Available online at www.garmian.edu.krd

Passer Journal of Basic and Applied Sciences

Passer 7 (Issue 1) (2025) 345-354

Original Article

Iterative Weighted Randomized Algorithm for Edge Server Deployment in Mobile Edge Computing

Sakar Omer Khdr[1], Sadoon Azizi*[2], Hiwa Omer Hassan[3]

[1]*Department of Software and Informatics Engineering, University of Raparin, Ranya, Kurdistan Reign, Iraq.*
[2]*Department of Computer Engineering and IT, University of Kurdistan, Sanandaj, Iran.*
[3]*Department of Computer Science, University of Human Development, Sulaymaniyah, Kurdistan Reign, Iraq.*

Received 28 September 2024; revised 26 April 2025;
accepted 09 May 2025; available online 02 June 2025

DOI: 10.24271/PSR.2025.480834.1757

ABSTRACT

The rapid growth of Internet of Things (IoT) devices and the increasing demand for low-latency services in smart city environments have made efficient edge server placement (ESP) a critical challenge in mobile edge computing (MEC) systems. Effective placement of edge servers is essential for reducing network delays and achieving balanced workload distribution across servers, both of which directly affect the user experience. This paper addresses the ESP problem by formulating it as a multi-objective optimization problem that simultaneously minimizes the average distance between edge servers and base stations while reducing workload imbalance among the servers. To tackle this challenge, we propose an *Iterative Weighted Randomized Algorithm* (*IWRA*). The algorithm generates multiple potential placement solutions by employing a weighted roulette wheel selection, where base station weights are determined by their workloads. For each solution, edge servers are iteratively assigned to base stations, and clusters are formed by associating each base station with the nearest server. The solutions are evaluated based on normalized access distance and workload standard deviation, and the solution with the best score is selected as the final placement configuration. We validated the proposed algorithm through extensive simulations using both synthetic datasets and a real-world dataset from Shanghai Telecom, covering diverse network scenarios. The results demonstrate that our approach outperforms conventional placement methods, such as Random and Top-K placement, achieving an average reduction of 33% in access distance and a 27% improvement in workload balancing. These findings underscore the superior effectiveness of our method in addressing the ESP problem.

Keywords: Internet of Things (IoT), Mobile edge computing (MEC), Edge server placement (ESP), Weighted randomized algorithm, Access delay, Workload balancing.

1 Introduction

The rapid growth of technologies like the Internet of Things (IoT), 5G and 6G networks, and especially smart mobile devices, has completely transformed the face of modern life in terms of smart living, learning, business, entertainment, and communication [1]. These technologies can be utilized in various applications such as real-time language translations using natural language processing (NLP) [2], multi-player gaming, augmented reality [3], and video and image analysis in identifying objects [4]. It is, however, important to note that as a result of the proliferation of these IoT devices, data generation has increased exponentially, and the need for efficient data processing and analysis techniques at the source has become apparent. IoT devices are normally limited in terms of storage, CPU capacity, memory, battery life, and computational power, which poses significant challenges to their effective use.

Traditional solutions such as mobile cloud computing (MCC) and cloudlet-based architectures have been proposed to address these limitations. While MCC securely provides computational power to the cloud, it cannot perform real-time responses due to its inherently latent nature. Cloudlets, sometimes also referred to as interconnected high-capacity computers, guarantee much better performance and lower latency but they are more expensive [5]. However, these solutions are not without their drawbacks, particularly in terms of scalability, energy efficiency, and scalability.

Mobile edge computing (MEC) emerges as a promising solution that brings computing power closer to end-users by positioning servers at the network's edge [2]. This proximity further enhances

* Corresponding author
E-mail address s.azizi@uok.ac.ir (Associate Professor).
Peer-reviewed under the responsibility of the University of Garmian.

processing capabilities but also significantly reduces latency, hence ideal for real-time applications in smart cities, wireless metropolitan area networks (WMAN), and IoV. However, despite the advantages, the placement of edge servers in the MEC environments is a complex challenge [6]. With a limited number of servers available due to cost and energy constraints, determining the optimal locations for these servers to minimize network latency and balance workloads becomes critical.

Poor placement of servers in large-scale networks with hundreds or thousands of base stations can lead to workload imbalances, where some servers are underutilized while others become overloaded [7]. This imbalance increases user request waiting times and negatively impacts system performance. Additionally, the distance between users and their assigned servers directly affects access delay and network congestion, particularly for latency-sensitive applications requiring real-time processing. By focusing on load balancing and minimizing access delay, we aim to reduce system response times and enhance network-level Quality of Service (QoS) for end-users. These two objectives are critical for optimizing edge server placement in MEC systems, as they work synergistically to improve both user experience and system efficiency.

This paper proposes an **Iterative Weighted Randomized Algorithm (IWRA) to** address these challenges. It is designed to optimize both workload distribution among servers and the average distance between base stations and edge servers. Our approach assigns weights to base stations based on their respective workloads and generates multiple placement solutions, selecting the optimal configuration to balance these competing objectives. By reducing the average distance, the algorithm improves network performance and minimizes latency, while effective load balancing ensures that no server becomes overwhelmed, maintaining optimal service quality. Furthermore, the algorithm's lightweight iterative structure makes it highly scalable, enabling efficient deployment in large-scale MEC systems where traditional methods may struggle. We validate the performance of the proposed algorithm through extensive simulations using both synthetic datasets and real-world data from Shanghai Telecom, demonstrating its effectiveness in addressing scalability and operational challenges in MEC environments.

The parts of this study are structured as follows. We describe the review of the related works in Sect. 2. In Sect. 3, we describe the proposed Iterative Weighted Randomized Algorithm for the ESP problem. Sect. 4 includes the evaluation of our proposed algorithm. Sect. 5 concludes and points out the possible direction of future research

2 Related Work

Researchers have increasingly focused on the problem of ESP in MEC systems, as it plays a vital role in enhancing overall network efficiency. Numerous strategies have been developed to tackle this issue, focusing on aspects such as resource allocation, workload balancing, and latency reduction.

