with minimal overhead, as shown in Fig 8a. When the number of services is less, the PSO faces limited placement and allocation options, making it harder to achieve optimal load balancing. It leads to increased iterations as the algorithm requires more time to explore and converge to a balanced solution.

In contrast, PSO exhibits an irregular trend with higher iteration requirements, indicating its sensitivity to the complexity of the solution space and lack of guided decision-making. Similarly, as shown in Fig. 8b, with increasing user requests, LSR-H shows a gradual rise in iterations, reflecting its adaptive behavior to load distribution and user demand. PSO, however, shows negligible change, suggesting limited responsiveness to dynamic load conditions. The stability and low convergence cost of LSR-H make it more suitable for 5G-MEC environments where quick decision-making and efficient resource utilization are critical. Specifically, for service variations, LSR-H reduces iterations by approximately $40–80\%$, and for varying user requests, it achieves a reduction of around $45–85\%$.

G. Overall Analysis of LSR-H Approach:

The LSR-H approach provides a deterministic solution in the first stage by systematically maximizing user coverage, complemented by a capacity-based, greedy load-balancing strategy. In the second stage, it iteratively minimizes variance and efficiently allocates requests. In contrast, the PSO algorithm often encounters challenges in balancing all optimization objectives simultaneously in a broader solution space. It is primarily due to the stochastic nature of service deployment and a less targeted optimization strategy, which frequently results in convergence to suboptimal solutions.

For the specific problem under consideration, the deterministic and variance-oriented design of the LSR-H approach proves more effective in terms of solution quality. Additionally, the convergence time for load balancing through variance minimization in the second stage of LSR-H is significantly lower than that of PSO, which is influenced by the number of particles and iterations used.

From a time complexity perspective, the PSO algorithm exhibits a computational complexity of $\mathcal{O}(I \cdot P \cdot (|\mathcal{U}| + |\mathcal{E}| \cdot |\mathcal{S}|))$, where I denotes the number of iterations and P the number of particles. In contrast, the proposed LSR-H demonstrates improved scalability with a time complexity of $\mathcal{O}(|\mathcal{E}| \cdot |\mathcal{U}| + |\mathcal{U}| \cdot \log |\mathcal{U}|)$, making it more efficient for large-scale environments. Meanwhile, the other optimal algorithms, LSR-O and JSR, exhibit a worst-case time complexity of $\mathcal{O}(2^{|\mathcal{U}| \cdot |\mathcal{E}| \cdot |\mathcal{S}|})$. These comparisons highlight the computational advantage of LSR-H, particularly in scenarios involving large numbers of users and edge servers, where reduced complexity directly contributes to faster execution and better scalability.

However, it is essential to note that the overall computational efficiency, measured by the time required to execute the approach, also depends on the convergence rate and the size of the solution space. From this perspective, the efficiency of LSR-H is lower compared to that of PSO. This trade-off between solution quality and computational time is evident from the experimental analysis. Although LSR-H is best suited for quasi-dynamic conditions rather than fully dynamic real-time environments, we plan to explore online adaptive strategies for multi-service edge deployments in future work.

VII. Conclusion and Future Work

This work addresses the critical challenge of load balancing in service deployment and request allocation within a distributed, overlapped 5G edge computing environment from a service provider's perspective. To ensure optimal and efficient resource load balancing, we proposed the LSR-O approach as a multi-objective optimal solution and the LSR-H strategy as a centralized two-stage heuristic method, suitable for a real-world, scalable 5G-MEC environment. The LSR-H approach efficiently maximizes user coverage and ensures deterministic service placement in the first stage. In contrast, the second stage iteratively minimizes utilization variance by making necessary arrangements of services and user requests. Performance evaluations, based on key metrics such as reducing server utilization variance(65%) than optimal JSR with maximum user coverage(16%) and minimum service deployments(15%) than PSO, demonstrate that efficient load balancing prevents resource overloading and improves service availability. The proposed approaches ensure efficient workload distribution across edge servers. Future research may focus on developing adaptive strategies that dynamically adjust to real-time variations in user service demand and user mobility, leveraging reinforcement learning techniques to enhance load balancing. Additionally, considering the service's QoS features, such as meeting latency constraints through different edge services, can improve service performance.

Acknowledgment

This work was supported by the project "Design and Development of Cost-Effective Vehicular Mobile Edge Computing Platform for Automotive Industry" funded by SERB under the Core Research Grant (CRG).

