DOI 10.15622/ia.25.1.8

I. FILIANIN, A. KAPITONOV, A. TIMOSHCHUK- BONDAR

RESEARCH ON REINFORCEMENT LEARNING ALGORITHMS FOR NETWORK LATENCY REDUCTION IN EDGE COMPUTING

Filianin I., Kapitonov A., Timoshchuk-Bondar A. **Research on Reinforcement Learning Algorithms for Network Latency Reduction in Edge Computing.**

Abstract. Current research on decision-making algorithms in multi-access edge computing (MEC) for resource allocation often relies on simplified network topology abstractions, which limits the applicability of the results in real-world mobile network operations. This work aims to develop a realistic cellular network model using stochastic geometry methods and to comprehensively evaluate the effectiveness of modern reinforcement learning algorithms in minimizing network latency in edge computing. To create a mathematically sound model of the network environment, we used stochastic geometry methods combined with real statistical data on cellular user distribution. Applying stochastic geometry ensured accurate modeling of the spatial placement of base stations and the calculation of inter-node distances, which are critically important for determining network latency. Experimental evaluation was conducted on a refined Lightweight MEC Platform Simulator (LWMECPS) platform with an extended Gymnasium API, supporting Proximal Policy Optimization (PPO), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC) algorithms. We developed a communication network model that considers the realistic spatial distribution of network elements and the temporal dynamics of user load. Based on this model, a virtualized test environment was created in LWMECPS, allowing for reproducible experiments with controllable parameters. Experimental results revealed distinct performance characteristics across the algorithms: PPO achieved a consistent latency reduction of up to 20% with stable convergence; SAC demonstrated the highest absolute improvement (a latency reduction of 38%) but exhibited initialization instability; TD3 showed moderate effectiveness (an improvement of up to 11%) but high sensitivity to hyperparameter tuning. The comparative analysis of reinforcement learning algorithms revealed key features of their application in MEC systems. The discrete nature of service placement tasks makes PPO the most suitable for practical implementation due to its convergence stability and natural support for discrete action spaces, despite SAC achieving higher peak performance. While SAC's superior absolute results are promising, its initialization challenges and original design for continuous action spaces require additional consideration for MEC deployment. The obtained results provide scientifically sound recommendations for MEC platform developers regarding the selection of optimal algorithmic solutions based on specific system requirements and constraints.

Keywords: reinforcement learning, multi-access edge computing (MEC), proximal policy optimization (PPO), soft actor-critic (SAC), twin delayed deep deterministic policy gradient (TD3), LWMECPS, Weights & Biases (WandB).

1. Introduction. Mobile Edge Computing (MEC) represents a distributed computing paradigm that brings data processing and service provision closer to end users, thereby minimizing data transmission delays and reducing the load on central cloud infrastructures [1]. The task of placing computing services in an MEC environment is a multi-criteria optimization problem, whose main objectives are:

– minimization of network latency [2];

– optimization of computational and network resource utilization [3];
– adaptation to data consumption conditions [4];
– energy efficiency [5].

In the first half of 2025, approximately 3,850 scientific articles indexed in Google Scholar discussed algorithms used in decision-making systems for MEC platforms. The core of these decision-making systems for placing computing services in MEC platforms, thus enabling data processing closer to end users, relies on various methods that can be broadly categorized as follows:

– optimization algorithms – they include exact and approximate methods for finding the best solutions in the service placement space;

– reinforcement learning algorithms – used for adaptive selection of placement strategies in dynamically changing environment conditions;

– statistical and heuristic methods – covering a wide range of approaches, including greedy algorithms, dynamic programming methods, and other heuristics that ensure acceptable solution quality with limited computational resources.

The choice of a specific decision-making method depends on the network representation model with MEC architecture. For instance, in the article [6], the communication network is presented in a simplified form as a set of computing nodes and tasks, without considering network parameters and inter-service interactions.

Meanwhile, the works [7] demonstrate the importance of considering multipath transmissions and traffic management in switching nodes to ensure timely packet delivery, which is critically important for minimizing latency in MEC systems.

The article [8] addresses the issue of determining the optimal placement of optical fiber for wave division multiplexing and passive optical network technologies, which enables the reduction of network latency and raises particular questions about the technology's practical applicability.

The article [9] represents the communication network as an RAN cascaded with a computer-server network, with stochastic geometry methods to calculate user and tower placement, which enables the calculation of the Signal-to-Interference Ratio (SIR) and most network communication abstractions.

After analyzing the articles, we can conclude that most of them attempt to represent the communication network mathematically. However, it is worth noting that the radio channel can use various communication standards, such as 4G, 5G, and subsequent generations. These standards, despite their general conceptual direction, are not interchangeable. Furthermore, the implementation of standards varies significantly depending on the equipment manufacturer. Even within the same manufacturer and device series, different hardware

platforms and specialized software can be used, which directly affects the characteristics of data transmission and processing in an MEC system. This, in turn, can significantly impact the quality of decisions made in MEC platforms.

The key optimization criteria when placing computing services are:

– minimizing network latency;

– reducing the cost of deploying and operating services (FinOps approaches);

– reducing the carbon footprint;

– improving user perception of quality of experience (QoE).

These criteria correlate with the parameters used to evaluate the effectiveness of the computational approaches discussed earlier. The scientific literature contains many works aimed at optimizing each of these criteria individually or in combination.

The purpose of this work was to develop a system for training and testing reinforcement learning algorithms in tasks of reducing the average network latency under conditions similar to real-world scenarios.

The objectives defined within this work are as follows:

– analyze the literature and select reinforcement learning algorithms for the decision-making system;

– refine LWMECPS with a gymnasium capability API to support new reinforcement learning algorithms;

– develop a test application that can generate loads based on specified parameters to conduct an experiment;

– conduct an experiment and evaluate reinforcement learning algorithms in a system that approximates a real-world environment.

As a result of this work, we obtained data on the influence of the network model on the performance of reinforcement learning algorithms in the task of placing computing services while reducing the average network latency.

2. Literature Review. The use of reinforcement learning algorithms in decision-making tasks for placing computing services is very popular. However, there are also algorithmic solutions, such as the Lyapunov function discussed in the article [10].

The problem with algorithmic solutions [11] is their strict dependence on the network model, which is rather difficult to describe in the context of real-world communication networks. The advantage of reinforcement learning algorithms is their independence from a specific network model, which can, in turn, accelerate the process of implementing decision-making systems in real cellular networks.

Developing solutions based on RL algorithms is accompanied by difficulties in organizing the test environment used for training models and

prototyping specific solutions [12]. Previously, as a solution to this problem, a Gymnasium capability API interface was developed within the LightWeight Multi Access Edge Computing Platform Simulator, allowing for the rapid organization of an interface for interaction with the target system [13].

Unfortunately, it only supported Q-Network and Deep Q-Network algorithms in a limited representation. Recently, several new reinforcement learning algorithms have emerged that can improve the quality of decisions made.

In particular, new-generation algorithms such as Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and Twin Delayed Deep Deterministic Policy Gradient (TD3) address specific problems of dynamic MEC environments and offer significant improvements over traditional Q-learning-based approaches.

These algorithms were developed with the unique characteristics of resource management tasks in edge computing in mind: high variability of environment states, the need for continuous learning under changing load patterns, and requirements for the stability and predictability of decisions when deployed in real-world operations.

Each algorithm is examined below.

The SAC algorithm is an off-policy actor-critic method with an entropy regularizer that trains a stochastic policy, automatically balancing between optimal convergence and exploration of the action space [14]. A key feature of SAC is the inclusion of an entropy term in the objective function, which promotes diversity in action selection and prevents premature convergence to suboptimal solutions. This approach ensures the most stable convergence under noisy reward signals, which is typical for real MEC systems.

In the context of service placement tasks, SAC demonstrates an ability to discover rare and non-standard placement solutions due to its tendency to explore unusual actions. It is especially valuable in scenarios where clusters of edge nodes are subject to abrupt changes, for example, during traffic spikes, unexpected resource constraints on nodes, or sudden equipment failures. The stochastic nature of the SAC policy enables the system to adapt to unforeseen situations by identifying alternative service placement paths overlooked by deterministic approaches.