Mazloomi et al. [4] introduced a reinforcement learning (RL)-based framework designed for Multi-Access Edge Computing (MEC) to optimize server placement and workload allocation. It successfully achieves a cost-efficient MEC design by tackling the dual objectives of minimizing network delay and reducing the number of edge servers. The developed approach, which used Q-learning and Temporal Difference with Eligibility Traces learning (TDMC), efficiently optimized the two essential tasks of base station allocation and edge server placement. In reference [8], the authors proposed a multi-agent reinforcement learning (RL) algorithm that focuses on achieving the optimal positioning of mobile edge servers in wireless networks. The authors have formulated the ESP problem as a multi-objective optimization task, aiming to reduce network latency and balance server workloads. Utilizing RL agents that dynamically learn and adapt to the network environment. The proposed method improves overall network performance by emphasizing the importance of information sharing among RL agents. The approach's effectiveness is confirmed through tests on Shanghai Telecom data. In [9], authors proposed a DQN-ESPA algorithm based on reinforcement learning and Q-network, which is a reinforcement learning-based ESP algorithm. The primary objective is to optimize placements without relying on prior experience. DQN-ESPA outperforms existing algorithms like TKPA, SAPA, KMPA, and RPA, achieving at least 15% or better deployment performance for 300 edge servers, considering access delay and workload balance.

In reference [10], authors have presented a strategy for deploying edge servers in the industrial internet, which is aimed at minimizing latency and improving reliability. They established a positioning model, based on an enhanced Grey Wolf-Genetic algorithm, which demonstrates superior performance in optimal deployment location compared to existing methods. Similarly, in reference [11], authors have introduced an innovative fault-tolerant strategy for intelligent manufacturing using a binary-based Grey Wolf-Genetic policy. The proposed approach has focused on load balancing and optimizing deployment costs while ensuring reliability through fault-tolerant server deployment. The result achieved significant load and time savings in simulations. Asghari et al. [12] divided the resource positioning area into smaller subregions to speed up convergence. They utilized Coral Reefs optimization to prevent local optima and Butterfly optimization for local search, resulting in energy savings and reduced network latency. Experimental results demonstrate its effectiveness. In reference [13], the authors have introduced a method for placing edge processing networks using the artificial bee colony (ABC) algorithm. Their proposed approach focuses on cost function criteria and load balancing. The proposed method's performance is evaluated against load balancing criteria and compared with K-means clustering, demonstrating its superiority in load balancing over the clustering approach. The authors of [14] have introduced a Particle Swarm Optimization (PSO) inspired method for resource allocation in the 5G networks, aiming to maximize MEC server profit and minimize access delays. The method employs a profit model that assigns servers to base stations, with binary vectors representing

the initial solution for optimization. Experimental results from the Shanghai Telecom dataset and mini5Gedge emulator demonstrate the algorithm's effectiveness in achieving the highest profit.

Zeng et al. [15] explored cost-effective and efficient deployment of edge servers in Wireless Metropolitan Area Networks (WMANs), aiming to minimize the number of servers while meeting the Quality of Service (QoS) requirements. The proposed method transformed the problem into a graph theory-based minimum dominating set problem, considering both on-demand and uniform capacity scenarios. They used greedy and simulated annealing algorithms to demonstrate their feasibility in meeting the reduction needed in the number of servers and capacity constraints. Similarly, in [16], the authors introduced a method for solving the ESP problem using Shanghai Telecom's dataset. They have formulated the problem as a multi-objective optimization problem to balance server workloads and reduce access delay. Simulated annealing, hill-climbing, and genetic algorithms are applied to find efficient solutions. Experimental results demonstrate the genetic algorithm's superior performance in quickly exploring the solution space and minimizing the cost function. The authors of [17] have addressed distance considerations and workload for task offloading to edge servers, aiming to enhance user experience. They formulated the problem as a mixed-integer linear programming model. Subsequently, they proposed a novel solution that combines simulated annealing for server deployment with task allocation based on the Lagrangian duality theory. Simulation results demonstrated improved performance compared to conventional heuristics.

In reference [18], authors have proposed the multi-start power of d choices (MPdC) algorithm, which combines a greedy-randomized with a multistart procedure technique to solve the ESP problem efficiently. This approach aims to achieve the reduced network latency and load balancing by strategically choosing edge server locations based on predefined criteria. The proposed algorithm is computationally efficient in producing improved solutions for edge server placement in MEC environments. Cao et al. [19] investigated the deployment of edge servers in the IoV and proposed the PCMaLIA algorithm. Their approach focuses on factors such as transmission delay, workload distribution, energy usage, network reliability, deployment expenses, and the number of edge servers. The proposed algorithm’s performance was confirmed by comparing it with state-of-the-art methods.

Wang et al. [7] addressed the ESP challenge in the smart city within MEC. They formulated the ESP as a multi-objective optimization task aiming to achieve workload balancing and minimize the access delay of edge servers for mobile users. The research employed a Mixed Integer Programming (MIP) for finding the optimal solution and used the public Shanghai Telecom dataset to evaluate it. The result demonstrates that the proposed approach excels in reducing access delays and balancing workloads when compared to existing methods. Similarly, in reference [20], authors have proposed a multi-objective optimization framework for the placement and allocation of edge servers in MEC, considering the minimization of latency and workload balancing. The proposed approach, using the NSGA-II algorithm, is able to handle the inherent tradeoffs between these objectives with superior Pareto front diversity and convergence compared to other optimization methods. In this framework, the experiments carried out based on real datasets showed that the average latency reduction was up to 8.79% and improved workload distribution. In [21], authors have introduced the Benders_SD algorithm to reduce latency and deployment costs in WMANs within MEC. They formulated the ESP problem as a mixed-integer nonlinear programming (MINLP) model. Their proposed approach optimizes edge server deployment considering edge location, user capacity, and association. Extensive simulations demonstrate the method's effectiveness compared to existing approaches.

