DEC-MAC: A Device-Edge-Cloud Dynamic Resource-Aware Security Task Scheduling Framework Based on Multi-Agent Collaboration

Yinong Chi[1,2], Xiaohui Han[3,1,2]*, Guangqi Liu[1,2], Lin Chen[1,2]
[1]Key Laboratory of Computing Power Network and Information Security, Ministry of Education,
Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences)
[2]Shandong Provincial Key Laboratory of Industrial Network and Information System Security,
Shandong Fundamental Research Center for Computer Science
[3]Quan Cheng Laboratory

***Abstract*—Mobile terminals face challenges in deploying complex security detection mechanisms due to resource constraints. Although the device-edge-cloud collaborative architecture can mitigate some of these limitations, its adaptability in dynamic environments remains inadequate. To address this issue, this paper formulates the security task scheduling problem in device-edge-cloud environments as a Partially Observable Markov Decision Process (POMDP). A reward function is constructed based on security task latency and detection accuracy, transforming the problem into one of maximizing the expected cumulative discounted reward. Subsequently, we propose a scheduling algorithm based on multi-agent reinforcement learning, which employs Multi-Agent Proximal Policy Optimization (MAPPO) to achieve cooperative training. To enhance the state estimation capability of the value network, a value decoupling mechanism is designed. It uses a Siamese network to extract robust feature representations from high-dimensional environmental states, while integrating a hierarchical perceptual contrastive learning mechanism and a reward-guided adaptive weighting mechanism to improve the discrimination of key decision-making states. Simulation results demonstrate that, compared to existing state-of-the-art methods, the proposed framework achieves a better balance between task processing latency and detection accuracy under dynamic workloads and resource variations.**

***Index Terms*—Multi-agent collaboration, Deep reinforcement learning, Security task scheduling, Siamese network**

I. INTRODUCTION

With the widespread adoption of 5G technology, mobile terminals have become essential components of modern information systems. However, due to inherent limitations in computational power, storage capacity, and energy supply, these terminals face significant challenges in deploying complex security mechanisms, rendering them highly vulnerable to diverse attack vectors [1].

Inspired by the edge computing paradigm, recent research has extensively explored Device-Edge-Cloud Collaboration (DECC) architectures [2]. DECC leverages a hierarchical computational infrastructure comprising distributed terminal devices, edge servers, and centralized cloud servers. By offloading network security data—such as network traffic and system logs—to edge and cloud nodes for detection, this approach alleviates the resource constraints of terminal devices.

Corresponding author. E-mail: hanxh@sdas.org.

However, conventional DECC approaches typically rely on static task allocation strategies, where security tasks are pre-assigned to specific device tiers and remain fixed during operation [3]–[11]. Such static paradigms cannot adapt to dynamic changes in resource availability across heterogeneous devices. For instance, over-concentrating detection tasks on the cloud may improve detection granularity but incurs significant latency, whereas assigning excessive tasks to edge devices underutilizes cloud resources and reduces overall detection granularity.

Existing research has attempted to employ heuristic algorithms (e.g., genetic algorithms [12]) to dynamically optimize security task allocation. These methods, however, require the heuristic algorithm to be executed periodically to generate an optimal allocation scheme adapted to the latest resource state. This process involves complex computational iterations, which are not only time-consuming but also exhibit slow responsiveness to rapid resource fluctuations.

To address the limited adaptability of existing methods in dynamic resource environments, we propose DEC-MAC, a multi-agent collaborative scheduling framework for device-edge-cloud systems. It deploys reinforcement learning agents across terminal, edge, and cloud layers, enabling them to dynamically adjust task allocation based on real-time states such as device load and task parameters, and to continuously optimize decisions through reward feedback. The main contributions are summarized as follows:

- We propose a multi-agent collaborative scheduling framework for device-edge-cloud environments that leverages real-time environmental perception to adaptively balance latency and detection accuracy under dynamic conditions.
- We introduce a Siamese network to enhance state representation. By minimizing the feature distance between different augmented views of the same state, the network learns perturbation-robust feature representations. Additionally, we design a reward-guided adaptive weighting

