GMM-TDQN: Two-Stage Multi-Objective Reinforcement Learning for Large-Scale Edge Server Deployment

Zhou Zhou, Tingyu Zheng, Yifu Zeng*
School of Computer Science and Engineering, Changsha University
zhzhou@ccsu.edu.cn, ztyzty@stu.ccsu.edu.cn, nscc2024@hnu.edu.cn

Abstract

The deployment of edge servers plays a crucial role in supporting large-scale edge computing systems, where multiple conflicting objectives—such as latency, energy consumption, load balancing, and service reliability—must be jointly optimized in complex, dynamic environments. Existing solutions often struggle to scale effectively or to balance these objectives in a unified learning framework. In this paper, we propose GMM-TDQN, a two-stage multi-objective reinforcement learning framework for large-scale edge server deployment. The first stage employs a Gaussian Mixture Model (GMM) to capture spatial and workload heterogeneity, enabling an efficient reduction of the deployment search space. Building upon this structured initialization, the second stage formulates the deployment problem as a sequential decision-making task and adopts a Transformer-enhanced Deep Q-Network (TDQN) to learn adaptive deployment policies that balance multiple objectives. Extensive experiments on real-world datasets demonstrate that GMM-TDQN consistently outperforms state-of-the-art methods, achieving reductions of 29.18% in average latency and 17.55% in energy consumption, while improving load balancing by 27.50% and system reliability by 32.55%. These results validate the effectiveness and scalability of the proposed framework for multi-objective edge server deployment.

1 Introduction

Edge computing has emerged as a key paradigm for supporting latency-sensitive and resource-intensive applications in large-scale Internet of Things (IoT) systems, such as intelligent transportation, smart cities, and industrial automation [He *et al.*, 2025]. By offloading computation from centralized cloud servers to geographically distributed edge servers (ESs), edge computing can significantly reduce service latency, alleviate backbone network congestion, and improve system reliability. However, the effectiveness of edge computing critically depends on the deployment strategy of ESs, which determines how computational resources are spatially distributed and dynamically utilized.

Designing an effective ES deployment strategy is challenging for several reasons. First, the deployment problem is inherently large-scale, involving numerous base stations (BSs) with heterogeneous spatial distributions and workload characteristics. Second, the deployment aim is multi-objective, requiring the simultaneous optimization of conflicting objectives, such as service latency, energy consumption, load balance, and system reliability. Improving one objective often degrades another, making the problem difficult to solve using traditional single-objective optimization techniques. Third, the deployment environment is frequently dynamic and uncertain, with time-varying user demands and network conditions, which limit the applicability of static or heuristic-based methods.

Existing approaches to ESs deployment can be broadly categorized into meta-heuristic optimization approaches, clustering and graph-based methods, and learning-based solutions [Wu *et al.*, 2024]. Heuristic and meta-heuristic approaches, while more scalable, usually lack adaptability and offer no guarantees on solution quality. Clustering-based methods typically group BSs or user nodes based on spatial proximity, traffic demand, or service similarity, and deploy ESs at the cluster centers. Recently, reinforcement learning (RL) has gained increasing attention for edge computing problems due to its ability to learn adaptive policies through interaction with the environment. Nevertheless, directly applying RL to large-scale ES deployments remains challenging, as high-dimensional state and action spaces can lead to slow convergence and unstable learning, especially in multi-objective settings.

To address these challenges, we propose GMM-TDQN, a two-stage multi-objective reinforcement learning framework for large-scale ES deployment. The core idea is to decompose the complex deployment problem into two complementary stages that jointly improve scalability and learning efficiency. In the first stage, a Gaussian Mixture Model (GMM) is employed to capture the spatial and workload heterogeneity of base stations, providing a structured initialization that significantly reduces the deployment search space. In the second stage, the deployment problem is formulated

*Corresponding author.

as a sequential decision-making process, and a Transformer-enhanced Deep Q-Network (TDQN) is developed to learn adaptive deployment policies that balance multiple objectives. The Transformer-based attention mechanism enables the model to capture long-range dependencies and complex interactions among deployment decisions, leading to more effective multi-objective optimization.

Extensive experiments conducted on real-world datasets demonstrate that the proposed framework consistently outperforms state-of-the-art methods across multiple performance metrics. The results show significant improvements in latency reduction, energy efficiency, load balancing, and system reliability, validating the effectiveness and scalability of GMM-TDQN in realistic large-scale edge computing environments.

The main contributions of this paper are summarized as follows:

- We formulate large-scale ES deployment as a multi-objective reinforcement learning problem, capturing multiple conflicting performance objectives within a unified learning framework;
- We propose a two-stage deployment strategy: a GMM-based clustering stage that reduces decision complexity and enhances scalability, followed by a Transformer-enhanced DQN (TDQN) that learns adaptive deployment policies;
- We design a TDQN architecture tailored for multi-objective deployment optimization, enabling effective modeling of complex dependencies among deployment decisions;
- We conduct extensive experiments on real-world datasets, demonstrating that the proposed framework significantly outperforms existing approaches across multiple objectives.

2 Related Work

Edge server placement (ESP) has emerged as a critical issue in mobile edge computing (MEC) and smart city networks, aiming to provide low-latency services while optimizing resource utilization and system stability. Prior research in ESP can be broadly categorized into three classes: meta-heuristic optimization approaches, clustering and graph-based methods, and machine learning and reinforcement learning-based approaches.

2.1 Meta-Heuristic Optimization Approaches

Meta-heuristic algorithms have been widely applied to ESP due to its NP-hard nature. Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and their hybrid variants are commonly used to search for near-optimal server placements under complex constraints. Han *et al*. [Han *et al.*, 2024] proposed GP4ESP, a hybrid GA-PSO algorithm that combines evolutionary exploration with fast convergence, achieving a reduction in overall response time compared with baseline approaches. Pandey *et al*. [Pandey *et al.*, 2023] introduced a traffic-aware meta-heuristic ESP scheme to improve load balancing and energy efficiency. Similarly, Sajadinia and Yari [Sajadinia and Yari, 2023] applied multi-objective genetic algorithms combined with clustering for virtual machine placement in cloud environments, demonstrating reduced energy consumption and improved resource utilization.