In [22], the authors have proposed an optimal edge-server placement (OESP) algorithm for healthcare applications, which decreases latency and deployment costs with full coverage. The work presents an AI-based priority mechanism to classify and prioritize the patients' cases in terms of urgency, which would improve the quality of service (QoS). Their algorithm illustrates, through simulations, the right placement of edge servers using dynamic determinations of location and shows cost reductions as high as 80% with no compromise into latency. The authors of [6] addressed ESP problem in MEC by proposing a recursive clustering approach for ESP, called RESP. The primary aim of RESP is optimizing the ESP to achieve reducing network traffic, minimize workload standard deviation and enhance load balancing strategies in MEC environments. The proposed approach is partitioning the MEC graph by using the recursive clustering technique based on the workload and geographical location of the base station. Recently, Zarei et al. [23] addressed the dual challenges of ESP and load distribution in MEC environments. They modeled these problems using the MINLP model and proposed an Ant Colony Optimization (ACO) algorithm for efficient server placement. The authors also proposed heuristic strategies for task allocation and scheduling to balance server workloads and minimize response times. This, through simulation for their approach using real trace drive data obtained from Shanghai Telecom, yields better results in the aspects of load balancing, average response time, and improving the chances of meeting the user deadline, compared to other approaches in the literature.

Although numerous valuable efforts have been made to address the ESP problem in MEC environments, exploring various aspects of the issue and proposing diverse solution approaches, many of these methods are computationally intensive and face scalability challenges in large-scale networks. This limitation highlights the need for simpler yet efficient methods capable of providing high-quality solutions within reasonable computation times. Motivated by this, our research focuses on developing a fast and effective algorithm that minimizes system response time by achieving a high degree of load balancing across edge servers and reducing the access delay between base stations and their nearest edge servers.

3 System Architecture and problem formulation

This section begins by describing the architectural framework of mobile edge computing environments. Following this, the edge server placement problem is mathematically modelled. A detailed description of the symbols used in the model, along with their corresponding definitions, is provided in Table 1.

Table 1: Symbols and Definitions.

Symbol	Definition
G	Undirected MEC network
B	A set of base stations
S	A set of edge servers
E	A set of links between base stations
s_i	i- th edge server
B_s	A set of base stations that send their requests to the server s
K	Number of edge servers
n	Number of base stations
L_b	Location of base station b
L_s	Location of edge server s
W_{s_i}	The workload of the edge server s_i
W_b	Workload of base station b
W'	The average workload of all edge servers
$d(b_i, s_j)$	Distance between base station b_i and server s_j
AD	Average Distance
SD	Workload standard deviation among edge servers
α and β	Weighting factors for balancing average distance and workload standard deviation.

3.1 System Architecture

We consider the architecture of the MEC environment, as illustrated in Figure 1, which consists of multiple end users, numerous base stations, and a collection of edge servers positioned at selected base stations, serving as cluster heads. The ESP problem can be modelled as an undirected network $G = (V, E)$ where $V = B \cup S$. Here, $B = \{b_1, b_2, \ldots, b_n\}$ represents the set of n base stations, and $S = \{S_1, S_2, \ldots, S_K\}$ denotes the set of K edge servers, where $K < n$. The set E represents the connections between base stations and edge servers in the graph. In this MEC network, edge servers are assumed to be co-located with existing base stations and are responsible for processing all user requests from the connected base stations [24]. Each edge server can connect to multiple base stations, which users access through these base stations. This architecture aims to reduce average distance and optimize workload balancing for service providers during edge server deployment.

We assume there are K edge servers designated for deployment across K distinct locations. Each edge server is presumed to have homogeneous computational capabilities with constrained resources for handling requests from mobile users. Each base station is connected to the nearest edge server, and all requests generated by the base station are routed to this server.Mobile

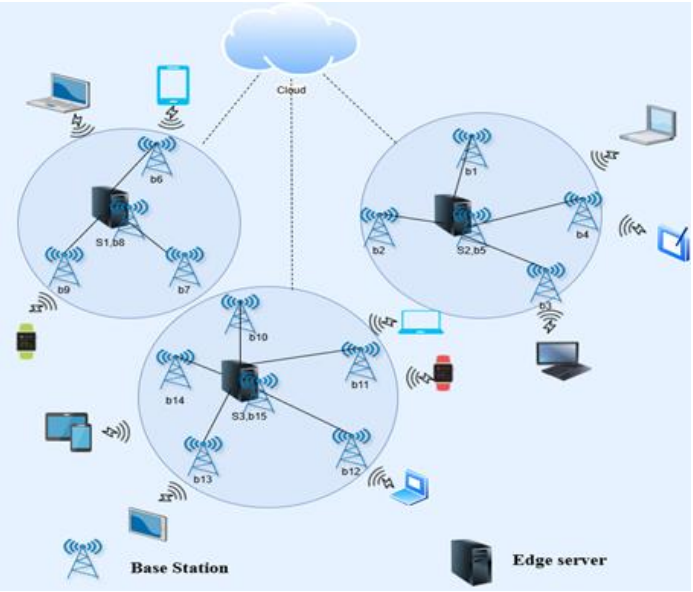


Figure 1: MEC System Architecture.

users first establish connections with their nearest base stations and subsequently access the edge servers through high-bandwidth links between base stations and edge servers.

3.2 Problem Definition and Formulation

In the context of MEC systems, the problem of edge server placement can be defined as follows: Given a network G with a significant number of base stations, the objective is to select K distinct base stations for deploying K edge servers. The primary goal is to optimize both the access delay between base stations and deployed edge servers and the workload distribution among the servers. Two key factors are considered to achieve these goals:

(1) Average Distance Between Edge Servers and Base Stations: This metric aims to minimize the data transmission distance, thereby reducing access delay.

(2) Standard Deviation of Workloads: This metric reflects the variability in server workloads across different clusters. Lower variability indicates a more balanced workload distribution, which is essential for efficient resource utilization.

These two factors are merged into a single objective function that seeks to minimize both workload variability and average distance, ensuring an optimal placement of edge servers.