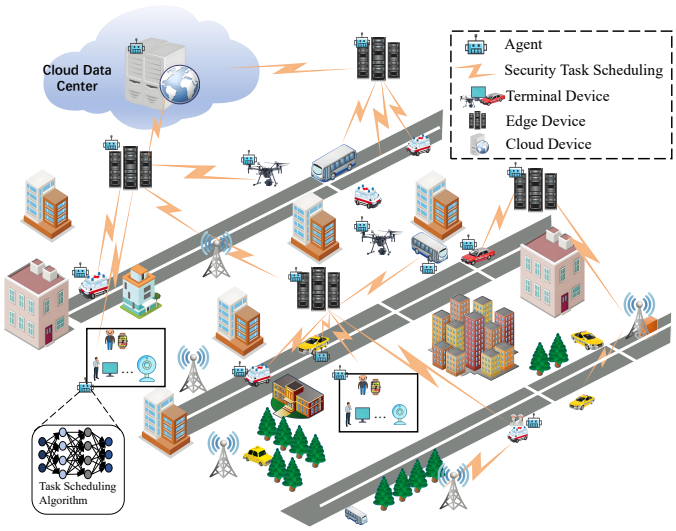


Fig. 1: Illustration of the security task scheduling scenario in the device-edge-cloud collaborative system. The system comprises three layers: terminal, edge, and cloud. Agents at each layer collaborate to determine the dynamic scheduling strategy for security tasks.

mechanism to focus representation learning on critical states with significant value estimation bias, thereby improving feature discriminability.

- We design a hierarchy-aware contrastive learning mechanism that employs an intra-layer consistency loss to promote feature aggregation among states with similar scheduling patterns by aligning their representations. Concurrently, an inter-layer collaboration loss maintains correlations between features of different dominant types, enabling the learning of representations that reflect a hierarchical division of labor while supporting cross-layer cooperation.

To facilitate future research, our code is open-sourced at https://github.com/C-Vencient/DEC-MAC.

II. SYSTEM MODEL

A. System Composition

This study considers a three-tier device-edge-cloud collaborative computing architecture, as illustrated in Figure 1. The system comprises three hierarchical device sets: the terminal device set $\mathcal{D}$, the edge device set $\mathcal{E}$, and the cloud device set $\mathcal{C}$. A specific device instance is denoted by d_i, where $i \in \mathbb{Z}^+$ is a unique positive integer index. The mapping from an index i to its corresponding device type is defined as follows:

$$d_i \in \begin{cases} \mathcal{D}, & \text{if } i \in [1, |\mathcal{D}|] \\ \mathcal{E}, & \text{if } i \in [|\mathcal{D}|+1, |\mathcal{D}|+|\mathcal{E}|] \\ \mathcal{C}, & \text{if } i \in [|\mathcal{D}|+|\mathcal{E}|+1, |\mathcal{D}|+|\mathcal{E}|+|\mathcal{C}|] \end{cases}, \quad (1)$$

where $|\cdot|$ denotes set cardinality. For each device d_i, its computational capability is denoted by $d_i.f$, measured in CPU cycles per unit time. The system operates over discrete time slots indexed by $t \in \{1, 2, ..., T\}$, where each slot represents a fixed interval for task generation and scheduling decision execution.

B. Task Scheduling Model

In each time slot t, $m_t^{d_i} \in \mathbb{Z}^+$ security tasks are generated at each terminal device $d_i \in \mathcal{D}$ and enter its local task queue Q_i, where an agent determines their target execution devices. Similarly, for each edge or cloud device $d_m \in \mathcal{E} \cup \mathcal{C}$, $w_t^{d_m} \in \mathbb{Z}^+$ tasks await detection in slot t and enter queue Q_m. An agent associated with d_m then decides whether to schedule these tasks to other devices or execute them locally.