The TD3 algorithm is an improved version of the DDPG algorithm – an off-policy deterministic actor-critic method. It includes three key improvements aimed at increasing training stability and the quality of the resulting solutions [15]: the first improvement is the use of two Q-networks («twin» Q-networks), which helps to mitigate the problem of overestimating action values typical of critic-based methods; the second

improvement is delayed actor updates performed less frequently than critic updates, contributing to greater stability in the training process; the third improvement'target policy smoothing – adds controlled noise to the actions of the target policy, thereby reducing overfitting to noise in the data.

A special feature of TD3 is its efficient use of samples from the experience buffer, which makes the algorithm economical in terms of data requirements. In MEC tasks, this property manifests itself in high resilience to noise in performance metrics and network parameters. TD3 is particularly well-suited for fine-tuning continuous placement parameters, such as CPU allocation, RAM allocation, and CPU affinity settings, which require precise control over continuous actions without sharp fluctuations in strategy.

The PPO algorithm is an on-policy method that features a «clipping» mechanism to constrain policy updates, preventing radical changes in strategy in a single training step [16]. This ensures a monotonic improvement in performance and high stability in the training process. The algorithm is known for its simple hyperparameter tuning and very rarely exhibits catastrophic training failures («gradient explosions» or sharp degradations in policy quality).

Although PPO requires more data for training compared to off-policy methods, it provides a highly predictable training dynamic and is well-suited for parallelization, which makes it attractive for large-scale MEC deployments. In the context of service placement, a key advantage of PPO is its «gentle» policy update, which protects against sudden degradations in service level objectives (SLOs) when deploying a model in an online environment. While other algorithms might show significant performance drops during fine-tuning or adaptation to new conditions, PPO ensures smooth transitions and stable maintenance of service quality.

Based on this analysis, traditional Q-learning and DQN methods demonstrate significant limitations in the context of MEC systems. Q-learning is effective only for environments with a small and finite number of states, but it struggles with large or continuous state spaces. Meanwhile, DQN suffers from overestimation problems and exhibits unstable convergence [17].

A fundamental drawback of these approaches is their inability to effectively handle continuous space, which is critical for precise resource management in edge computing. Modern algorithms, including SAC, TD3, and PPO, overcome these limitations, providing more stable training and efficient performance in the high-dimensional continuous state and action spaces typical of MEC environments.

3. Decision-Making Methods for Reducing Network Latency in Edge Computing. In the context of edge computing, as previously mentioned, the decision of where to place computing services is a multidimensional

optimization task that requires accounting for many interconnected factors of the network infrastructure. As shown in previous sections, existing approaches to modeling MEC environments often use simplified abstractions of network topology, which limits the applicability of the results in real-world operational conditions.

A key problem in current research is the insufficiently accurate representation of the spatial distribution of network infrastructure elements and the dynamics of user load. The use of stochastic geometry methods is necessary to define a model for the physical location of cellular towers and to determine the distance between them, which is critical for accurately calculating network latencies and radio signal quality.

Traditional approaches to modeling MEC environments rely on deterministic topologies or overly simplistic probabilistic models that do not reflect the real complexity of modern cellular networks. All this leads to the generation of training data that can differ significantly from real-world conditions, which, in turn, reduces the quality of trained RL models deployed in a production environment.

This section presents an approach to creating a more realistic MEC environment model based on integrating stochastic geometry methods with a practical simulation platform. We hypothesize that the use of stochastic geometry methods to calculate the UE per BS distribution in combination with a realistic LWMECPS environment based on Minikube will improve the quality of RL algorithm training by providing higher-quality data.

The proposed approach aims to overcome the limitations of existing methods by creating a hybrid model that combines the mathematical rigor of stochastic modeling with the practical applicability of containerized computing environments. This allows for the creation of a test environment that maintains the controllability and reproducibility of experiments while ensuring a high degree of correspondence to the real-world operating conditions of MEC systems.

We can represent the environment under investigation in a general form as:

$$State_t = \{Node_t^1, Node_t^2, \ldots, Node_t^i, Deployment_t^i, AvgLatency\}, \quad (1)$$

where

$$Node_t^i = \{CPU_t^i, RAM_t^i, TX_t^i, RX_t^i\}, \quad i = 1,2,3,\ldots,n, \quad (2)$$

$$
\begin{aligned}
Deployment_t^i = \{&CPU\_usage_t^i, \\
&RAM\_usage_t^i, \\
&TX\_usage_t^i, \\
&RX\_usage_t^i, \\
&Replicas_t^i\},
\end{aligned}
\tag{3}
$$

then we can represent the action as

$$
Action_t = \{Replicas_t^1, Replicas_t^2, \ldots, Replicas_t^i\}, \quad i = 1,2,3,\ldots,n, \tag{4}
$$

where

$$
Replicas_t^i \in [0, R_{\max}], \tag{5}
$$

and $R_{\max}$ is the maximum number of replicas per deployment.

$$
Imbalance_r = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i^r - \bar{x}^r)^2}. \tag{6}
$$

The following section examines each machine learning algorithm individually.

To describe Proximal Policy Optimization for network latency reduction tasks, we can represent the system state at discrete time points $t = 0,1,\ldots,T$ as:

$$
\begin{aligned}
s_t = \Big\{&Node_t^{(1)}, \ldots, Node_t^{(n)}, \\
&Deployment_t^{(1)}, \ldots, Deployment_t^{(n)}, \\
&AvgLatency_t \Big\},
\end{aligned}
\tag{7}
$$

where

$$Node_t^{(i)} = \left(CPU_t^{(i)}, RAM_t^{(i)}, TX_t^{(i)}, RX_t^{(i)}\right), \tag{8}$$

are the remaining resources of the i-th node;

$$\begin{aligned} Deployment_t^{(i)} = \Big(&CPU\_usage_t^{(i)}, RAM\_usage_t^{(i)}, \\ &TX\_usage_t^{(i)}, RX\_usage_t^{(i)}, \\ &Replicas_t^{(i)}\Big), \end{aligned} \tag{9}$$

and $AvgLatency_t$ is the moving average of the response latency.

Agent's action:

$$a_t = \left(Replicas_t^{(1)}, \ldots, Replicas_t^{(n)}\right), \quad Replicas_t^{(i)} \in \{0, \ldots, R_{\max}\}. \tag{10}$$

It is a multimodal discrete value (Multi-Discrete action space) that determines the new number of replicas for each deployment, while the reward function linearly combines metrics:

$$\begin{aligned} R_t = -\alpha AvgLatency_t - \beta \sum_{i=1}^{n} Replicas_t^{(i)} \\ - \gamma \sum_{r \in \{CPU, RAM, TX, RX\}} Imbalance_t^{(r)}, \end{aligned} \tag{11}$$

where $\alpha, \beta, \gamma > 0$ are the weights chosen empirically for the target «quality/cost» ratio.

An agent in PPO learns to find a stochastic policy as follows:

$$\pi_\theta(a_t|s_t) = \prod_{i=1}^{n} \pi_\theta^{(i)}(Replicas_t^{(i)}|s_t). \tag{12}$$

This is parameterized by a «shared neural network head» θ. Factorization by i simplifies the approximation of an enormous action space $(R_{\max}+1)^n$. The state is evaluated by a critic $V_\phi(s_t)$ with parameters ϕ.

After collecting a batch of trajectories of length T, we fix the old policy $\pi_{\theta_{old}}$ and calculate the probability ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}. \tag{13}$$

For a more stable gradient step, a clipped surrogate objective is used:

$$L_{\text{clip}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t \right) \right], \tag{14}$$

where $\varepsilon \in [0.1, 0.3]$ is a hyperparameter and $\hat{A}_t$ is an advantage estimate.

We also use a generalized advantage estimate (GAE-λ):

$$\hat{A}_t = \sum_{k=0}^{T-t-1} (\gamma\lambda)^k \left(R_{t+k} + \gamma V_\phi(s_{t+k+1}) - V_\phi(s_{t+k}) \right), \tag{15}$$

with discounting $\gamma \in (0,1)$ and a smoothing parameter $\lambda \in [0,1]$.

The overall loss function, which is optimized using the stochastic gradient method, also includes the critic's loss and an entropy regularization term:

$$L(\theta, \phi) = -L_{\text{clip}}(\theta) + c_v \hat{\mathbb{E}}_t \left[(V_\phi(s_t) - R_t)^2 \right] - c_s \hat{\mathbb{E}}_t \left[\mathscr{H}(\pi_\theta(\cdot|s_t)) \right], \tag{16}$$

where

$$R_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k}, \tag{17}$$

$\mathscr{H}$ is the policy entropy, and c_v, $c_s > 0$ are coefficients.