Constraints

To formulate the problem more precisely, we define the following constraints:

Workload Assignment: Each edge server s_i is responsible for processing all incoming requests from the base stations that it covers, as defined below.

$$W_{s_i} = \sum_{b \epsilon B_s} W_b \quad \text{where } W_{s_i} \text{ is the workload of edge server } s_i \epsilon S \tag{1}$$

Exclusive Assignment of Base Stations: Every base station is linked to exactly one edge server, and no base station is shared between multiple servers, as stated in the following equation.

$$E_i \cap E_j = \emptyset, \quad \text{for all } i, j \in \{1, 2, \dots, K\} \tag{2}$$

Objective Functions

In the ESP problem, our main goals are minimum average distance from base stations to servers and workload balancing among servers.

Average Distance: This metric corresponds to the actual distance between base stations from the nearest edge server, with shorter distances leading to lower latency and better user experience. The average distance is computed as shown in equation (3).

$$AD = \frac{\sum_{j=1}^{K} \sum_{b_i \in B_j} d(b_i, s_j)}{n} \tag{3}$$

Where $d(b_i, s_j)$ is the Euclidean distance between the base station b_i and its nearest edge server s_j, which represents the access delay and helps minimizes response time. It is calculated using the following equation.

$$d(b_i, s_j) = \sqrt{(Long_{b_i} - Long_{s_j})^2 + (Lat_{b_i} - Lat_{s_j})^2} \tag{4}$$

Where, Long $(_{b_i})$ is the longitude and Lat$(_{b_i})$ latitude of the i-th base station. Similarly, Long_{s_j} and Lat_{s_j} represent the latitude and longitude of the i-th edge server.

Load Balancing: To evaluate workload balance, we use the standard deviation of the workload distribution across all edge servers, as defined in equation (5). The average workload is calculated in equation (6), providing a measure of how evenly distributed the workloads are across the servers.

$$SD = \sqrt{\frac{\sum_{i=1}^{K} (W_{s_i} - W')^2}{K}} \tag{5}$$

$$W' = \frac{1}{K} \sum_{i=1}^{K} W_{s_i} \tag{6}$$

where W_{s_i} denotes the workload of server s_i and W' is the average workload across all servers.

Our objective function is to optimize the performance of the edge server's placement in such a way that both the average distance (AD) and the standard deviation of workloads (SD) are minimized, leading to reduced access delay and a balanced distribution of workloads between the servers.

$$\min(\alpha . AD + \beta . SD) \tag{7}$$

where α and β are weighting factors that normalize the importance of minimizing average distance versus workload standard deviation; these two factors are combined into a single function. These factors can be adjusted based on the specific requirements or priorities of the system.

4 Proposed Algorithm

In this section, we propose a heuristic algorithm to solve the edge server placement problem in MEC environments. Therefore, main goals of the proposed algorithm are integrated optimization of the average distance between edge servers and base stations, and the standard deviation of workloads among the servers. To do so, we suggest a weighted randomized algorithm that generates a potential solution for the problem. This algorithm will run several iterations and choose the best solution from those iterations as the final solution. The following subsections detail the proposed algorithm and its corresponding pseudocode.

The proposed algorithm takes the number of edge servers K as input, the MEC network consisting of a set of base stations and the topology between them, and also the workload of each base station. The algorithm proceeds through the following steps to generate an optimal solution:

Step 1: In this step, each base station is assigned a weight depending on its workload. The total workload of all base stations is calculated initially to achieve the weight of the base stations. Then, the workload of each base station is divided by the total to determine its relative weight.

Step 2: In this step, one base station is selected for each of the K servers according to its weight. The selection is made using the roulette wheel selection method, ensuring that each server is placed on a different base station to satisfy Constraint (2).

Step 3: Clusters are formed in this step. The base stations hosting the edge servers act as cluster heads. The remaining base stations are assigned to the cluster of the nearest cluster-head, based on the minimum distance to the corresponding edge server.

Step 4: After forming the clusters, a solution is effectively generated based on the weighted random strategy. A multi-start procedure is executed to improve this strategy in this step. Specifically, the weighted random strategy is run independently ***Max_itr*** times, resulting in *Max_itr* distinct solutions.

Step 5: This step involves selecting the best solution from the solutions generated in Steps 1 to 4. The normalized weighted sum technique is used to identify the final solution. First, the average distance and the standard deviation of the workloads for all generated solutions are calculated using Eqs. (3) and (5), respectively. The maximum average distance and maximum standard deviation of the workload are then identified. The normalized average distance and the normalized workload standard deviation are computed to evaluate the performance of each solution Sol_i as follows:

$$\mathrm{Sol}_{\mathrm{i}}^{\mathrm{normAD}} = \frac{\mathrm{Sol}_{\mathrm{i}}^{\mathrm{AD}}}{\mathrm{maxAD}} \quad (8)$$

$$\mathrm{Sol}_{\mathrm{i}}^{\mathrm{normSD}} = \frac{\mathrm{Sol}_{\mathrm{i}}^{\mathrm{SD}}}{\mathrm{maxSD}} \quad (9)$$

The score for each solution is then calculated using the following formula:

$$Sol_i^{score} = \alpha \times \mathrm{Sol}_{\mathrm{i}}^{\mathrm{normAD}} + \beta \times \mathrm{Sol}_{\mathrm{i}}^{\mathrm{normSD}} \quad (10)$$

Step 6: Finally, the solution with the lowest $\mathrm{Sol}_i^{\mathrm{score}}$ is selected as the final solution.

Algorithm 1 illustrates the pseudocode of the proposed algorithm, with each step numbered consistently with the detailed explanations provided above. This structured approach facilitates easier comprehension of the algorithm's workflow and alignment with the textual descriptions.