For a specific security task j, let g_j denote its required CPU cycles and k_j its data volume. The total execution latency l_j^{total} consists of three components: transmission latency l_j^{up}, computation latency l_j^c, and queuing latency l_j^{wait}. Since a task may be transmitted across multiple hops, we define its scheduling path as a sequence of devices $q = (d_i, ..., d_m)$, where d_i is the source terminal device, $d_m \in \mathcal{E} \cup \mathcal{C}$ is the final execution device, and H_j is the number of transmission hops. The transmission latency is:

$$l_j^{up} = \sum_{k=0}^{H_j-1} \frac{k_j}{B_t^{q[k],q[k+1]}}, \quad q[0] \in \mathcal{D}, q[H_j] \in \mathcal{E} \cup \mathcal{C}, \quad (2)$$

where $B_t^{d_i,d_m}$ denotes the transmission rate between devices d_i and d_m in time slot t. The computation latency on the execution device d_m is

$$l_j^c = \frac{g_j}{(d_m.f) \cdot c_j}, \quad d_m \in \mathcal{E} \cup \mathcal{C}, \quad (3)$$

where c_j is the ratio of CPU cycles allocated to task j on d_m. Thus, the total latency for task j is: $l_j^{total} = l_j^{up} + l_j^c + l_j^{wait}$.

Let F_t be the set of security tasks completed within time slot t. The average latency of security tasks in time slot t is then given by: $l_t^{avg} = \frac{1}{|F_t|} \sum_{j \in F_t} l_j^{total}$, while the average detection accuracy during this slot is expressed as: $acc_t^{avg} = \frac{y_t}{|F_t|}$, where y_t is the number of correct classifications. The security task scheduling problem addressed in this paper is to find an optimal strategy for scheduling tasks generated in each time slot, with the objective of minimizing average processing latency while maximizing average detection accuracy.

III. APPROACH

A. POMDP Modeling

In the considered device-edge-cloud collaboration environment, full global state synchronization is prohibitively costly, so each device must rely on local observations. To capture this partial observability, we formulate the security task scheduling problem as a Partially Observable Markov Decision Process (POMDP), defined by the tuple $(\mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{R})$, where $\mathcal{N}$ is the number of agents, $\mathcal{S}$ the global state space, $\mathcal{A}$ the joint action space, and $\mathcal{R}$ the reward space.

State: At time slot t, the state $s_t \in \mathcal{S}$ comprises the observations of all agents: $s_t = \{o_{1,t}, ..., o_{|\mathcal{D}|+|\mathcal{E}|+|\mathcal{C}|,t}\}$. Each observation $o_{i,t} = \{|Q_i|_t, L_t^{oth}, U_t^{cpu}, B_t^i\}$, where $|Q_i|_t$ denotes the task queue length of device d_i at time slot t, L_t^{oth} is the set of queue lengths of all edge and cloud devices, U_t^{cpu} is the set of their CPU utilization rates, and B_t^i is the set of available bandwidths between d_i and all edge/cloud devices.