The condition «node resources are not exceeded» ($CPU\_usage_t^{(i)} \leqslant CPU_t^{(i)}$, etc.) is implemented through masking. Forbidden components $\hat{A}_t^{(i)}$ are assigned a zero probability before sampling, which ensures the correctness of $r_t(\theta)$. This strategy is compatible with the PPO formulation and does not violate the requirements of a monotonic policy. Algorithm 1 displays the sequence of actions for PPO training.

Algorithm 1. Proximal Policy Optimization (PPO) with Clipped Objective

Input: Initial policy parameters θ, value function parameters ϕ; clipping threshold ε; number of epochs K; mini-batch size B; horizon T; learning rates η_θ, η_ϕ; discount factor γ; GAE parameter λ
while not converged **do**
  Collect trajectories $\{(s_t, a_t, r_t, s_{t+1})\}$ for T timesteps using current policy π_θ
  Compute advantage estimates $\hat{A}_t$ using GAE:

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t), \quad \hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

  Compute empirical returns: $\hat{R}_t = \hat{A}_t + V_\phi(s_t)$
  Compute old log probabilities: $\log \pi_\theta(a_t|s_t)$
  for $k = 1$ to K **do** ▷ Number of policy updates per batch
    Sample mini-batch of size B from collected data
    Compute probability ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

    Compute clipped surrogate objective:

$$\mathscr{L}_\theta^{\text{CLIP}} = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t \right) \right]$$

    Update policy parameters:

$$\theta \leftarrow \theta + \eta_\theta \nabla_\theta \mathscr{L}_\theta^{\text{CLIP}}$$

    Update value function by minimizing:

$$\mathscr{L}_V(\phi) = \hat{\mathbb{E}}_t \left[\left(V_\phi(s_t) - \hat{R}_t \right)^2 \right]$$

$$\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathscr{L}_V(\phi)$$

  end for
end while
Output: Optimized policy parameters θ

The Soft Actor-Critic algorithm is adapted for discrete multi-action environments in network resource allocation tasks. The policy $\pi_\theta(a_t \mid s_t)$ is parameterized by a neural network θ that outputs action logits for each deployment dimension. The factorization by i simplifies the approximation of an enormous action space $(R_{\max} + 1)^n$.

The state is evaluated by two critic networks $V_{\phi_1}(s_t)$ and $V_{\phi_2}(s_t)$ with parameters ϕ_1 and ϕ_2, where the minimum Q-value is used for stability.

Actor-Critic Architecture we can represent like this:

– **Actor Network:** Outputs logits for categorical distribution over replica counts $[0, R_{\max}]$ for each deployment;

– **Critic Network:** Two Q-networks that estimate $Q(s_t, a_t)$ for state-action pairs;

– **Target Network:** Soft updates with parameter τ for critic target networks.

Critic Update: Minimize the Bellman error with entropy regularization:

$$L_{\text{critic}} = \mathbb{E}\left[\left(Q(s_t,a_t) - \left(r_t + \gamma\left(\min Q'(s_{t+1},a_{t+1}) - \alpha \log \pi(a_{t+1} \mid s_{t+1})\right)\right)\right)^2\right].$$

Actor Update: Maximize expected reward plus entropy:

$$L_{\text{actor}} = \mathbb{E}\left[\alpha \log \pi(a_t \mid s_t) - \min(Q_1(s_t,a_t), Q_2(s_t,a_t))\right].$$

Temperature Update: Automatic entropy tuning:

$$L_\alpha = \mathbb{E}\left[-\alpha \log \pi(a_t \mid s_t) - \alpha H_{\text{target}}\right].$$

Algorithm 2 displays the sequence of actions for SAC training. The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm is adapted for discrete multi-action environments in network resource allocation tasks.

The policy $\pi_\theta(a_t \mid s_t)$ is parameterized by a neural network θ that outputs discrete action indices for each deployment dimension. The factorization by i simplifies the approximation of an enormous action space $(R_{\max}+1)^n$.

The state is evaluated by two critic networks $V_{\phi_1}(s_t)$ and $V_{\phi_2}(s_t)$ with parameters ϕ_1 and ϕ_2, where the minimum Q-value is used to reduce overestimation bias.

Algorithm 2. Soft Actor-Critic (SAC)

Input: Policy parameters θ, critic parameters ϕ_1, ϕ_2, target networks $\bar{\phi}_1 \leftarrow \phi_1$, $\bar{\phi}_2 \leftarrow \phi_2$, temperature coefficient α, discount factor γ, target smoothing coefficient τ, learning rates $\eta_Q, \eta_\pi, \eta_\alpha$, experience replay buffer $\mathscr{D}$
while not converged **do**
  Collect transitions (s_t, a_t, r_t, s_{t+1}) by interacting with the environment using policy $\pi_\theta(a|s)$
  Store transitions in replay buffer: $\mathscr{D} \leftarrow \mathscr{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
  for all mini-batches sampled from $\mathscr{D}$ **do**
    Sample next action $a_{t+1} \sim \pi_\theta(\cdot|s_{t+1})$
    Compute target Q-value:

$$y_t = r_t + \gamma \left(\min_{i=1,2} Q_{\bar{\phi}_i}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\theta(a_{t+1}|s_{t+1}) \right)$$

    Update each critic by minimizing the loss:

$$\phi_i \leftarrow \phi_i - \eta_Q \nabla_{\phi_i} \left(Q_{\phi_i}(s_t, a_t) - y_t \right)^2, \quad i = 1, 2$$

    Update the policy parameters via:

$$\theta \leftarrow \theta - \eta_\pi \nabla_\theta \hat{\mathbb{E}}_{a_t \sim \pi_\theta} \left[\alpha \log \pi_\theta(a_t|s_t) - \min_i Q_{\phi_i}(s_t, a_t) \right]$$

    If using adaptive temperature, update α:

$$\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha \hat{\mathbb{E}}_{a_t \sim \pi_\theta} \left[-\alpha \left(\log \pi_\theta(a_t|s_t) + \mathscr{H}_{\text{target}} \right) \right]$$

    Update target networks:

$$\bar{\phi}_i \leftarrow \tau \phi_i + (1 - \tau) \bar{\phi}_i, \quad i = 1, 2$$

  end for
end while
Output: Learned parameters θ, ϕ_1, ϕ_2, and α

Actor-Critic Architecture:

– **Actor Network:** Outputs discrete action indices via arg max over replica counts $[0, R_{\max}]$ for each deployment;

– **Critic Networks:** Two Q-networks that estimate $Q(s_t, a_t)$ for state-action pairs;

– **Target Networks:** Soft updates with parameter τ for all target networks;

– **Critic Update:** Minimize the Bellman error using twin critics:

$$L_{\text{critic}} = \hat{\mathbb{E}}\left[\left(Q(s_t,a_t) - \left(r_t + \gamma \min\begin{pmatrix} Q'_1(s_{t+1},\pi'(s_{t+1})), \\ Q'_2(s_{t+1},\pi'(s_{t+1})) \end{pmatrix}\right)\right)^2\right];$$

– **Actor Update:** Maximize Q-value with policy delay:

$$L_{\text{actor}} = \hat{\mathbb{E}}\left[-Q_1(s_t,\pi(s_t))\right].$$

The standard TD3 algorithm has been adapted for discrete action spaces by:

– converting discrete action indices to float for critic networks;
– removing target policy smoothing (not applicable to discrete actions);
– using arg max for action selection instead of continuous outputs;
– adding exploration noise through discrete action perturbation.

Algorithm 3 displays the sequence of actions for TD3 training. Based on the mathematical formulations presented above, the PPO, SAC, and TD3 algorithms were implemented for network latency reduction in Mobile Edge Computing (MEC) environments.

The three algorithms demonstrate distinct characteristics in their approach to policy optimization. PPO employs a clipped surrogate objective with generalized advantage estimation, ensuring stable policy updates. SAC utilizes entropy regularization with automatic temperature tuning to balance exploration and exploitation. TD3 implements twin critics with policy delay mechanisms to reduce overestimation bias and enhance training stability.