Algorithm 1. Proposed Algorithm	
Input:	K: Number of edge servers B: Set of base stations W_b: Workload of each base station $b \in B$ Max_itr: Maximum number of iterations
Output:	Optimal placement of K edge servers

1. *Initialize* the set of base stations B and their workloads W_b.
2. *Compute* the total workload $W^{\mathrm{total}} = \sum W_b$ for all $b \in B$.
3. for each base station $b \in B$ do
   Compute its weight as W_b / W^{total}.
4. for $itr = 1$ to Max_itr do
   for each server, $k \in K$ do
   Select a base station b_k using roulette wheel selection based on weights.
   Place server k on base station b_k.
   Form clusters by assigning each base station $b \in B$ to the nearest server's cluster.
   Compute AD and SD for the current solution using Eqs. (3) and (5), respectively.
5. *Normalize* AD and SD across all generated solutions using Eqs. (8) and (9), respectively, and *Compute* $\mathrm{Sol}_i^{\mathrm{score}}$ for each solution using Eq. (10).
6. *Select* the solution with the minimum score as the final solution.

4.1 Time Complexity Analysis

The time complexity of our proposed algorithm can be summarized as follows:

Step 1: Calculate the total workload and weights by taking $O(n)$, where n is the number of base stations.

Step 2: Select K base stations using roulette wheel selection with a complexity of $O(K \times n)$.

Step 3: Construct clusters require computing distances, which are equal to $O(K \times n)$.

Step 4: The total complexity of repeating the weighted random strategy Max_itr times is $O(Max\_itr \times K \times n)$.

Step 5: Normalizing and scoring the solutions are equal to $O(Max\_itr)$.

Overall, the time complexity of the algorithm is $O(Max\_itr \times K \times n)$, which is efficient given the iterative nature of the multi-start approach.

5 Performance Evaluation

In this section, we assess and compare the performance of the proposed algorithm using both real-world and synthetic datasets. We begin by detailing the simulation settings, followed by a description of the datasets used in the experiments. Next, we introduce the algorithms used for comparison, and finally, we present and explain the simulation results.

5.1 Experiment setup

To assess the performance of our proposed method, we performed tests utilizing both a real dataset sourced from Shanghai Telecom and a simulated dataset. We implemented our algorithm and evaluated its performance, contrasting it with other standard algorithms on randomly generated networks. The experiments were conducted using Python 3.11, with the corresponding source code developed for this purpose. The experimental design was structured around three key components: First, we analyzed the artificial dataset created specifically for evaluating our approach. Second, we determined the optimal number of edge servers and deployed them using our algorithm. Finally, we compared the results of our approach with alternative placement methodologies, such as the random and top-k algorithms, to assess its relative performance.

5.2 Dataset Description

To demonstrate the effectiveness of our proposed algorithm, we performed simulations using real network data from the Shanghai Telecom dataset in China, as depicted in Figure 2. This dataset contains mobile user information accessing 3,233 base stations and their workloads for 15 days. After preprocessing and filtering out irrelevant information, we extracted useful data for 2,770 base stations. The dataset includes the geographical coordinates (longitude and latitude) of each base station, the number of users, and the workload, represented by the number of requests from different users for 15 consecutive days. For our experiments, we randomly selected a subset of this processed dataset.

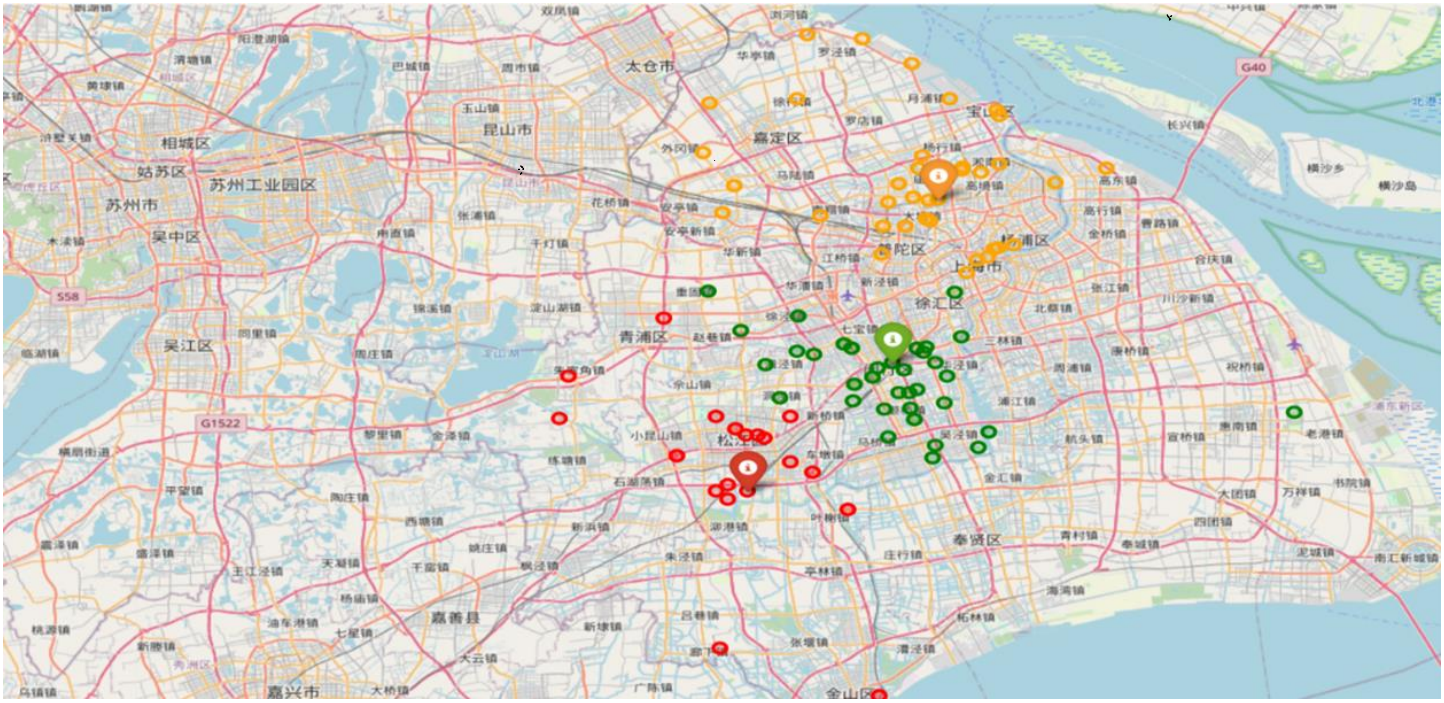


Figure 2: Placement of K-edge servers in the distribution map of the Shanghai Telecom's mobile communication base station dataset.