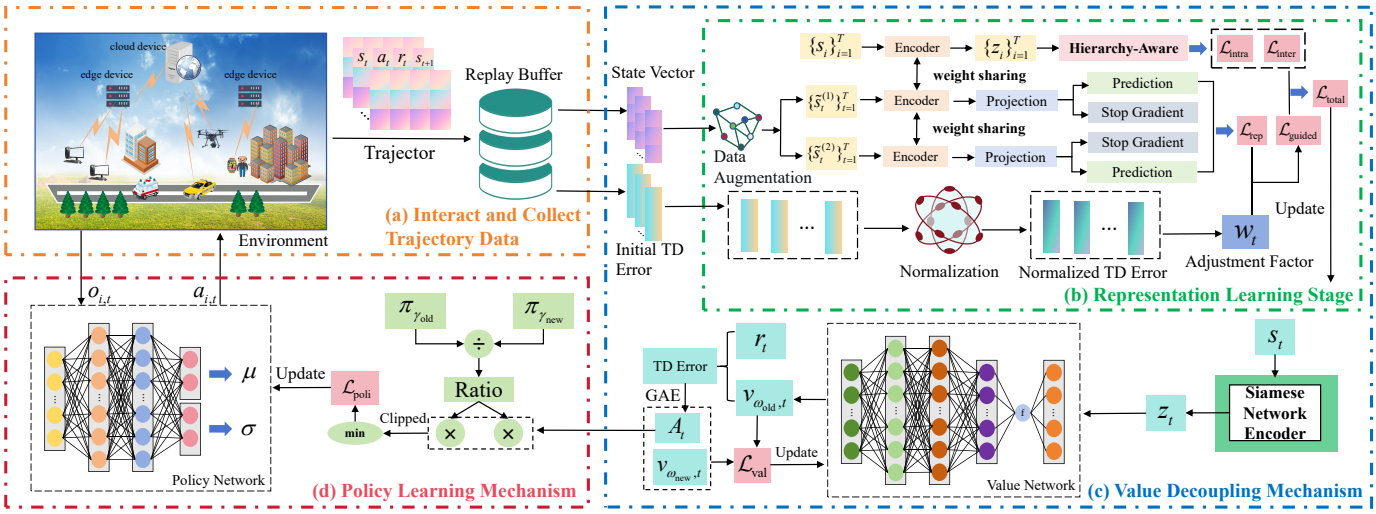


Fig. 2: Overview of the DEC-MAC framework. (a) Agent–environment interaction and experience collection. (b) Representation learning stage. (c) Value decoupling mechanism. (d) Policy learning mechanism.

Action: The joint action $a_t \in \mathcal{A}$ of the agents at time slot t is defined as the set of individual actions of all agents. $a_t = \{a_{1,t}, .., a_{|\mathcal{D}|+|\mathcal{E}|+|\mathcal{C}|,t}\}$. Each individual action $a_{i,t}$ is represented as $a_{i,t} = \left\{a_{i,t}^{|\mathcal{D}|+1}, ..., a_{i,t}^{|\mathcal{D}|+|\mathcal{E}|+|\mathcal{C}|}\right\}$, where $a_{i,t}^m$ denotes the proportion of tasks allocated from device d_i to device d_m, with $d_i \in \mathcal{D} \cup \mathcal{E} \cup \mathcal{C}$ and $d_m \in \mathcal{E} \cup \mathcal{C}$.

Reward: The reward $r_t \in \mathcal{R}$ for the agents at time slot t is defined as the linear weighted sum of the average latency and the average detection accuracy of all completed security tasks at that time slot, expressed as:

$$r_t = -\alpha \cdot l_t^{avg} + (1-\alpha) \cdot acc_t^{avg}, \tag{4}$$

where $\alpha \in [0,1]$ is a weighting coefficient that balances the two objectives. The negative sign on latency aligns the goal of minimizing latency with the reward maximization objective.

Based on this POMDP formulation, the security task scheduling problem is to find an optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward:

$$\pi^* = \text{Max } \mathbb{E}\left[\sum_{t=1}^{T} \gamma^t r_t | \pi\right], \tag{5}$$

where $\gamma \in (0,1)$ is the discount factor, characterizing the importance of immediate rewards relative to future rewards.

B. DEC-MAC

Traditional multi-agent DRL employs neural networks for state-value approximation. However, significant heterogeneity in computing, storage, and bandwidth among terminals, edge, and cloud forces agents to simultaneously account for local device states, cross-layer collaborator states, and global network context, creating a high-dimensional state space. This complexity challenges traditional neural networks in extracting decision-critical features, leading to biased value estimates and compromised policy convergence.

To address this, we propose DEC-MAC, a MAPPO-based framework introducing a value decoupling mechanism that explicitly decomposes state value estimation into two stages: representation learning, which maps high-dimensional states into robust features, and value prediction, which estimates state value based on the learned representations, as shown in Figure 2(c). Agents execute actions based on observations, receive rewards, and transition to new states, storing experience tuples (s_t, a_t, r_t, s_{t+1}) in a replay buffer for joint training of both modules, as depicted in Figure 2(a).

1) Representation Learning Stage: The representation learning stage employs a Siamese network, the architecture of which is illustrated in Figure 2(b). Specifically, given the global state s_t, we generate two augmented views $\tilde{s}_t^{(1)}$ and $\tilde{s}_t^{(2)}$, processed by a shared encoder f_θ to extract features:

$$z_t^{(1)} = f_\theta(\tilde{s}_t^{(1)}), \quad z_t^{(2)} = f_\theta(\tilde{s}_t^{(2)}). \tag{6}$$

These features are then projected into a latent space by a projector g_ϕ and further transformed by a predictor h_ψ:

$$p_t^{(1)} = h_\psi(g_\phi(z_t^{(1)})), \quad p_t^{(2)} = h_\psi(g_\phi(z_t^{(2)})). \tag{7}$$

The Siamese network is trained with a consistency loss that maximizes the similarity between the projections of the two augmented views of the same state:

$$\mathcal{L}_{\text{rep}} = \frac{1}{2}\left[D(p_t^{(1)}, \text{sg}(g_\phi(z_t^{(2)}))) + D(p_t^{(2)}, \text{sg}(g_\phi(z_t^{(1)})))\right], \tag{8}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operation.