References

[1] ETSI, "Multi-access Edge Computing (MEC): Phase 2: Use Cases and Requirements", Tech. Rep. ETSI GS MEC 002, ETSI, 2018.

[2] Ting He, Hana Khamfroush, Shiqiang Wang, Tom La Porta, and Sebastian Stein, "It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources", in *IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018, pp. 365–375.

[3] Konstantinos Poularakis, Jaime Llorca, Antonia M. Tulino, Ian Taylor, and Leandros Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks", in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 10–18.

[4] Zeinab Nezami, Kamran Zamanifar, Karim Djemame, and Evangelos Pournaras, "Decentralized edge-to-cloud load balancing: Service placement for the internet of things", *Ieee Access*, vol. 9, pp. 64983–65000, 2021.

[5] Phu Lai, Qiang He, Guangming Cui, Xiaoyu Xia, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, and Yun Yang, "Edge user allocation with dynamic quality of service", in *Service-Oriented Computing: 17th International Conference, ICSOC 2019, Toulouse, France, October 28–31, 2019, Proceedings 17*. Springer, 2019, pp. 86–101.

[6] Raj Mohan Singh, Geeta Sikka, and Lalit Kumar Awasthi, "Lbatsm: Load balancing aware task selection and migration approach in fog computing environment", *IEEE Systems Journal*, 2024.

[7] Matthew Turner and Hana Khamfroush, "Meeting users' qos in a edge-to-cloud platform via optimally placing services and scheduling tasks", in *2020 International Conference on Computing, Networking and Communications (ICNC)*, 2020, pp. 368–372.

[8] Konstantinos Poularakis, Jaime Llorca, Antonia M. Tulino, and Ian Taylor, "Service placement and request routing in mec networks with storage, computation, and communication constraints", *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1047–1060, 2020.

[9] Jingya Zhou, Jianxi Fan, Jin Wang, and Juncheng Jia, "Dynamic service deployment for budget-constrained mobile edge computing", *Concurrency and Computation: Practice and Experience*, vol. 31, 2019.

[10] Vajiheh Farhadi, Fidan Mehmeti, Ting He, Thomas F La Porta, Hana Khamfroush, Shiqiang Wang, Kevin S Chan, and Konstantinos Poularakis, "Service placement and request scheduling for data-intensive applications in edge clouds", *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 779–792, 2021.

[11] Imane Taleb, Jean-Loup Guillaume, and Benjamin Duthil, "A survey on services placement algorithms in integrated cloud-fog/edge computing", *ACM Computing Surveys*, vol. 57, no. 11, pp. 1–36, 2025.

[12] Lianlian Yang, Xing Zhang, Junjie Li, and Bo Lei, "Delay-minimization and load-balancing task offloading in mobile edge computing", in *2024 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, 2024, pp. 72–77.

[13] Perumal Geetha, SJ Vivekanandan, R Yogitha, and MS Jeyalakshmi, "Optimal load balancing in cloud: Introduction to hybrid optimization algorithm", *Expert Systems with Applications*, vol. 237, pp. 121450, 2024.

[14] Nadia Motalib Laboni, Sadia Jahangir Safa, Selina Sharmin, Md Abdur Razzaque, Md Mustafizur Rahman, and Mohammad Mehedi Hassan, "A hyper heuristic algorithm for efficient resource allocation in 5g mobile edge clouds", *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 29–41, 2024.

[15] Wei-Zhe Zhang, Ibrahim A Elgendy, Mohamed Hammad, Abdullah M Iliyasu, Xiaojiang Du, Mohsen Guizani, and Ahmed A Abd El-Latif, "Secure and optimized load balancing for multitier iot and edge-cloud computing systems", *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8119–8132, 2020.

[16] Manyou Ma, Di Wu, Yi Tian Xu, Jimmy Li, Seowoo Jang, Xue Liu, and Gregory Dudek, "Coordinated load balancing in mobile edge computing network: a multi-agent drl approach", in *IEEE International Conference on Communications(ICC)*, 2022, pp. 619–624.

[17] Yueyue Dai, Du Xu, Sabita Maharjan, and Yan Zhang, "Joint load balancing and offloading in vehicular edge computing and networks", *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2018.

[18] "Eua-dataset", https://github.com/swinedge/eua-dataset, 2021, [Online; accessed 10-April-2023].

[19] IBM, "Ibm cplex optimizer", https://www.ibm.com/in-en/analytics/cplex-optimizer, 2021, [Online; accessed 15-April-2023].