Recent studies have also focused on robustness and cost-awareness. In Ref. [Liu *et al.*, 2024], an improved snake optimization-based dynamic ESP scheme was introduced , which minimizes service delay and operational cost through an interior-point-assisted optimization strategy. In addition, multi-objective evolutionary algorithms have also been extensively explored for ESP. For instance, an NSGA-II-based edge server placement and allocation framework was proposed in [Ghasemzadeh *et al.*, 2024], where latency and server workload variance were jointly optimized to obtain balanced Pareto-optimal solutions. Although effective in handling conflicting objectives, such approaches rely on population-based search and lack learning capability, which limits their adaptability to highly dynamic MEC environments.

2.2 Clustering and Graph-Based Methods

Clustering and graph-based techniques exploit the spatial and topological characteristics of BSs and ESs to improve placement efficiency. Vali *et al*. [Vali *et al.*, 2024] proposed RESP, a recursive clustering-based ESP approach that places ESs according to cluster medians, achieving balanced workloads and reduced network traffic. Yucel [Yucel, 2021] extended spectral clustering with hierarchical and density-aware enhancements to support collaborative virtual services. In addition, Zhang *et al*. [Zhang *et al.*, 2024] introduced a graph convolutional network (GCN)-based framework combined with differentiable K-Means, transforming ESP into an end-to-end optimization task that jointly minimizes latency and server workload. For application-specific scenarios, Doostmohammadi *et al*. [Doostmohammadi *et al.*, 2023] proposed DyMECC, a dynamic clustering strategy for MEC video streaming based on queuing theory, reducing delivery delay under fluctuating traffic. Qin *et al*. [Qin *et al.*, 2024] studied joint optimization of base station clustering and service caching, demonstrating that spatial clustering plays a critical role in latency and cache cost reduction. However, such methods typically rely on static assumptions and lack adaptability to time-varying workloads.

2.3 Machine Learning and Reinforcement Learning-Based Approaches

Recent studies increasingly leverage machine learning (ML) and RL to capture dynamic network conditions and optimize ESP adaptively. Chen and Guo [Chen and Guo, 2023] proposed an RL-based framework (ESDR) for dynamic MEC environments, achieving notable energy savings through learned deployment policies. Ling *et al*. [Ling *et al.*, 2023] integrated graph convolutional networks to predict user traffic and guide ESP decisions, improving both latency and energy efficiency. Deep reinforcement learning has also been applied to reliability and mobility-aware scenarios. Zhang and Jabbari [Zhang and Jabbari, 2024] employed deep RL for joint content placement and IoT device allocation. Toufga *et al*. [Shinde and Tarchi, 2024] proposed a multi-time-scale

reinforcement learning approach for joint service placement and task offloading, utilizing aerial platforms and advanced deep Q-learning to minimize latency and energy consumption.

3 System Architecture and Problem Formulation

This section explores the system architecture and problem formulation of ESs placement.

3.1 System Architecture

Fig. 1 illustrates a three-layer architecture for ESs deployment, consisting of the user layer, edge layer, and cloud layer . This work focuses on the edge layer, which includes n base stations (BSs) and m ESs, denoted by $B = \{b_1, b_2, \ldots, b_n\}$ and $S = \{s_1, s_2, \ldots, s_m\}$, respectively.

The user layer comprises heterogeneous intelligent devices (e.g., mobile terminals, sensors, and cameras) that continuously generate computation requests [Nguyen *et al.*, 2024]. These requests are first transmitted to the nearest BS and then forwarded to the edge layer for processing.

The edge layer hosts ESs with limited computing resources to serve latency-sensitive tasks locally. When edge resources are insufficient, tasks are offloaded to the cloud layer, which provides abundant computation and storage at the cost of higher transmission delay and energy consumption. Therefore, efficient edge server deployment strategies are critical to achieving low latency, energy efficiency, and reliable service delivery.

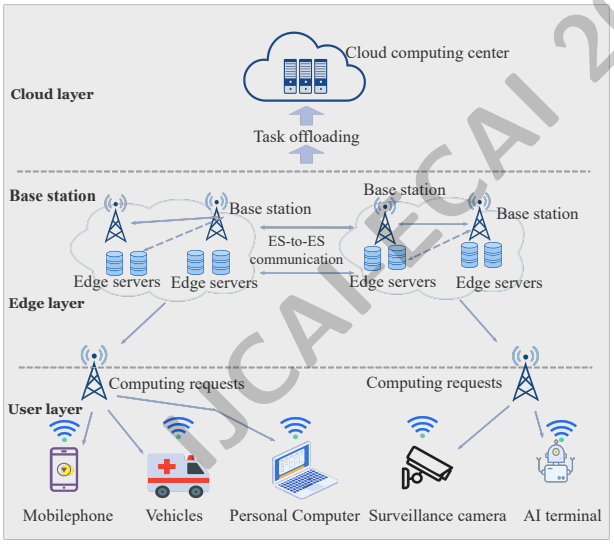


Figure 1: System architecture

3.2 Related Definition and the Optimization Problem

In the ES placement algorithm, four critical performance metrics are identified: delay, energy consumption, load balancing, and network reliability. Each of them is described below.

Delay Metrics

The delay of an ES s_j is defined as the total geographical distance to all BSs it serves:

$$\text{Delay}(s_j) = \sum_{b_i \in B_j} d(b_i, s_j), \tag{1}$$

where B_j is the set of BSs assigned to s_j, and $d(b_i, s_j)$ is the great-circle distance calculated using the Haversine formula:

$$d(b_i, s_j) = 2R\times \arcsin\sqrt{\sin^2\frac{\Delta\text{lat}}{2} + \cos(\text{lat}_i)\cos(\text{lat}_j)\sin^2\frac{\Delta\text{lon}}{2}}. \tag{2}$$

Where R is the Earth's radius, and Δlat, Δlon are the latitude/longitude differences (in radians) between b_i and s_j.