For each base station in the dataset, we recorded the average workload over 15 days based on mobile user requests accessing edge servers in an MEC environment. We then calculated the total number of user requests directed to each base station during this period and determined the average daily requests to assess the base stations' workload, as shown in Table 2. The workload for each base station was determined using the start and end times of the mobile user requests.

Table 2: An overview on the Shanghai dataset.

BS_ID	BS_Lat	BS_Long	BS_Workload	Num_Requests
BS_1	31.237872	121.470259	76	2
BS_2	31.25563	121.508513	97	1
BS_3	31.240686	121.480699	508	14
BS_4	31.223665	121.458791	391	5

BS_5	31.251584	121.4917	529	30
BS_6	31.174053	121.282683	605	33
BS_7	31.042415	121.482986	361	3
BS_8	31.137509	121.40768	1464	68
BS_9	30.996346	121.297687	60	1
BS_10	31.132785	121.413457	649	27

5.3 Comparison algorithms

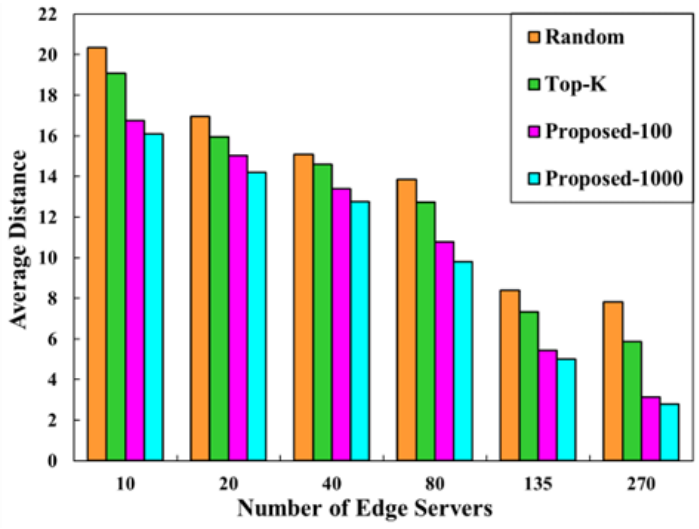
To assess the performance of our proposed algorithm in terms of workload balance and access distance, we conducted comparisons with several placement approaches, including Random Placement, the Top-K Placement Algorithm, and our proposed algorithm with different iteration counts.

- **Random Placement**: In this approach, edge servers are randomly placed at base stations. Once the edge servers are positioned, base stations are iteratively assigned to the nearest edge server.
- **Top-K Placement [7]**: This method places the K edge servers at the K base stations with the highest workloads. The remaining base stations are then assigned to the closest edge servers.
- **Proposed-100**: For this variation of our algorithm, the maximum number of iterations *Max_itr* is set to 100. This configuration allows us to evaluate the algorithm's performance with a moderate iteration count.
- **Proposed-1000**: In this case, the maximum number of iterations is increased to 1000, enabling us to assess the impact of a higher iteration count on the performance of the proposed algorithm.

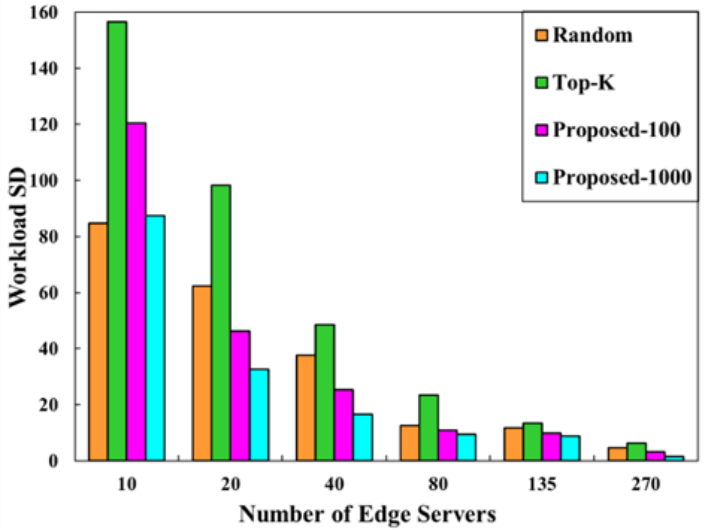
It is worth mentioning that for the proposed algorithm, we set $\alpha = \beta = 0.5$.

5.4 Results

In the first scenario, we analyzed the Shanghai Telecom dataset, specifically examining 2,770 base stations, and compared the performance of different placement methods by varying the number of edge servers to 10, 20, 80, 135, and 270. The results of this comparison are presented in **Figure 3**, which illustrates the performance of our algorithm against other algorithms. **Figure 3(a)** compares the algorithms in terms of average distance, while **Figure 3(b)** compares them in terms of workload standard deviation. As observed in the results, increasing the number of edge servers leads to a reduction in both the average distance and workload standard deviation across all algorithms. The Top-K algorithm, which greedily selects base stations with the highest workloads for server placement, results in a significant workload imbalance among the servers. In contrast, our proposed algorithm, with both 100 and 1000 iterations, outperforms the Random and Top-K algorithms. Additionally, increasing the number of iterations in our algorithm improves its performance in terms of both average distance and workload standard deviation.