In environments with extensive discrete action spaces, methods based on value function learning (such as Q-learning and its modifications) are subject to fundamental scalability limitations. The computational complexity of Q-value updates increases linearly with the size of the action space, while maximization bias increases proportionally to the logarithm of the number of alternatives. Traditional exploration strategies, such as the ε-greedy approach and Boltzmann policies, become exponentially less efficient as the cardinality of the action space increases. This results in slower convergence and greater variance in estimates.

Actor-critic methods (Proximal Policy Optimization, Soft Actor-Critic and Twin Delayed Deep Deterministic Policy Gradient) offer architectural solutions to these issues. Direct parametric policy learning eliminates the need to accurately approximate value functions for the entire action space, allowing computational resources to be concentrated on statistically significant regions. Regularization mechanisms – KL divergence in PPO, entropy

constraints in SAC and ensemble critics in TD3 – stabilize learning and mitigate overestimation artifacts. Integrated stochastic exploration strategies implemented through entropy bonuses, stochastic policy parameterization and noise perturbations support effective exploration given the combinatorial complexity of the action space.

Algorithm 3. Twin Delayed Deep Deterministic Policy Gradient (TD3)

Input: Initial actor parameters θ, critic parameters ϕ_1, ϕ_2, target networks

$$\bar{\theta} \leftarrow \theta, \quad \bar{\phi}_1 \leftarrow \phi_1, \quad \bar{\phi}_2 \leftarrow \phi_2;$$

exploration noise $\mathscr{N}$; target smoothing noise scale σ; policy update delay d; learning rates η_Q, η_π; target update rate τ; discount factor γ; replay buffer $\mathscr{D}$

while not converged **do**

  Collect action with exploration: $a_t = \pi_\theta(s_t) + \mathscr{N}_t$

  Execute a_t, observe r_t, s_{t+1}

  Store transition (s_t, a_t, r_t, s_{t+1}) in buffer $\mathscr{D}$

  for all mini-batches sampled from $\mathscr{D}$ **do**

    Add clipped noise: $\tilde{a}_{t+1} = \pi_{\bar{\theta}}(s_{t+1}) + \varepsilon$, where $\varepsilon \sim \mathscr{N}(0, \sigma)$, clipped to $[-c, c]$

    Compute target Q-value:

$$y_t = r_t + \gamma \min_{i=1,2} Q_{\bar{\phi}_i}(s_{t+1}, \tilde{a}_{t+1})$$

    Update critics:

$$\phi_i \leftarrow \phi_i - \eta_Q \nabla_{\phi_i} \left(Q_{\phi_i}(s_t, a_t) - y_t \right)^2, \quad i = 1, 2$$

    if iteration % d == 0 **then**

      Update actor (policy gradient):

$$\theta \leftarrow \theta - \eta_\pi \nabla_\theta Q_{\phi_1}(s_t, \pi_\theta(s_t))$$

      Update target networks:

$$\bar{\phi}_i \leftarrow \tau \phi_i + (1 - \tau) \bar{\phi}_i, \quad \bar{\theta} \leftarrow \tau \theta + (1 - \tau) \bar{\theta}$$

    end if

  end for

end while

Output: Trained actor θ and critics ϕ_1, ϕ_2

These reinforcement learning approaches provide a comprehensive framework for adaptive resource management in dynamic MEC environments,

enabling intelligent decision-making for replica allocation that directly impacts network performance and user experience.

4. Limitations. The model considered in this study has certain limitations related to the challenges of emulating processes within a communication network, and these must be taken into account.

For instance, we opted for a uniform distribution with a latency of 10 ms. While this decision may raise questions, there is an explanation. Currently, the absence of straightforward traffic balancing tools within the LWMECPS network hinders the implementation of transitive latencies.

Since their spatial locations are known, we could of course determine the distance between nodes and use simple formulas to calculate the latency between them. However, we must remember that traffic within the network will follow a path formed by an IGP protocol, such as OSPF, EIGRP or IS-IS.

Simultaneously, we utilize the distribution of users in time and space, enabling us to influence network latency by adjusting the load on UEs.

Furthermore, solutions based on SDN, IPIP tunnels and network slicing allow for flexible traffic management. However, these are difficult to account for in all use cases. Therefore, for now, we have decided to use linearly increasing latencies, but we could calculate them more accurately in the future.

5. Development of a System for Evaluating RL Algorithms in Network Latency Reduction Tasks. To practically verify the proposed hypotheses and obtain quantitative estimates of the effectiveness of modern RL algorithms in service placement tasks, it is necessary to create a comprehensive experimental environment.

Such an environment must not only ensure a correct simulation of network conditions approximated to real-world scenarios but also provide tools for a systematic comparison of different decision-making approaches in MEC systems.

Therefore, to organize an experiment that would allow us to test our hypothesis, it was necessary to refine the previously discussed LWMECPS system [13] and develop a test application that could emulate network activity and measure the latency between nodes, with support for experiments and network activity distribution.

We can highlight the following requirements for refining LWMECPS-GYM:

– add support for PPO, SAC, and TD3 RL algorithms;

– add the ability to create experiments with specific parameters for the corresponding machine learning algorithms;

– add the ability to collect data on average latency from the LWMECPS-testapp-client by `group_id`;

– add the ability to save machine learning models, logs, and metrics to the WandB machine learning model storage system.

In addition, the `lwmecps-testapp` test application had to meet the following requirements:

– ability to configure load parameters;
– support for mechanisms to measure and log performance metrics;
– ability to emulate various user behavior scenarios.

The `lwmecps-testapp` test application is a distributed system designed to investigate the effectiveness of reinforcement learning algorithms when optimizing network latencies in edge computing.

Based on a microservice approach, the system's architecture includes three main components: a client application, a server application, and a data management system based on MongoDB.

The `lwmecps-testapp-client` service is implemented using the FastAPI framework and performs the functions of a load generator and an experiment coordinator. The main functional module includes managing the experiment lifecycle, generating a controlled load, and collecting performance metrics.

The experiment management system enables the creation, initialization, and control of experiment states through REST API endpoints. Each experiment features a set of parameters, including the configuration of target servers, load profiles, and time constraints. Load profiles consist of three parameters (N, T, D), where N is the number of concurrent users, T represents the interval between requests, and D denotes the profile duration.

The load generation module implements an asynchronous architecture using `asyncio`, ensuring high throughput with minimal resource consumption. The system supports up to 100 concurrent tasks with configurable connection timeouts. The latency measurement algorithm includes host prioritization and automatic failover in case of node failure.

The `lwmecps-testapp-server` service emulates the behavior of edge computing nodes under variable load. The server architecture consists of a request processing module, a resource emulation system, and metric export.

The main endpoint `/api/latency` provides latency measurement with a realistic modeling of edge node behavior. The resource emulation system implements synthetic modeling of CPU and RAM usage with configurable resource limits.

Figure 1 shows the architecture of the LWMECPS test setup. LitmusChaos, as discussed in previous articles, is also used as a system for introducing latency.

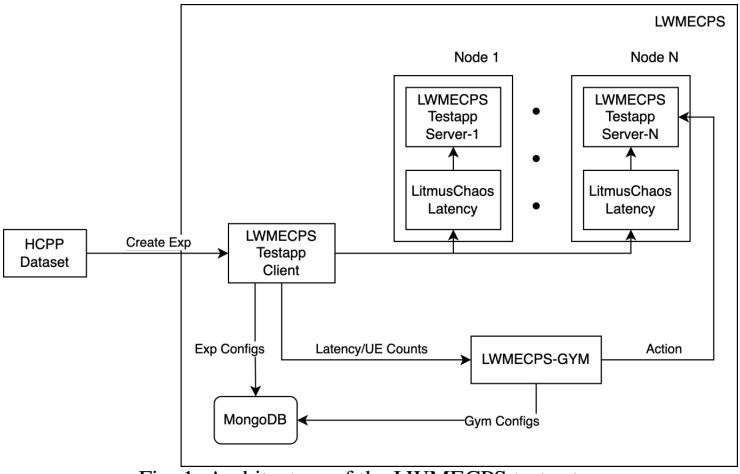

Fig. 1. Architecture of the LWMECPS test setup

As initial data, the user distribution from article [18] was taken and processed through the Hard-Core Poisson Process (HCPP) algorithm to determine the distribution of users per BS, based on the method discussed in [19].

Using the distribution data of BSs and UEs per DB, we formed a user distribution throughout the day based on the previously discussed statistics and created a graph from 00:00 to 23:59 with a 10-second step, as shown in Figure 2 [20].