The overall system delay is the sum over all ESs:

$$\text{Delay}(s_j) = \sum_{b_i \in B_j} d(b_i, s_j). \tag{3}$$

Load Balancing Metrics

The load on ES s_j is defined as the sum of its associated BSs' workloads:

$$W(s_j) = \sum_{b_i \in B_j} w_i, \tag{4}$$

where B_j is the set of BSs served by s_j, and w_i is the workload of b_i.

The load balancing performance is measured by the coefficient of variation of ES loads:

$$Balance(S) = \frac{\sigma_L}{\mu_L}, \tag{5}$$

with σ_L and μ_L being the standard deviation and mean of the ES loads, respectively.

Energy Consumption Metrics

The energy consumption of an ES s_j is determined by its power profile and CPU utilization:

$$Energy(s_j) = \int_{t_1}^{t_2} \left[P_{\text{idle}} + (P_{\max} - P_{\text{idle}}) \cdot U_P(t)\right] dt, \tag{6}$$

where P_{idle} and $P_{\max}$ denote the idle and maximum power, and $U_P(t)$ is the instantaneous CPU utilization.

CPU utilization is modeled as a piecewise function based on the server load $W(s)$ and a capacity threshold W_{TH}:

$$U_P(s) = \begin{cases} \dfrac{W(s)}{W_{\text{TH}}}, & W(s) < W_{\text{TH}} \\ 1, & W(s) \geq W_{\text{TH}}. \end{cases} \tag{7}$$

The total energy consumption of the system is:

$$Energy(S) = \sum_{s_j \in S} Energy(s_j). \tag{8}$$

Network Reliability

Network reliability quantifies the system's ability to maintain service under failures. Redundancy is enhanced when BSs are covered by multiple ESs.

Define a binary indicator C_{ji} for whether ES s_j can serve base station b_i within its coverage radius R_e:

$$C_{ji} = \begin{cases} 1, & d(b_i, s_j) \le R_e, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The normalized reliability $R(S)$ is the average coverage per BS:

$$R(S) = \frac{1}{|S|\,|B|} \sum_{j=1}^{|S|} \sum_{i=1}^{|B|} C_{ji}. \tag{10}$$

A value $R(S) \in [0, 1]$ close to one indicates high redundancy and fault tolerance.

3.3 Problem Formulation

Given the definitions in Section 3.2, the ESs deployment problem aims to determine the optimal placement of m ESs over n BSs. Each BS is associated with its nearest ES based on the Haversine distance, and each ES serves the traffic demands of its assigned BSs.

The problem is formulated as a multi-objective optimization by jointly considering system delay, energy consumption, load balancing, and network reliability:

$$\begin{aligned} &\textbf{Minimize: } F(S) = \lambda_1 \times \mathrm{Delay}(S) + \\ &\lambda_2 \times \mathrm{Energy}(S) + \lambda_3 \times \mathrm{Balance}(S) - \lambda_4 \times R(S) \end{aligned} \tag{11}$$

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1. \tag{12}$$

The optimization is subject to the following constraints:

$$U_P \le 1, \quad W(s_i) \le W_{\mathrm{TH}},\ \forall s_i \in S, \tag{13}$$

$$\sum_{s_j \in S} a_{ij} = 1, \quad a_{ij} \in \{0, 1\},\ \forall b_i \in B, \tag{14}$$

$$b_i \to \arg\min_{s_j \in S} d(b_i, s_j), \tag{15}$$

$$|W(s_i) - \bar{W}| \le \delta,\ \forall s_i \in S, \tag{16}$$

$$d(b_i, s_j) \le d_{\max}, \quad E(s_i) \le E_{\max}, \tag{17}$$

$$R(S) \ge R_{\min}. \tag{18}$$

These constraints ensure exclusive BS–ES association, bounded resource utilization, balanced workload distribution, limited communication distance, energy feasibility, and minimum network reliability. Due to its combinatorial and non-convex nature, the formulated ES deployment problem is NP-hard.

To address this challenge, a hybrid **GMM–TDQN** framework is adopted, where GMM provides spatial initialization and a TDQN refines ES placement through reinforcement learning.

4 GMM–TDQN Algorithm

In this section, we introduce the ESs placement algorithm GMM–TDQN.

4.1 The Overall Process of the GMM–TDQN

To address the ESs placement problem under complex spatial distributions and service demands, we propose a two-stage optimization framework termed GMM–TDQN, as illustrated in Fig. 2. The framework integrates GMM clustering for deployment initialization with a TDQN for adaptive refinement [K *et al.*, 2024].

In the first stage, a GMM is employed to initialize ES locations by clustering BSs according to their spatial distribution and traffic load. After convergence of the EM algorithm, each BS is assigned to its most probable Gaussian component, and the centroid of each cluster is selected as the initial ES position. This procedure places ESs in dense and high-demand regions, providing a favorable starting point for optimization.

In the second stage, the TDQN iteratively refines ES positions through reinforcement learning. Agents interact with the environment under an ε-greedy policy, and transition samples are stored in a replay buffer. The Transformer-based Q-network is updated using mini-batch training to capture global spatial dependencies among ESs and improve deployment decisions.

By combining probabilistic clustering, global attention modeling, and value-based reinforcement learning, the proposed GMM-TDQN framework enables efficient initialization and adaptive optimization of ES deployment. Detailed descriptions of the GMM and TDQN components are provided in Sections 4.2 and 4.3, respectively.

4.2 Initialization Using GMM

To provide a principled initialization for ES deployment, a GMM is adopted to capture the spatial distribution of BSs. Each BS $b_i \in B$ is characterized by its geographical coordinates and traffic load, and the overall BS distribution is modeled as a mixture of m Gaussian components, where m corresponds to the number of ESs.