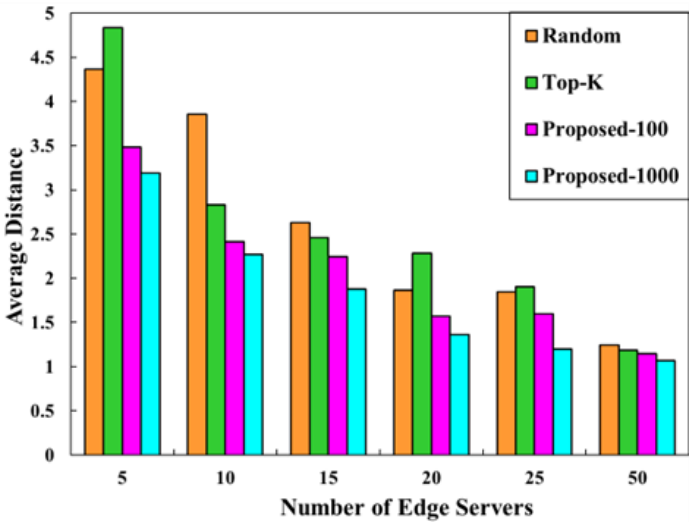
(a) Average Distance



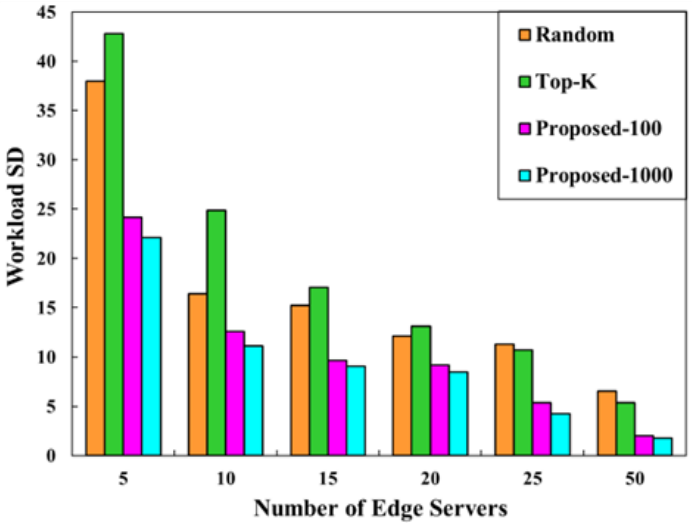
(b) Workload Standard Deviation

Figure 3: Simulation results for the Shanghai dataset

In the second scenario, we considered two cases: one with 100 base stations and the other with 500 base stations, with the results presented in Figures 4 and 5, respectively. In both cases, the number of servers was varied among 5, 10, 15, 20, 25, and 50. The key observations from this scenario are as follows. As in the first scenario, increasing the number of edge servers reduces both the average distance and workload imbalance for all algorithms. However, as the number of servers increases, the performance of the Top-K algorithm in terms of average distance declines sharply. The primary reason for this is that Top-K typically places servers at base stations near city centers, which results in a significant increase in the distance between other base stations and their assigned edge servers. Consequently, this results in a significant increase in the average distance for the Top-K algorithm. In contrast, our proposed algorithm, which considers both workload balancing and average distance when placing servers, consistently outperforms the Random and Top-K algorithms on both metrics. The number of iterations increases in our algorithm, and the quality of the final solution improves. In conclusion, the proposed algorithm demonstrates superior performance, effectively reducing both the average distance and workload standard deviation. It also achieves a well-balanced optimization between these two critical metrics.

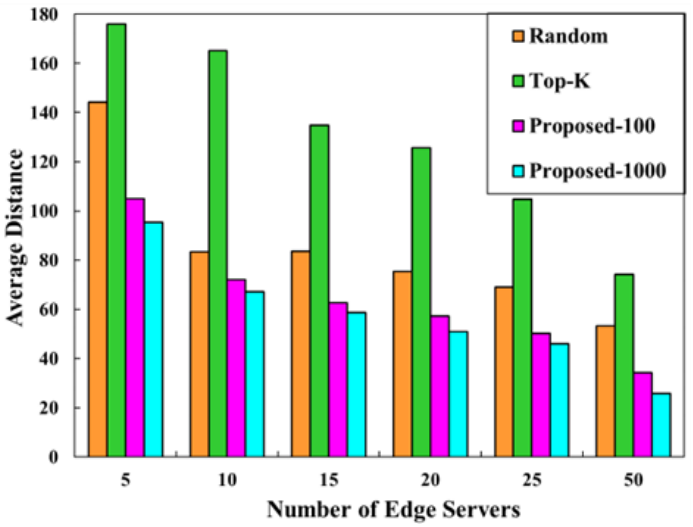


(a) Average Distance

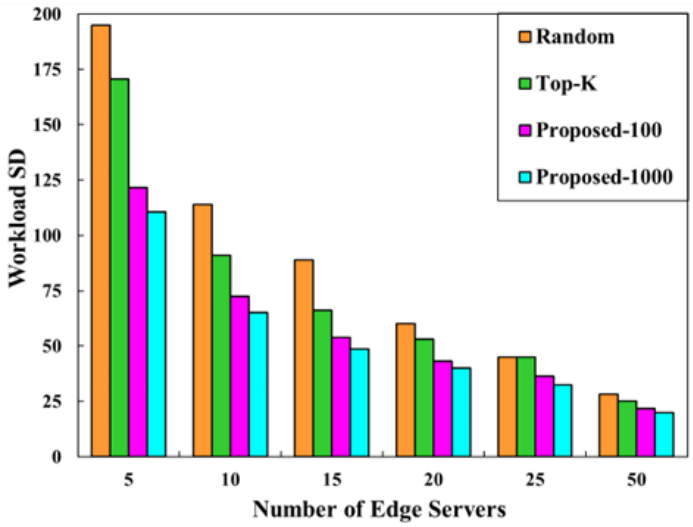


(b) Workload Standard Deviation

Figure 4: Simulation results for the artificial dataset with 100 base stations



(a) Average Distance



(b) Workload Standard Deviation

Figure 5: Simulation results for the artificial dataset with 500 base stations

Conclusion and Future Work

In this study, we address the edge server placement problem in MEC environments to optimize both workload distribution and access distance. More specifically, an iterative weighted randomized algorithm is proposed, combining the strengths of random and Top-K approaches. Next, our algorithm is extensively tested through simulation-based evaluations by using both real-world datasets from the Shanghai Telecom network and artificially generated datasets. These results confirm that our algorithm outperforms the current placement methods of Random and Top-K across different scenes on the metrics of the minimum average distance between edge servers and base stations, with both a minimal average distance and a balanced workload distribution. Experiments were conducted on various scenarios ranging from small-scale to large-scale networks so as to confirm the robustness and efficiency of our approach. It provides a significant enhancement in edge server placement strategies that are both practical and efficient for real-world MEC deployments. Future work should focus on further developing real-world multi-objective optimization algorithms and adapting them to highly mobile users with dynamic location changes.