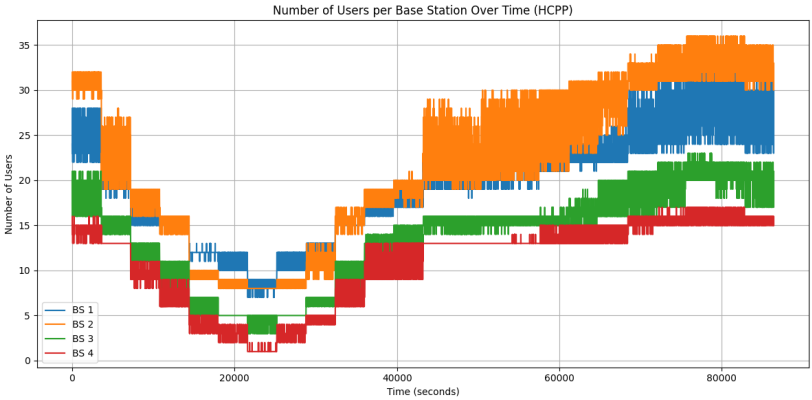

Fig. 2. Distribution diagram of UEs per BS during 1 day

This distribution enables more accurate training of the RL model, allowing it to account for the non-uniform usage of communication networks depending on time. The generated distribution promoted the preparation of experiments for each `lwmecps-testapp-server` deployed on the LWMECPS nodes.

Figure 3 shows the architecture of the `lwmecps-testapp` for experiment 1, from the perspective of the communication network and the introduced latencies.

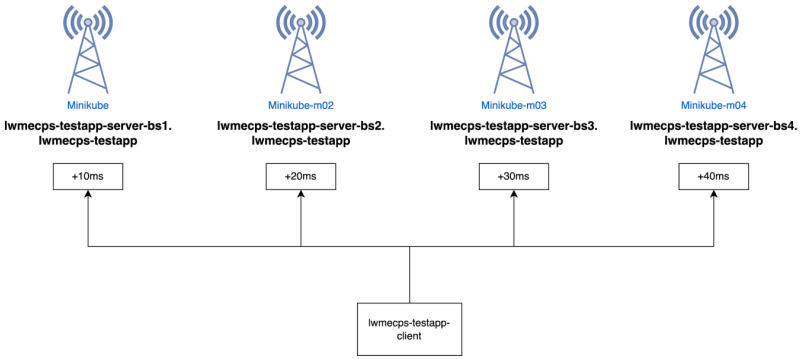

Fig. 3. lwmecps-testapp architecture for experiment 1

We must note the problem of transitive latencies. At this stage, it is not possible to implement transitive latencies due to the complexities of modeling traffic flow. However, this problem has the following solution.

The experiment involves 4 BSs, with identifiers `bs_1`, ..., `bs_4`. Each `bs_n` has a unique load profile and host list. The `lwmecps-testapp-client` algorithm works as follows: if the first `lwmecps-testapp-server` endpoint in the queue is unavailable, it will try the next one. Thus, the location of computing services dynamically changes, and client traffic will flow.

To train the reinforcement learning algorithms, we prepared two network load profiles: from 0 to 21600 seconds (6 hours) and from 0 to 86400 seconds (24 hours). All the RL algorithms were trained on each network load profile for the corresponding duration, allowing us to test not only the quality of the algorithms against each other but also the influence of changing network load over time on the quality of decisions made.

To evaluate the quality of decisions made to reduce network latency, six verification experiments were prepared, each running for 100 episodes. This approach enabled the evaluation of six machine learning models based on three algorithms, all of them trained for 6 and 24 hours.

Figures 4, 5, and 6 show the results of network latency optimization by the PPO, SAC, and TD3 algorithms for models trained for 6 and 24 hours.

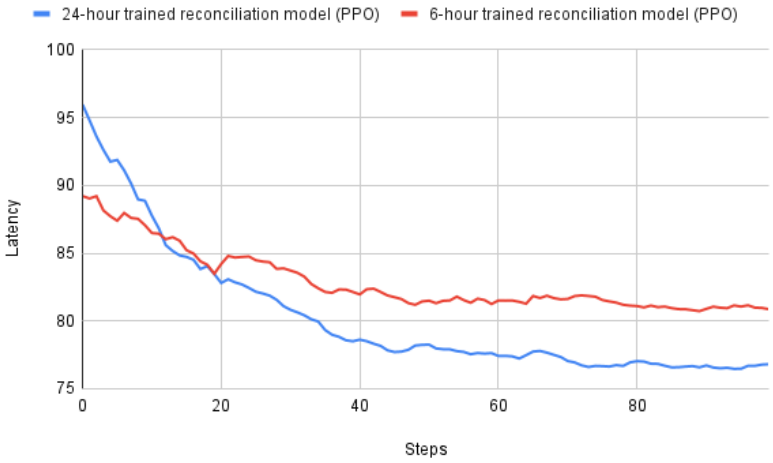

Fig. 4. Network latency optimization graph by PPO algorithm for 6 and 24 hour models

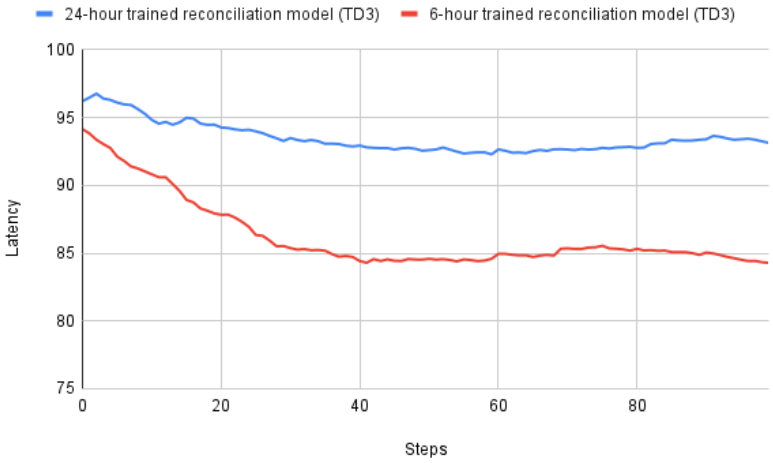

Fig. 5. Network latency optimization graph by TD3 algorithm for 6 and 24 hour models

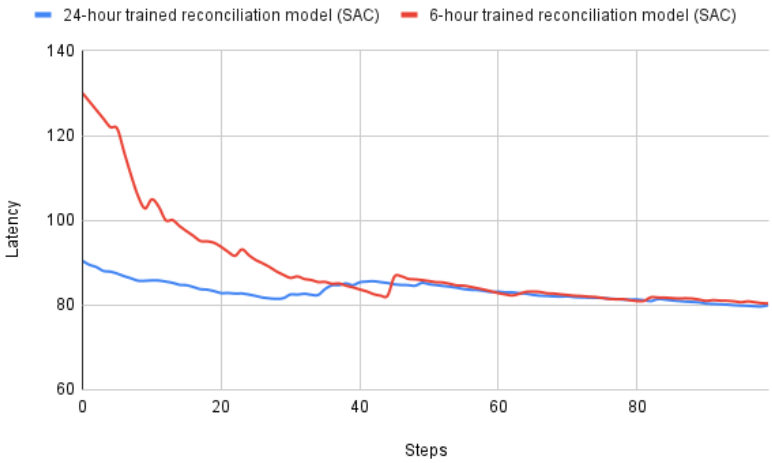

Fig. 6. Network latency optimization graph by SAC algorithm for 6 and 24 hour models

Table 1 shows the numerical results of the comparison of the PPO, SAC, and TD3 models trained for 6 and 24 hours to assess the impact of training duration on the quality of decisions made. For comparison, the initial and final average delay values were taken, as well as their delta in absolute and percentage terms.

Table 1. Comparison of RL algorithms for 6 and 24 hour training

Algorithm	Initial Latency	Final Latency	Absolute Delta (ms)	Percent Change
PPO 6 h	$\approx$ 90 ms	$\approx$ 81 ms	-9	-10%
PPO 24 h	$\approx$ 96 ms	$\approx$ 77 ms	-19	-19%
TD3 6 h	$\approx$ 94 ms	$\approx$ 84 ms	-10	-11%
TD3 24 h	$\approx$ 95 ms	$\approx$ 90 ms	-5	-5%
SAC 6 h	$\approx$ 90 ms	$\approx$ 79 ms	-11	-12%
SAC 24 h	$\approx$ 130 ms	$\approx$ 80 ms	-50	-38%

Based on the analysis of the data in the table, we can draw the following conclusions.