The probability density of BS b_i is given by

$$p(b_i \mid \Theta) = \sum_{k=1}^{m} \pi_k \, \mathcal{N}(b_i \mid \mu_k, \Sigma_k), \tag{19}$$

where π_k, μ_k, and Σ_k denote the mixture weight, mean, and covariance of the k-th component, respectively, and Θ represents the model parameters. In this work, the feature dimension is set to $d = 3$, incorporating spatial coordinates and traffic load.

The parameters are estimated via the Expectation–Maximization (EM) algorithm by maximizing the log-likelihood

$$\mathcal{L}(\Theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{m} \pi_k \mathcal{N}(b_i \mid \mu_k, \Sigma_k) \right). \tag{20}$$

After convergence, each BS is assigned to the Gaussian component with the highest posterior responsibility.

Server Initialization

Based on the clustering results, the initial position of ES s_k is determined as the centroid of BSs belonging to cluster B_k:

$$s_k^{(0)} = \frac{1}{|B_k|} \sum_{b_i \in B_k} b_i. \tag{21}$$

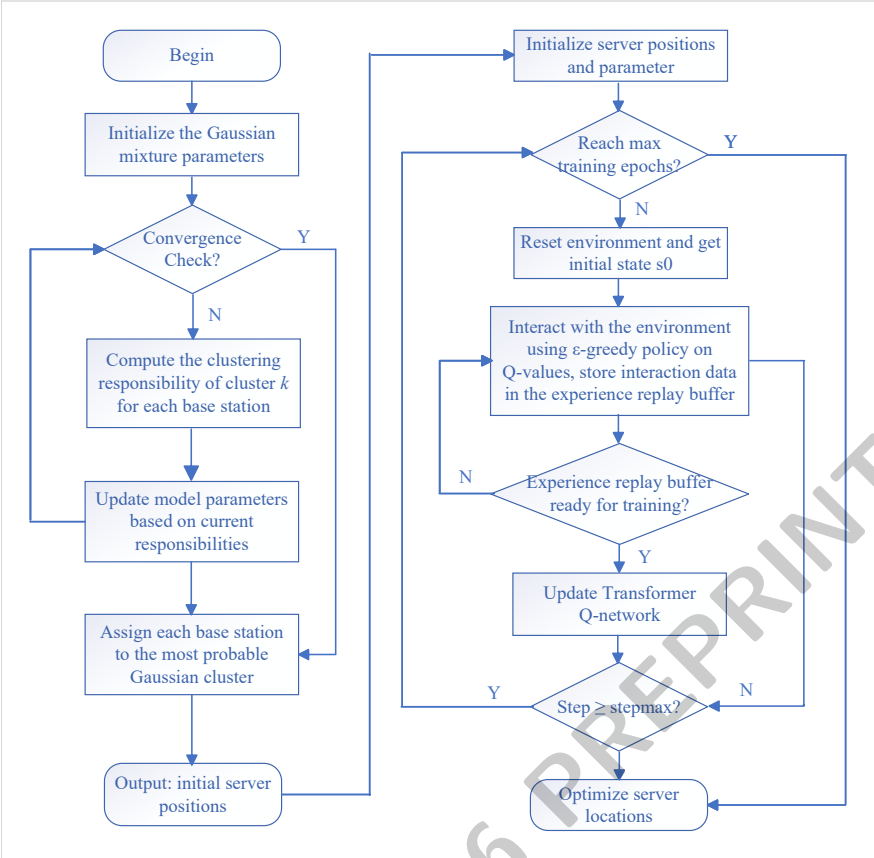


Figure 2: Overall process of the GMM–TDQN framework

This initialization places ESs close to dense and high-demand regions, providing a favorable starting point for reducing latency and energy consumption while improving load balance and network reliability. The subsequent reinforcement learning module further refines the deployment to optimize the overall objective.

Algorithm 1 summarizes the GMM-based ES initialization procedure.

Algorithm 1 Initialization of ES deployment based on GMM

Input: Base station set B
Output: Initial ES positions $S^{(0)}$

Randomly initialize μ_k, Σ_k, π_k;
Set iteration counter $t = 0$ and log-likelihood $\mathcal{L}^{(0)} = -\infty$;
while not converged **do**
    $t \leftarrow t + 1$;
    Compute the responsibility of component k for each base station b_i;
    Update model parameters based on current responsibilities μ_k, Σ_k, π_k;
    Compute the updated log-likelihood $\mathcal{L}^{(t)}$;
    Stop if $\left|\mathcal{L}^{(t)} - \mathcal{L}^{(t-1)}\right| < \xi$;
end while
Assign each base station b_i to the most probable Gaussian component B_k;
Compute the initial position of each ES as the centroid of its assigned cluster $s_k^{(0)}$;
return $S^{(0)} = \{s_1^{(0)}, s_2^{(0)}, \ldots, s_m^{(0)}\}$.

4.3 Optimal ES Deployment Based on the TDQN Algorithm

To further enhance the ES deployment optimization, we propose a TDQN algorithm that leverages the self-attention mechanism to capture complex dependencies among ES. This section introduces the details of the ES environment modeling and the TDQN algorithm for ES deployment.

Environment Modeling for ES Deployment

The ES deployment environment is modeled as a grid-based system with the following components:

State Space: The state representation for each ES includes its normalized coordinates, nearby BS statistics, and load information. Formally, the state for ES i is represented as:

$$o_i = \left[\frac{lon_i}{G_{\max}}, \frac{lat_i}{G_{\max}}, \frac{N_{\text{nearby}}}{N_{\text{total}}}, \frac{\bar{d}}{D_{\max}}, \frac{\bar{W}_{nearby}}{W_{\text{total}}}\right]$$

where lon_i and lat_i denote the spatial coordinates of ES i, $G_{\max}$ is the maximum grid size of the deployment area, N_i^{near} and N_{total} represent the number of nearby BSs associ-

Algorithm 2 TDQN for ES deployment

Input: Initial ES positions from Algorithm 1
Output: Final ES deployment positions $\mathbf{S}^*$