References

[1] Jia, M., Cao, J., & Liang, W., Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks, *IEEE Trans. Cloud Comput* **5**, 725-737 (2015).

[2] Alfakih, T., Hassan, M. M., Gumaei, A., Savaglio, C., & Fortino, G., Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA, *IEEE Access* **8**, 54074-54084 (2020).

[3] Hammoud, A., Sami, H., Mourad, A., Otrok, H., Mizouni, R., & Bentahar, J., AI, blockchain, and vehicular edge computing for smart and secure IoV: Challenges and directions, *IEEE Internet Things Mag* **3**, 68-73 (2020).

[4] Mazloomi, A., Sami, H., Bentahar, J., Otrok, H., & Mourad, A., Reinforcement learning framework for server placement and workload allocation in multiaccess edge computing, *IEEE Internet Things J* **10**, 1376-1390 (2022).

[5] Bahl, P., Han, R. Y., Li, L. E., & Satyanarayanan, M., Advancing the state of mobile cloud computing, *Proc. 3rd ACM Workshop Mob. Cloud Comput. Serv*, 21-28, (2012).

[6] Vali, A. A., Azizi, S., & Shojafar, M., RESP: A Recursive Clustering Approach for Edge Server Placement in Mobile Edge Computing, *ACM Trans. Internet Technol* **24**, 1-25 (2024).

[7] Wang, S., Zhao, Y., Xu, J., Yuan, J., & Hsu, C.-H., Edge server placement in mobile edge computing, J. *Parallel Distrib. Comput* **127**, 160-168 (2019).

[8] Kasi, M. K., Abu Ghazalah, S., Akram, R. N., & Sauveron, D., Secure mobile edge server placement using multi-agent reinforcement learning, *Electronics* **10**, 2098 (2021).

[9] Luo, F., Zheng, S., Ding, W., Fuentes, J., & Li, Y., An edge server placement method based on reinforcement learning, *Entropy* **24**, 317 (2022).

[10] Wang, Z., Zhou, Y., Jin, X., Chen, Y., & Lu, C., An edge server deployment approach for delay reduction and reliability enhancement in the industrial internet, *Wirel. Netw* **30**, 1-15 (2023).

[11] Wang, Z., Zhang, W., Jin, X., Huang, Y., & Lu, C., An optimal edge server placement approach for cost reduction and load balancing in intelligent manufacturing, *J. Supercomput* **78**, 4032-4056 (2022).

[12] Asghari, A., Sayadi, M., & Azgomi, H., Energy-aware edge server placement using the improved butterfly optimization algorithm, *J. Supercomput* **79**, 14954-14980 (2023).

[13] Zhou, B., Lu, B., & Zhang, Z., Placement of Edge Servers in Mobile Cloud Computing using Artificial Bee Colony Algorithm, *Int. J. Adv. Comput. Sci. Appl* **14**, 621-637 (2023).

[14] Li, Y., Zhou, A., Ma, X., & Wang, S., Profit-aware edge server placement, *IEEE Internet Things J* **9**, 55-67 (2021).

[15] Zeng, F., Ren, Y., Deng, X., & Li, W., Cost-effective edge server placement in wireless metropolitan area networks, *Sensors* **19**, 1-21 (2018).

[16] Kasi, S. K. et al., Kasi, S.K., Kasi, M.K., Ali, K., Raza, M., Afzal, H., Lasebae, A., Naeem, B., Saif ul Islam., & Rodrigues, Joel J. P. C., Heuristic edge server placement in industrial Internet of things and cellular networks, *IEEE Internet Things J* **8**, 10308-10317 (2020).

[17] Huang, P.-C., Chin, T.-L., & Chuang, T.-Y., Server placement and task allocation for load balancing in edge-computing networks, *IEEE Access* **9**, 138200-138208 (2021).

[18] Havas, S., Azizi, S., & Abdollahpouri, A., A Multistart Power of d Choices Strategy for Edge Server Placement Problem, *in 2023 7th Int. Conf. Internet Things Appl. (IoT)*, IEEE, 1-6 (2023).

[19] Cao, B., Fan, S., Zhao, J., Tian, S., Zheng, Z., Yan, Y., & Yang, P., Large-scale many-objective deployment optimization of edge servers, *IEEE Trans. Intell. Transp. Syst* **22**, 3841-3849 (2021).

[20] Ghasemzadeh, A., Aghdasi, H. S., & Saeedvand, S., Edge server placement and allocation optimization: a tradeoff for enhanced performance, *Cluster Comput* **27,** 1-15 (2024).

[21] Shao, Y., Shen, Z., Gong, S., & Huang, H., Cost-Aware Placement Optimization of Edge Servers for IoT Services in Wireless Metropolitan Area Networks, *Wirel. Commun. Mob. Comput* **2022**, (2022).

[22] Jasim, A. M., & Al-Raweshidy, H., Optimal intelligent edge-servers placement in the healthcare field, *IET Netw* **13**, 13-27 (2024).

[23] Zarei, S., Azizi, S., & Ahmed, A., Optimizing edge server placement and load distribution in mobile edge computing using ACO and heuristic algorithms, *J. Supercomput* **81**, 1-32 (2025).

[24] Peng, K., Qian, X., Zhao, B., Zhang, K., & Liu, Y., A new cloudlet placement method based on affinity propagation for cyber-physical-social systems in wireless metropolitan area networks, *IEEE Access* **8**, 34313-34325 (2020).