The PPO algorithm trained for 24 hours shows the best performance in network latency reduction, lowering it by nearly 20%. According to the graphs, PPO also has the smoothest trajectory – the latency curve converges to a local minimum and stabilizes quickly.

The TD3 algorithm shows the best latency reduction among 6-hour experiments. However, it shows poor results when trained for 24 hours, with several runs yielding different outcomes. The TD3 algorithm is overly sensitive

to hyperparameters whose influence can significantly alter the experimental results.

The SAC algorithm demonstrates the most dramatic improvement in the 24-hour training scenario, achieving a 38% latency reduction despite starting from a higher initial latency ($\sim$130 ms). However, the high initial latency suggests potential initialization or early training instability issues that warrant further investigation.

Based on the data obtained, we can conclude that the PPO algorithm is optimal for reducing the average network latency. Unlike robotics or autonomous vehicle systems, which require the fine-tuning of continuous parameters like steering angle or engine power, the task of service placement in an MEC environment is characterized by a discrete action space – selecting a specific node from an available set to deploy a particular service [21]. The SAC algorithm was developed for continuous action spaces [22]. Although modifications exist for discrete tasks [23], they require additional adaptation, which can negatively affect performance [24].

The discreteness of the action space poses significant challenges for policy optimization in deep reinforcement learning methods. The inability to differentiate the argmax operation prevents the use of gradient methods for direct action optimization. Meanwhile, the discrete redistribution of probability mass between actions leads to discontinuous changes in policy. These features exacerbate bootstrap error in critic algorithms and increase sensitivity to inaccuracies in Q-function estimates for rare actions. They also contribute to premature policy determination, which leads to the collapse of state and action space exploration.

We define algorithmic stability as low performance variance across seeds, absence of policy collapse during extended training, stable value estimates and gradients, and robustness to moderate distribution shifts.

PPO exhibits sensitivity to prolonged training through on-policy data reuse. Extended epochs on fixed batches with constant clipping induce batch overfitting, where surrogate improvements fail to translate to reward gains. Additionally, entropy decay despite bonuses leads to deterministic policies with reduced robustness. Our protocol limits epochs to two per batch, monitors adaptive KL divergence, enforces entropy floors, and employs early stopping with validation-based checkpoint selection.

SAC's maximum entropy framework enhances stability by maintaining state coverage. However, automatic temperature adjustment reduces entropy over time, causing determinization and robustness loss. Fixed replay buffers create distribution shifts as aged samples bias Q-functions, while bootstrap artifacts accumulate over extended horizons. Mitigation strategies include

minimum alpha constraints, critic weight decay, Polyak EMA updates, and EMA checkpoint preservation.

TD3's twin critics and target smoothing stabilize early training phases. Extended training can induce conservative Q-underestimation, freezing improvement and amplifying noise sensitivity. Suboptimal Polyak coefficients and update frequencies desynchronize networks, increasing TD-error and reducing replay diversity. Stability is maintained through careful policy delay and target noise control.

In contrast, the PPO algorithm naturally supports both discrete and continuous action spaces, making it more suitable for service placement tasks without architectural modifications.

Understanding the specifics of MEC platforms, we can assert that the requirements characteristic of this domain – convergence stability, resilience to changes in network conditions, and the need to maintain quality of service – align particularly well with the architectural features of PPO. Unlike studies in robotics [25] where algorithm comparisons involve tasks with continuous control, MEC tasks are combinatorial optimization problems with discrete choices. The observed instability of TD3 in the experiments is also explained by the fact that this algorithm, like SAC, is optimized for continuous actions, and its application to discrete tasks can lead to suboptimal behavior.

6. Conclusion. This study conducted a comprehensive analysis of the effectiveness of modern reinforcement learning algorithms – Proximal Policy Optimization (PPO), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC) – in the task of optimizing service placement to minimize network latency in edge computing.

To ensure the correctness of the experimental studies, we significantly refined the LWMECPS platform by upgrading the Gymnasium API interface to support the investigated machine learning algorithms. Additionally, a comprehensive test environment, `lwmecps-testapp`, was developed to emulate realistic network load scenarios using stochastic geometry methods to model the spatial distribution of base stations and user devices.

The experimental results demonstrate significant differences in the effectiveness of the algorithms:

– The PPO algorithm showed the best results, achieving an average network latency reduction of up to 20% when trained on a 24-hour load profile. PPO's key advantages are its high convergence stability, natural support for discrete action spaces, and resilience to variations in network conditions, making it the most suitable algorithm for practical application in MEC systems.

– The TD3 algorithm demonstrated moderate effectiveness with a maximum latency reduction of 11% for a 6-hour training profile. However,

we revealed a critical sensitivity of the algorithm to hyperparameter tuning, leading to significant variability in results and limiting its practical applicability without meticulous configuration.

– The SAC algorithm demonstrated mixed results: while achieving the highest absolute latency reduction (38%) in extended training scenarios, it exhibited initialization challenges with elevated initial latency values. This behavior is consistent with SAC's original design for continuous action spaces, suggesting that discrete MEC service placement tasks may require additional algorithmic adaptations for optimal performance.

The results obtained confirm the hypothesis that using more realistic models of the network environment, based on stochastic geometry methods, improves the quality of RL algorithm training. This is especially important for ensuring the applicability of the resulting solutions in real-world MEC system operations.

Future work should focus on investigating hybrid approaches that combine the stability of PPO with the exploration capabilities of other algorithms, as well as validating these findings in real MEC testbed environments.

References

1. Cruz P., Achir N, Viana A.C. On the edge of the deployment: A survey on multi-access edge computing. ACM Computing Surveys. 2022. vol. 55. no. 5. pp. 1–34. DOI: 10.1145/3529758.
2. Yi Y., Zhang G., Jiang H. Mobile Edge Computing Networks: Online Low-Latency and Fresh Service Provisioning. IEEE Transactions on Communications. 2025.
3. Wang K., Akhtar S.F., Al-Zahrani F.A. An Efficient Algorithm for Resource Allocation in Mobile Edge Computing Based on Convex Optimization and Karush–Kuhn–Tucker Method. Complexity. 2023. vol. 2023(1). pp. 1–15. DOI: 10.1155/2023/9604454.
4. Qin Y., Chen J., Jin L., Yao R., Gong Z. Task offloading optimization in mobile edge computing based on a deep reinforcement learning algorithm using density clustering and ensemble learning. Scientific Reports. 2025. vol. 15. no. 1. DOI: 10.1038/s41598-024-84038-3.
5. Rodríguez-Liria A.F., Cárdenas R., Arroba P., Moya J.M., Risco-Martín J.L., Wainer G. Decision Support Framework for Automating the Optimization of Edge Computing Federations. Proceedings of 2023 Annual Modeling and Simulation Conference (ANNSIM). 2023. pp. 49–60.
6. Liu J., Ren J., Zhang Y., Peng X., Zhang Y., Yang Y. Efficient Dependent Task Offloading for Multiple Applications in MEC-Cloud System. IEEE Transactions on Mobile Computing. 2021. vol. 22. no. 4. pp. 2147–2162. DOI: 10.1109/TMC.2021.3119200.
7. Bogatyrev V.A., Bogatyrev S.V., Bogatyrev A.V. Control of multipath transmissions in the nodes of switching segments of reserved paths. Proceedings of International Conference on Information, Control, and Communication Technologies (ICCT). 2022. pp. 1–5.
8. Wang X., Ji Y., Zhang J., Bai L., Zhang M. Low-Latency Oriented Network Planning for MEC-Enabled WDM-PON Based Fiber-Wireless Access Networks. IEEE Access. 2019. vol. 7. pp. 183383–183395. DOI: 10.1109/ACCESS.2019.2926795.