Initialize TDQN parameters θ and target network θ^-;
Initialize replay buffer $\mathcal{D}$ and environment;
for episode $e = 1$ to E **do**
  Reset environment, obtain initial global state o^0, construct embedding X_0;
  for time step $t = 0$ to $T_{\max} - 1$ **do**
    Encode X_t to Z_t^L via Transformer;
    For each ES i, compute Q-values $Q_i(o_i, a_i; \theta)$ for $a_i \in \mathcal{A}$;
    Select actions for all ESs using ϵ-greedy;
    Execute joint action $\mathbf{a}^t = (a_1^t, \ldots, a_m^t)$, update ES positions;
    Observe reward r and next state o';
    Store transition (o, a, r, o') in buffer $\mathcal{D}$;
    if buffer $\mathcal{D}$ contains at least B samples **then**
      Sample mini-batch $\{(o, a, r, o')\}$ of size B;
      Compute current Q-values $Q_i(o_i, a_i; \theta)$ for all ESs i;
      Compute TD targets according to Eq.(24);
      Update θ by minimizing DQN loss according to (23);
      Every C steps update target parameters $\theta^- \leftarrow \theta$;
    end if
  end for
end for
Apply the learned optimal policy $\pi^*(o)$ to determine the final ES positions.
return Final ES deployment positions $\mathbf{S}^*$

ated with ES i and the total number of BSs in the system, $\bar{d}_i$ is the average Euclidean distance between ES i and its nearby BSs, $D_{\max}$ denotes the maximum possible distance within the considered region, $\bar{W}^{\text{nearby}}$ is the mean traffic load of the nearby BSs, and W_{total} denotes the aggregated load of all BSs.

Action Space: Each ES can perform one of five possible actions:

$$A = \{\text{Up}, \text{Down}, \text{Left}, \text{Right}, \text{Stay}\}$$

Reward Function: The reward function incorporates multiple optimization objectives, including latency, energy consumption, load balancing, and network reliability:

$$\begin{aligned} Reward &= \lambda_1 \times \text{Delay}(S) + \\ &\lambda_2 \times \text{Energy}(S) + \lambda_3 \times \text{Balance}(S) - \lambda_4 \times R(S) \end{aligned} \tag{22}$$

where λ_1 to λ_4 are weighting coefficients that balance the importance of each objective.

ES Deployment Based on TDQN

To enhance the representation capability of conventional DQN, a Transformer encoder is integrated to model spatial correlations and interaction dependencies among multiple ESs, as illustrated in Fig. 3. Unlike CNN- or MLP-based

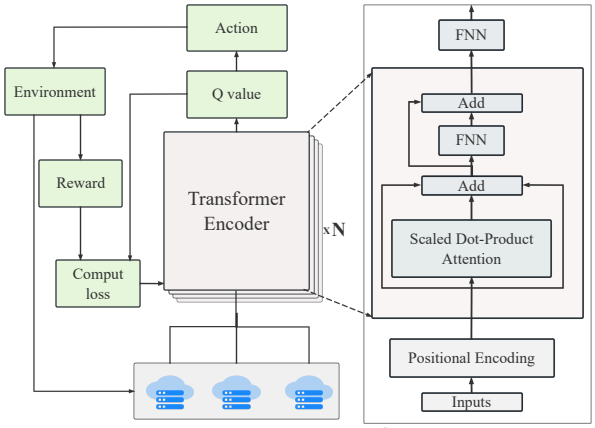


Figure 3: TDQN architecture

architectures, the proposed TDQN explicitly captures global deployment patterns through self-attention.

Each ES i is represented by an embedding vector $x_i \in R^d$, constructed from normalized spatial coordinates, local BS statistics, and load indicators. The embedding sequence is denoted as

$$X = [x_1, x_2, \ldots, x_m]^\top \in R^{m \times d},$$

which is projected to the model dimension and combined with positional encodings:

$$H_0 = XW_E + P.$$

The Transformer encoder consists of L stacked self-attention layers, enabling each ES to aggregate information from all other ESs and generate a context-aware representation $Z_L \in R^{m \times d_{\text{model}}}$. This representation encodes global spatial dependencies and workload interactions among ESs.

The encoded features are fed into a linear Q-head to produce per-server Q-values over a discrete action set

$$\mathcal{A} = \{\text{Up}, \text{Down}, \text{Left}, \text{Right}, \text{Stay}\}.$$

Let $Q_i(o_i, a_i)$ denote the Q-value of ES i. The TDQN is trained using the standard DQN loss:

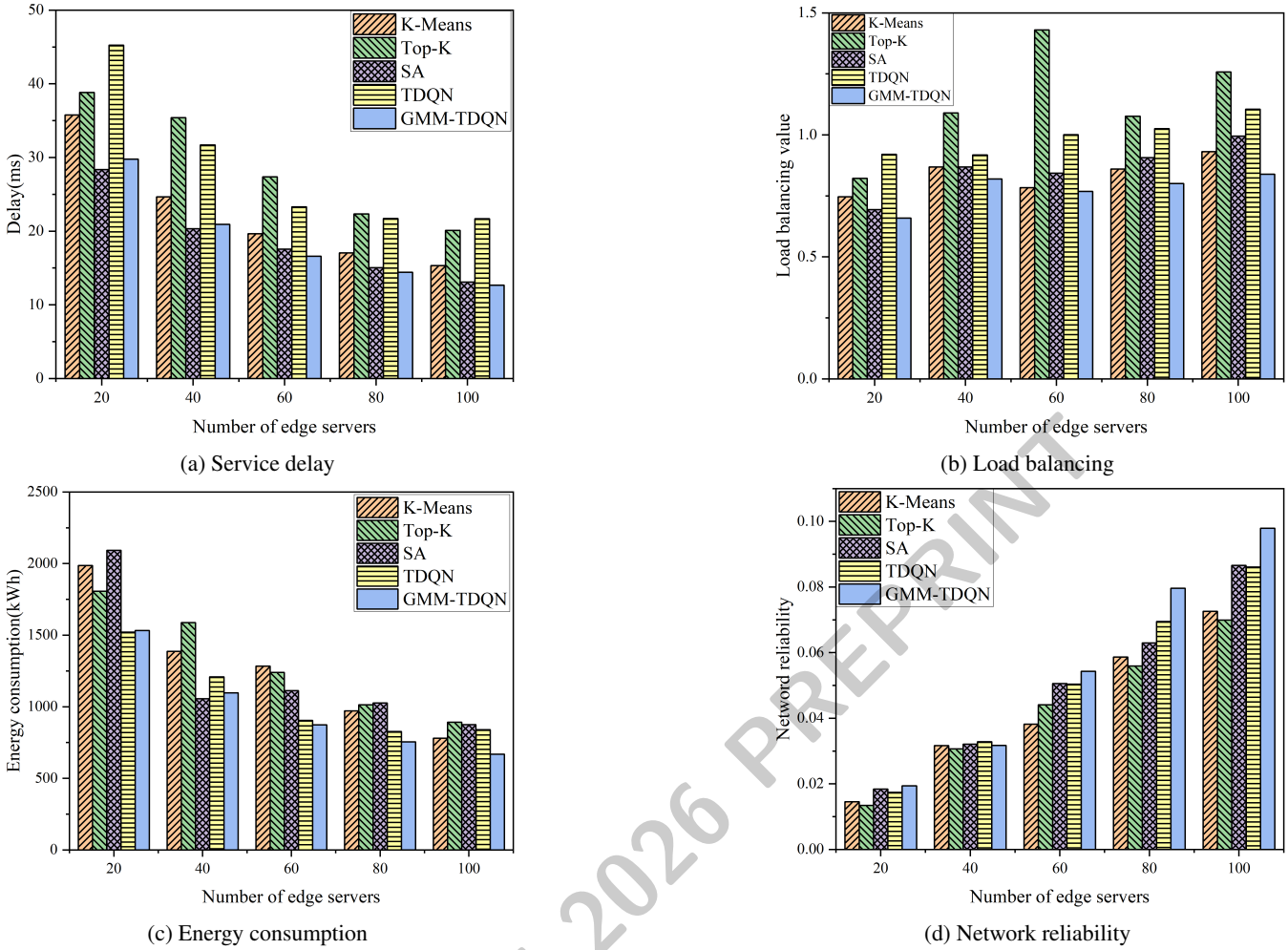
$$\mathcal{L}_{\text{DQN}} = E\left[\frac{1}{m}\sum_{i=1}^{m}\big(y_i - Q_i(o_i, a_i)\big)^2\right], \tag{23}$$

where the temporal-difference target is defined as

$$y_i = r + \gamma(1 - \text{done}) \max_{a_i'} Q_i'(o_i', a_i'). \tag{24}$$

During training, each ES selects actions following an ϵ-greedy policy, and all ESs execute their actions simultaneously to update deployment positions. A replay buffer is used to store transition tuples (o, a, r, o'), and a target network is periodically synchronized to stabilize training.

By continuously interacting with the environment, the TDQN learns an optimal deployment policy $\pi^*(o)$ that adaptively adjusts ES locations to jointly reduce latency, energy consumption, and load imbalance while maintaining network reliability. Benefiting from the self-attention mechanism, the



(a) Service delay

(b) Load balancing

(c) Energy consumption

(d) Network reliability

Figure 4: Performance comparison of ES deployment strategies with different numbers

proposed method enables coordinated server movements and achieves faster convergence and more balanced deployment compared with conventional DQN-based approaches.

Algorithm 2 summarizes the pseudocode of the TDQN algorithm.

Base Station ID	Latitude	Longitude	Load/min
1292	31.212450	121.637072	54600
2386	31.306850	121.519579	46080
952	31.177646	121.555355	105840
615	31.103599	121.416145	7620
2574	31.362367	121.461011	7020

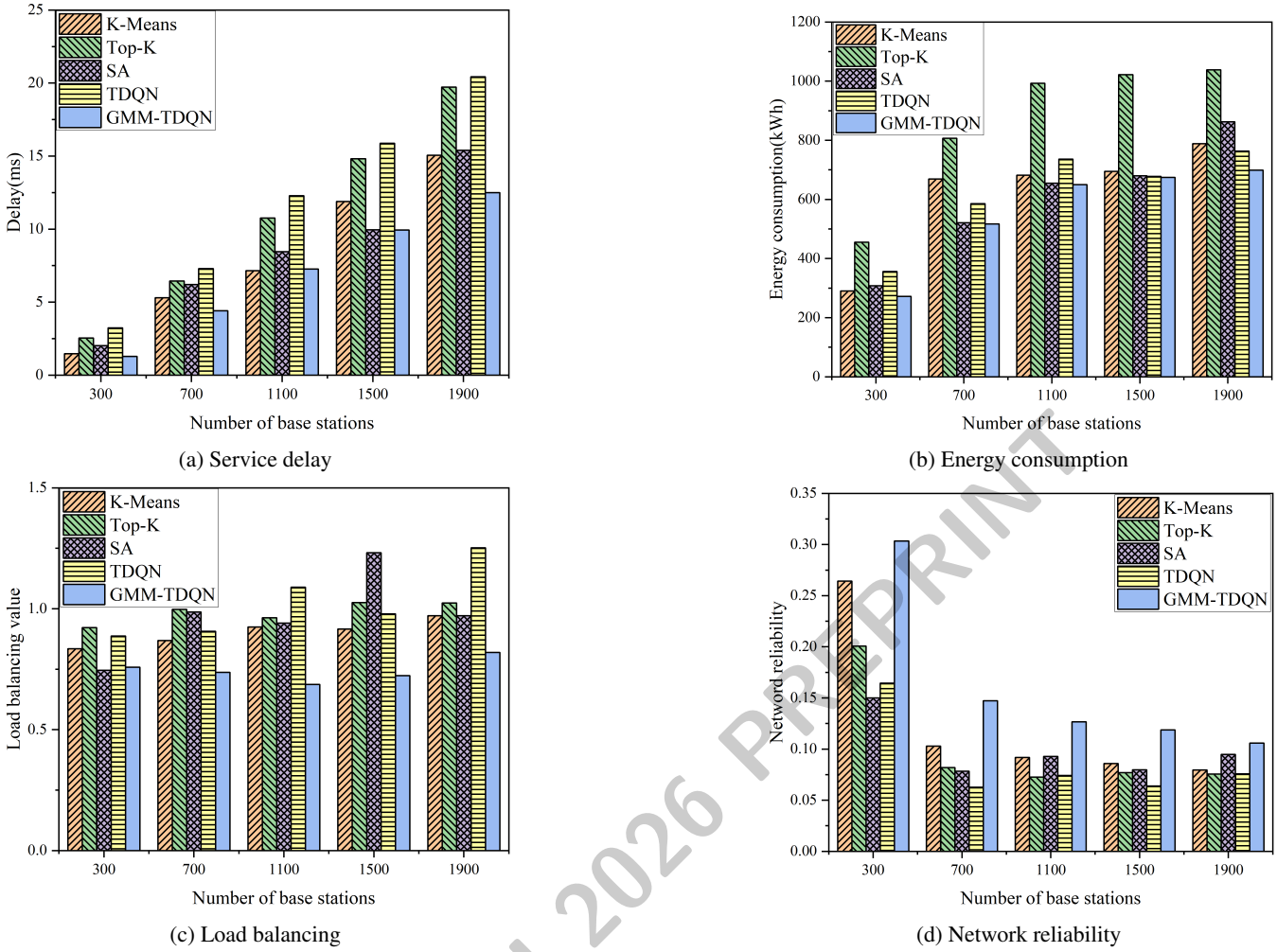
Table 1: Partial base station information