9. Ko S.-W., Han K., Huang K. Wireless Networks for Mobile Edge Computing: Spatial Modeling and Latency Analysis. IEEE Transactions on Wireless Communications. 2018. vol. 17. no. 8. pp. 5225–5240. DOI: 10.1109/TWC.2018.2840120.
10. Elghitani F. Dynamic UAV routing for multi-access edge computing. IEEE Transactions on Vehicular Technology. 2024. vol. 73. no. 6. pp. 8878–8888. DOI: 1109/TVT.2024.3360253.
11. Dankolo N.M., Radzi N.H.M., Mustaffa N.H., Arshad N.I., Nasser M., Gabi D., Yusuf M.N. Optimizing resource allocation for IoT applications in the edge cloud continuum using hybrid metaheuristic algorithms. Scientific Reports. 2024. vol. 15. no. 1. DOI: 10.1038/s41598-025-97648-2.
12. Ismail A.A., Khalifa N.E., El-Khoribi R.A. A survey on resource scheduling approaches in multi-access edge computing environment: A deep reinforcement learning study. Cluster Computing. 2025. vol. 28. no. 3. DOI: 10.1007/s10586-024-04893-7.
13. Filianin I., Kapitonov A., Timoshchuk-Bondar A. Gymnasium Library Interface for Multi Access Edge Computing. Proceedings of 2024 6th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA). 2024. pp. 162–166. DOI: 10.1109/SUMMA64428.2024.10803891.
14. Sun C., Wu X., Li X., Fan Q., Wen J., Leung V.C.M. Cooperative Computation Offloading for Multi-Access Edge Computing in 6G Mobile Networks via Soft Actor Critic. Proceedings of IEEE Transactions on Network Science and Engineering. 2021. vol. 11. no. 6. pp. 5601–5614. DOI: 10.1109/TNSE.2021.3076795.
15. Saad M.M., Jamshed M.A., Adedamola A.I., Nauman A., Kim D. Twin Delayed DDPG (TD3)-Based Edge Server Selection for 5G-Enabled Industrial and C-ITS Applications. IEEE Open Journal of the Communications Society. 2025. vol. 6. pp. 3332–3343. DOI: 10.1109/OJCOMS.2025.3545566.
16. An L., Wang Z., Yue J., Ma X. Joint Task Offloading and Resource Allocation via Proximal Policy Optimization for Mobile Edge Computing Network. Proceedings of International Conference on Networking and Network Applications (NaNA). 2021. pp. 466–471. DOI: 10.1109/NaNA53684.2021.00087.
17. Zhu L., Tan L., Li B., Tian H. An optimization scheme for vehicular edge computing based on Lyapunov function and deep reinforcement learning. IET Communications. 2024. vol. 18. no. 15. pp. 908–924. DOI: 10.1049/cmu2.12800.
18. Facchini C., Holland O., Granelli F., da Fonseca N.L.S., Aghvami H. Dynamic green self-configuration of 3G base stations using fuzzy cognitive maps. Computer Networks. 2013. vol. 57. no. 7. pp. 1597–1610. DOI: 10.1016/j.comnet.2013.02.011.
19. Filianin I. Lwmecps-gym. GitHub repository. Available at: https://github.com/adeptvin1/lwmecps-gym (accessed 01.08.2025).
20. Filianin I. Lwmecps-testapp. GitHub repository. Available at: https://github.com/adeptvin1/lwmecps-testapp (accessed 03.08.2025).
21. Nieto G., de la Iglesia I., Lopez-Novoa U., Perfecto C. Deep Reinforcement Learning techniques for dynamic task offloading in the 5G edge-cloud continuum. Journal of Cloud Computing. 2024. vol. 13. no. 1. DOI: 10.1186/s13677-024-00658-0.
22. Haarnoja T., Zhou A., Abbeel P., Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. Proceedings of the 35th International Conference on Machine Learning. 2018. vol. 80. pp. 1861–1870.
23. Christodoulou P. Soft actor-critic for discrete action settings. arXiv preprint arXiv:1910.07207. 2019. DOI: 10.48550/arXiv.1910.07207.
24. Ismail A.A., Khalifa N.E., El-Khoribi R.A. A survey on resource scheduling approaches in multi-access edge computing environment: A deep reinforcement learning study. Cluster Computing. 2025. vol. 28. no. 3. DOI: 10.1007/s10586-024-04893-7.

25. Mock J.W., Muknahallipatna S.S. A comparison of PPO, TD3 and SAC reinforcement algorithms for quadruped walking gait generation. Journal of Intelligent Learning Systems and Applications. 2023. vol. 15. no. 1. pp. 36–56. DOI: 10.4236/jilsa.2023.151003.

Filianin Ivan — Ph.D. student, Faculty of software engineering and computer systems, ITMO University. Research interests: machine learning algorithms, deep neural networks, reinforcement learning, and their application in decision-making tasks for deploying computing services in geographically distributed data processing nodes, as well as in production optimization tasks. The number of publications — 6. adeptvin1@gmail.com; 49, Kronverksky Ave., 197101, St. Petersburg, Russia; office phone: +7(999)080-3237.

Kapitonov Aleksandr — Ph.D., Associate Professor, Deputy dean, New Uzbekistan University. Research interests: robot economics, blockchain technology applications in robotics, autonomous systems and UAVs, control theory, multi-agent systems, the integration of Large Language Model (LLM) agents in robotic systems. The number of publications — 77. kap2fox@gmail.com; 1, Movarounnahr, 100000, Tashkent, Uzbekistan; office phone: +7(950)000-2896.

Timoshchuk-Bondar Artem — Software engineer, Faculty of software engineering and computer systems, ITMO University. Research interests: machine learning algorithms, deep neural networks, reinforcement learning, and their application in decision-making tasks for deploying computing services in geographically distributed data processing nodes, as well as in production optimization tasks. The number of publications — 1. artbondar2003@gmail.com; 49, Kronverksky Ave., 197101, St. Petersburg, Russia; office phone: +7(950)517-2406.

УДК 006.72 **DOI** 10.15622/ia.25.1.8

И.В. Филянин, А.А. Капитонов, А.И. Тимощук- Бондар

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ ДЛЯ СНИЖЕНИЯ СЕТЕВОЙ ЗАДЕРЖКИ В ГРАНИЧНЫХ ВЫЧИСЛЕНИЯХ

Филянин И.В., Капитонов А.А., Тимощук-Бондар А.И. **Исследование алгоритмов обучения с подкреплением для снижения сетевой задержки в граничных вычислениях.**

Аннотация. Современные исследования алгоритмов принятия решений в системах multi-access edge computing (MEC) для задач распределения ресурсов зачастую основываются на упрощенных абстракциях сетевой топологии, что ограничивает применимость полученных результатов в реальных условиях эксплуатации мобильных сетей. Целью данной работы является разработка реалистичной модели сети сотовой связи с использованием методов стохастической геометрии и комплексная оценка эффективности современных алгоритмов обучения с подкреплением в задачах минимизации сетевых задержек в граничных вычислениях. Метод. Для создания математически обоснованной модели сетевой среды использовались методы стохастической геометрии в сочетании с реальными статистическими данными распределения пользователей сотовых сетей. Применение стохастической геометрии обеспечило корректное моделирование пространственного размещения базовых станций и расчет межузловых расстояний, критически важных для определения сетевых задержек. Экспериментальная оценка проводилась на базе доработанной платформы LWMECPS с расширенным Gymnasium API, поддерживающим алгоритмы PPO, TD3 и SAC. Основные результаты. Разработана модель сети связи, учитывающая реалистичное пространственное распределение сетевых элементов и временную динамику пользовательской нагрузки. На основе данной модели создано виртуализированное тестовое окружение в LWMECPS, позволяющее проводить воспроизводимые эксперименты с контролируемыми параметрами. Результаты экспериментов показали различия в характеристиках производительности различных алгоритмов: PPO обеспечил стабильное сокращение задержки до 20% со стабильной конвергенцией; SAC продемонстрировал наибольшее абсолютное улучшение (сокращение задержки на 38%), но проявил нестабильность при инициализации; TD3 показал умеренную эффективность (улучшение до 11%), но высокую чувствительность к настройке гиперпараметров. Обсуждение. Проведенный сравнительный анализ алгоритмов машинного обучения с подкреплением выявил ключевые особенности их применения в MEC-системах. Установлено, что дискретный характер задач размещения сервисов делает алгоритм PPO наиболее подходящим для практического внедрения в системы принятия решений благодаря его стабильности сходимости и естественной поддержке дискретных пространств действий. Полученные результаты предоставляют научно обоснованные рекомендации для разработчиков MEC-платформ по выбору оптимальных алгоритмических решений.