Parameter	**Value**
Replay Buffer Capacity	2,000
Batch Size	64
Learning Rate	1e-4
Discount Factor (γ)	0.99
Exploration Decay Rate	0.995
Training Episodes	100
Steps per Episode	20
Latency Weight	0.3
Energy Consumption Weight	0.1
Load Balancing Weight	0.4
Reliability Weight	0.2
Idle Power Consumption (P_{idle})	0.3 W
Maximum Power Consumption (P_{max})	0.5 W
Threshold Workload (W_{TH})	3×10^5 min

Table 2: Experimental parameters and configurations

5 Experimental Results and Analysis

To comprehensively evaluate the performance of the GMM-TDQN approach, we compare the GMM–TDQN algorithm against several representative baseline methods spanning diverse optimization paradigms. These baseline algorithms include K-Means [Huang *et al.*, 2025], Top-K [Chen and Xiao, 2025], Simulated Annealing (SA) [Lan *et al.*, 2025], and TDQN (without GMM Initialization).



(a) Service delay

(b) Energy consumption

(c) Load balancing

(d) Network reliability

Figure 5: Performance comparison under different conditions

5.1 Experimental Parameters and Configuration

This study utilizes a real-world network dataset provided by the Shanghai Telecommunications Bureau. The processed dataset comprises records for 2,753 base stations, including their unique identifiers, latitude and longitude coordinates, and access durations over a continuous 15-day period, as shown in Table 1 and Table 2.

5.2 Experimental Comparison and Analysis

Performance Comparison under Different Numbers of ESs

In this experiment, the number of base stations is fixed at 2753, while the number of ESs increases from 20 to 100. The five deployment schemes are evaluated in terms of system latency, load balancing, energy consumption, and network reliability, as illustrated in Fig. 4(a)–4(d).

The experimental results consistently demonstrate that GMM-TDQN significantly outperforms the four baseline strategies across all evaluation metrics. Its superiority becomes increasingly pronounced as system scale grows. This performance advantage stems from GMM-TDQN's multi-objective decision mechanism, which jointly optimizes latency, energy usage, load distribution, and reliability within a unified learning framework. In contrast, clustering-based K-Means and heuristic SA methods are limited by their static objectives, while the TDQN and Top-K strategies lack the capacity to reason about spatial–temporal patterns or global resource coordination. These findings underscore the necessity of intelligent and adaptive optimization methods for large-scale edge server deployment in complex and dynamic environments.

Performance Comparison under Different Numbers of BSs

To comprehensively evaluate the performance of the proposed GMM-TDQN algorithm, we compare it with four baseline methods, including K-Means, Top-K, SA, and TDQN deployment, under different numbers of base stations (300–1900). The evaluation metrics include average latency, energy consumption, load balancing, and system reliability.

As shown in Fig. 5(a)–5(d), experimental results con-

firm that the GMM-TDQN algorithm exhibits better scalability and robustness compared with traditional clustering- and heuristic-based methods. It dynamically adapts to the increase in network density while optimizing multiple objectives, demonstrating its potential for practical deployment in large-scale edge computing environments.

6 Conclusion

We investigated the problem of large-scale ESs deployment under multi-objective optimization requirements. To address the challenges posed by scalability, conflicting objectives, and dynamic environments, we proposed GMM-TDQN, a two-stage multi-objective reinforcement learning framework. Experimental results on real-world datasets demonstrate that the proposed approach consistently outperforms state-of-the-art methods, highlighting its effectiveness and scalability in realistic edge computing scenarios.

Future work will explore extending the proposed framework to more dynamic settings, such as online deployment with time-varying workloads and mobile edge nodes.

Ethical Statement

There are no ethical issues.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (grant 62402065), Humanities and Social Sciences Research Planning Fund of the Ministry of Education of China (grant 25YJA710076 and 24YJC710107), the Hunan Provincial Education Science Research Project during the 14th Five-Year Plan period (grant XJK23CXX004), the Hunan Provincial Degree and Postgraduate Teaching Reform Research Project (grant 2025JGYB360), the Natural Science Foundation of Hunan Province (grant 2025JJ50344), the Natural Science Foundation of Changsha city (grant kq2502340).

References

[Chen and Guo, 2023] Cen Chen and Yingya Guo. Energy-aware edge server placement in dynamic scenario using reinforcement learning. In *2023 15th International Conference on Communication Software and Networks (ICCSN)*, pages 183–187, 2023.

[Chen and Xiao, 2025] Chaoyu Chen and Yanshan Xiao. Maximum margin clustering with top-k loss. In *2025 IEEE 15th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 102–106, 2025.

[Doostmohammadi *et al.*, 2023] Ali Doostmohammadi, Mohammad Reza Khayyambashi, Naser Movahedinia, and Zdenek Becvar. Dynamic clustering for low-delay delivery of video content cached in mec servers. *IEEE Systems Journal*, 17(4):5842–5853, 2023.

[Ghasemzadeh *et al.*, 2024] Ardalan Ghasemzadeh, Hadi S. Aghdasi, and Saeed Saeedvand. Edge server placement and allocation optimization: a tradeoff for enhanced performance. *Cluster Computing*, 27(5):5783–5797, August 2024.

[Han *et al.*, 2024] Fang Han, Hui Fu, Bo Wang, Yaoli Xu, and Bin Lv. Gp4esp: a hybrid genetic algorithm and particle swarm optimization algorithm for edge server placement. *PeerJ Computer Science*, 10:e2439, 2024.

[He *et al.*, 2025] Qiang He, Quanwei Li, Chuangchuang Zhang, Xingwei Wang, Yuanguo Bi, Liang Zhao, Ammar Hawbani, and Keping Yu. Task optimization allocation in vehicle based edge computing systems with deep reinforcement learning. *IEEE Transactions on Computers*, 74(9):3156–3167, 2025.

[Huang *et al.*, 2025] Zhijia Huang, Kefan Cai, and Jinzi Li. Research on intelligent water management detection based on the k-means clustering algorithm. In *2025 International Conference on Electrical Drives, Power Electronics Engineering (EDPEE)*, pages 108–113, 2025.

[K *et al.*, 2024] Uthej K, Kothuru Gurunadh, Nagandla Krishna Sai Keerthan, Nikhil Kumar Musunuru, and Amudha J. Energy management system - deep reinforcement learning based dueling dqn (deep q-networks). In *2024 Asia Pacific Conference on Innovation in Technology (APCIT)*, pages 1–7, 2024.

[Lan *et al.*, 2025] Lihua Lan, Hanping Huang, Fupei Ning, Yanbin Zheng, and Qin Yang. Research on dual-objective robust optimization based on monte carlo simulation and simulated annealing algorithm. In *2025 International Conference on Electrical Drives, Power Electronics Engineering (EDPEE)*, pages 978–982, 2025.

[Ling *et al.*, 2023] Chen Ling, Zebang Feng, Liyan Xu, Qian Huang, Yinsheng Zhou, Weizhe Zhang, and Rahul Yadav. An edge server placement algorithm based on graph convolution network. *IEEE Transactions on Vehicular Technology*, 72(4):5224–5239, 2023.

[Liu *et al.*, 2024] Jinjin Liu, Xiaofeng Wu, and Peiyan Yuan. A dynamic edge server placement scheme using the improved snake optimization algorithm. *Applied Sciences*, 14(22), 2024.

[Nguyen *et al.*, 2024] Truong Thanh Hung Nguyen, Vo Thanh Khang Nguyen, Quoc Hung Cao, Van Binh Truong, Quoc Khanh Nguyen, and Hung Cao. Enhancing the fairness and performance of edge cameras with explainable ai. In *2024 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–4, 2024.

[Pandey *et al.*, 2023] Chandrasen Pandey, Vaibhav Tiwari, Sambit Pattanaik, and Diptendu Sinha Roy. A strategic metaheuristic edge server placement scheme for energy saving in smart city. In *2023 International Conference on Artificial Intelligence and Smart Communication (AISC)*, pages 288–292, 2023.

[Qin *et al.*, 2024] Langtian Qin, Hancheng Lu, Yao Lu, Chenwu Zhang, and Feng Wu. Joint optimization of base station clustering and service caching in user-centric mec. *IEEE Transactions on Mobile Computing*, 23(5):6455–6469, 2024.

[Sajadinia and Yari, 2023] Alireza Sajadinia and Alireza Yari. Virtual machine placement strategy using clustering

and genetic algorithm for increasing cloud performance and power saving. In *2023 28th International Computer Conference, Computer Society of Iran (CSICC)*, pages 1–5, 2023.

[Shinde and Tarchi, 2024] Swapnil Sadashiv Shinde and Daniele Tarchi. Multi-time-scale markov decision process for joint service placement, network selection, and computation offloading in aerial iov scenarios. *IEEE Transactions on Network Science and Engineering*, 11(6):5364–5379, 2024.

[Vali *et al.*, 2024] Ali Akbar Vali, Sadoon Azizi, and Mohammad Shojafar. Resp: A recursive clustering approach for edge server placement in mobile edge computing. *ACM Trans. Internet Technol.*, 24(3), jul 2024.

[Wu *et al.*, 2024] Qiong Wu, Wenhua Wang, Pingyi Fan, Qiang Fan, Jiangzhou Wang, and Khaled B Letaief. Urllc-awared resource allocation for heterogeneous vehicular edge computing. *IEEE Transactions on Vehicular Technology*, 73(8):11789–11805, 2024.

[Yucel, 2021] Sakir Yucel. Hierarchical and density aware spectral clustering approach to server placement for collaborative virtual services. In *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1805–1810, 2021.

[Zhang and Jabbari, 2024] Liang Zhang and Bijan Jabbari. Machine learning for caching placement in edge computing networks. In *2024 International Conference on Computing, Networking and Communications (ICNC)*, pages 259–264, 2024.

[Zhang *et al.*, 2024] Shanshan Zhang, Jiong Yu, and Mingjian Hu. An edge server placement based on graph clustering in mobile edge computing. *Scientific Reports*, 14(1):29986, 2024.