Ключевые слова: обучение с подкреплением, граничные вычисления с множественным доступом (Multi-Access Edge Computing), оптимизация политики по приближению (Proximal Policy Optimization), Soft Actor-Critic, алгоритм TD3 (Twin Delayed Deep Deterministic Policy Gradient), LWMECPS, Weights & Biases (WandB).

Литература

1. Cruz P., Achir N, Viana A.C. On the edge of the deployment: A survey on multi-access edge computing. ACM Computing Surveys. 2022. vol. 55. no. 5. pp. 1–34. DOI: 10.1145/3529758.
2. Yi Y., Zhang G., Jiang H. Mobile Edge Computing Networks: Online Low-Latency and Fresh Service Provisioning. IEEE Transactions on Communications. 2025.
3. Wang K., Akhtar S.F., Al-Zahrani F.A. An Efficient Algorithm for Resource Allocation in Mobile Edge Computing Based on Convex Optimization and Karush–Kuhn–Tucker Method. Complexity. 2023. vol. 2023(1). pp. 1–15. DOI: 10.1155/2023/9604454.
4. Qin Y., Chen J., Jin L., Yao R., Gong Z. Task offloading optimization in mobile edge computing based on a deep reinforcement learning algorithm using density clustering and ensemble learning. Scientific Reports. 2025. vol. 15. no. 1. DOI: 10.1038/s41598-024-84038-3.
5. Rodríguez-Liria A.F., Cárdenas R., Arroba P., Moya J.M., Risco-Martín J.L., Wainer G. Decision Support Framework for Automating the Optimization of Edge Computing Federations. Proceedings of 2023 Annual Modeling and Simulation Conference (ANNSIM). 2023. pp. 49–60.
6. Liu J., Ren J., Zhang Y., Peng X., Zhang Y., Yang Y. Efficient Dependent Task Offloading for Multiple Applications in MEC-Cloud System. IEEE Transactions on Mobile Computing. 2021. vol. 22. no. 4. pp. 2147–2162. DOI: 10.1109/TMC.2021.3119200.
7. Bogatyrev V.A., Bogatyrev S.V., Bogatyrev A.V. Control of multipath transmissions in the nodes of switching segments of reserved paths. Proceedings of International Conference on Information, Control, and Communication Technologies (ICCT). 2022. pp. 1–5.
8. Wang X., Ji Y., Zhang J., Bai L., Zhang M. Low-Latency Oriented Network Planning for MEC-Enabled WDM-PON Based Fiber-Wireless Access Networks. IEEE Access. 2019. vol. 7. pp. 183383–183395. DOI: 10.1109/ACCESS.2019.2926795.
9. Ko S.-W., Han K., Huang K. Wireless Networks for Mobile Edge Computing: Spatial Modeling and Latency Analysis. IEEE Transactions on Wireless Communications. 2018. vol. 17. no. 8. pp. 5225–5240. DOI: 10.1109/TWC.2018.2840120.
10. Elghitani F. Dynamic UAV routing for multi-access edge computing. IEEE Transactions on Vehicular Technology. 2024. vol. 73. no. 6. pp. 8878–8888. DOI: 1109/TVT.2024.3360253.
11. Dankolo N.M., Radzi N.H.M., Mustaffa N.H., Arshad N.I., Nasser M., Gabi D., Yusuf M.N. Optimizing resource allocation for IoT applications in the edge cloud continuum using hybrid metaheuristic algorithms. Scientific Reports. 2024. vol. 15. no. 1. DOI: 10.1038/s41598-025-97648-2.
12. Ismail A.A., Khalifa N.E., El-Khoribi R.A. A survey on resource scheduling approaches in multi-access edge computing environment: A deep reinforcement learning study. Cluster Computing. 2025. vol. 28. no. 3. DOI: 10.1007/s10586-024-04893-7.
13. Filianin I., Kapitonov A., Timoshchuk-Bondar A. Gymnasium Library Interface for Multi Access Edge Computing. Proceedings of 2024 6th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA). 2024. pp. 162–166. DOI: 10.1109/SUMMA64428.2024.10803891.
14. Sun C., Wu X., Li X., Fan Q., Wen J., Leung V.C.M. Cooperative Computation Offloading for Multi-Access Edge Computing in 6G Mobile Networks via Soft Actor Critic. Proceedings of IEEE Transactions on Network Science and Engineering. 2021. vol. 11. no. 6. pp. 5601–5614. DOI: 10.1109/TNSE.2021.3076795.
15. Saad M.M., Jamshed M.A., Adedamola A.I., Nauman A., Kim D. Twin Delayed DDPG (TD3)-Based Edge Server Selection for 5G-Enabled Industrial and C-ITS Applications.

IEEE Open Journal of the Communications Society. 2025. vol. 6. pp. 3332–3343. DOI: 10.1109/OJCOMS.2025.3545566.

16. An L., Wang Z., Yue J., Ma X. Joint Task Offloading and Resource Allocation via Proximal Policy Optimization for Mobile Edge Computing Network. Proceedings of International Conference on Networking and Network Applications (NaNA). 2021. pp. 466–471. DOI: 10.1109/NaNA53684.2021.00087.
17. Zhu L., Tan L., Li B., Tian H. An optimization scheme for vehicular edge computing based on Lyapunov function and deep reinforcement learning. IET Communications. 2024. vol. 18. no. 15. pp. 908–924. DOI: 10.1049/cmu2.12800.
18. Facchini C., Holland O., Granelli F., da Fonseca N.L.S., Aghvami H. Dynamic green self-configuration of 3G base stations using fuzzy cognitive maps. Computer Networks. 2013. vol. 57. no. 7. pp. 1597–1610. DOI: 10.1016/j.comnet.2013.02.011.
19. Filianin I. Lwmecps-gym. GitHub repository. Available at: https://github.com/adeptvin1/lwmecps-gym (accessed 01.08.2025).
20. Filianin I. Lwmecps-testapp. GitHub repository. Available at: https://github.com/adeptvin1/lwmecps-testapp (accessed 03.08.2025).
21. Nieto G., de la Iglesia I., Lopez-Novoa U., Perfecto C. Deep Reinforcement Learning techniques for dynamic task offloading in the 5G edge-cloud continuum. Journal of Cloud Computing. 2024. vol. 13. no. 1. DOI: 10.1186/s13677-024-00658-0.
22. Haarnoja T., Zhou A., Abbeel P., Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. Proceedings of the 35th International Conference on Machine Learning. 2018. vol. 80. pp. 1861–1870.
23. Christodoulou P. Soft actor-critic for discrete action settings. arXiv preprint arXiv:1910.07207. 2019. DOI: 10.48550/arXiv.1910.07207.
24. Ismail A.A., Khalifa N.E., El-Khoribi R.A. A survey on resource scheduling approaches in multi-access edge computing environment: A deep reinforcement learning study. Cluster Computing. 2025. vol. 28. no. 3. DOI: 10.1007/s10586-024-04893-7.
25. Mock J.W., Muknahallipatna S.S. A comparison of PPO, TD3 and SAC reinforcement algorithms for quadruped walking gait generation. Journal of Intelligent Learning Systems and Applications. 2023. vol. 15. no. 1. pp. 36–56. DOI: 10.4236/jilsa.2023.151003.

Филянин Иван Викторович — аспирант, факультет программной инженерии и компьютерной техники, Университет ИТМО. Область научных интересов: граничные вычисления с множественным доступом, мобильные сети связи. Число научных публикаций — 6. adeptvin1@gmail.com; Кронверкский проспект, 49А, 197101, Санкт-Петербург, Россия; р.т.: +7(999)080-3237.

Капитонов Александр Александрович — канд. техн. наук, доцент, заместитель декана, Новый Университет Узбекистана. Область научных интересов: применение технологии блокчейн в робототехнике, автономные системы и беспилотные летательные аппараты, теория управления, мультиагентные системы и интеграция агентов Large Language Model (LLM) в робототехнических системах.. Число научных публикаций — 77. kap2fox@gmail.com; Моваройннахра, 1, 100000, Ташкент, Узбекистан; р.т.: +7(950)000-2896.

Тимощук-Бондарь Артем Игоревич — инженер-программист, факультет программной инженерии и компьютерной техники, Университет ИТМО. Область научных интересов: искусственный интеллект, системы принятия решений. Число научных публикаций — 1. artbondar2003@gmail.com; Кронверкский проспект, 49, 197101, Санкт-Петербург, Россия; р.т.: +7(950)517-2406.