To enable representation learning to focus on critical states with large value estimation errors, we introduce a reward-guided adaptive weighting mechanism. Specifically, for a given state s_i with reward r_i, we compute the temporal-difference error δ_i and use its absolute value $e_i = |\delta_i|$ as an importance indicator of value estimation bias. These TD-error magnitudes are then normalized and transformed into importance weights via a tunable temperature parameter τ:

$$w_i = \frac{\exp\left(\frac{e_i}{\tau}\right)}{\sum_{j=1}^{T} \exp\left(\frac{e_j}{\tau}\right)}. \tag{9}$$

The weighted representation loss is $\mathcal{L}_{\text{guided}} = \sum_{i=1}^{T} w_i \cdot \ell_i$, where ℓ_i is the consistency loss for state s_i.

The device-edge-cloud system has a natural hierarchy, where cross-layer differences in resources and functions result in varying scheduling characteristics such as latency and accuracy. Feeding the global state directly into the value network mixes cross-layer information, hindering the extraction of structured decision-making features. To address this, we design a hierarchy-aware contrastive learning mechanism that groups states by dominant type, namely edge or cloud. Each state s_i is assigned a label $c_i \in \{0, 1\}$, with 0 indicating edge-dominance and 1 indicating cloud-dominance. Let $z_i = f_\theta(s_i)$ and let $\mathcal{G}_c$ be the index set of samples with dominant type c in a batch. The intra-layer loss enforces intra-group feature similarity:

$$\mathcal{L}_{\text{intra}} = \sum_{c\in\{0,1\}} \mathbb{E}_{i,j\sim G_c} \left[1 - \frac{z_i}{\|z_i\|_2} \cdot \frac{z_j}{\|z_j\|_2}\right]. \tag{10}$$

The inter-level collaboration loss maintains a target correlation between the features of the two dominant types of states to facilitate cross-layer cooperation. The mean feature vector for each type is computed as $\mu_c = \mathbb{E}_{i\sim\mathcal{G}_c}[z_i]$, and the loss is defined as:

$$\mathcal{L}_{\text{inter}} = \left(\frac{\mu_0}{\|\mu_0\|_2} \cdot \frac{\mu_1}{\|\mu_1\|_2} - \rho\right)^2, \tag{11}$$

where ρ controls the desired correlation. The total loss for representation learning is: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{guided}} + \lambda_{\text{intra}}\mathcal{L}_{\text{intra}} + \lambda_{\text{inter}}\mathcal{L}_{\text{inter}}$, where λ_{intra} and λ_{inter} are balancing coefficients.

2) Value and Policy Learning: After pre-training the encoder f_θ, we extract a robust state representation $z_t = f_\theta(s_t)$. A value network $v_{\omega,t}$, implemented as a two-layer MLP, estimates the state value: $v_{\omega,t} = W_2 \cdot \text{ReLU}(W_1 z_t + b_1) + b_2$.

The advantage A_t is estimated using Generalized Advantage Estimation (GAE) [13]. The value network is trained by minimizing the mean squared error against the target value derived from rewards and bootstrapping.

The policy network, illustrated in Figure 2(d), maps an agent's local observation $o_{i,t}$ to parameters of a Gaussian distribution over its continuous action space. For agent i:

$$h = \text{ReLU}\left(W_4\text{ReLU}\left(W_3 o_{i,t} + b_1\right) + b_2\right), \tag{12}$$

$$\mu = \tanh(W_\mu h + b_\mu), \quad \sigma = \text{softplus}(W_\sigma h + b_\sigma). \tag{13}$$

The policy π_γ is expressed as $\pi_\gamma(\cdot \mid o_{i,t}) = \mathcal{N}(\mu_i, \sigma_i^2)$, and actions are sampled as $a_{i,t} \sim \pi_\gamma(\cdot \mid o_{i,t})$. The policy is updated via proximal policy optimization, which clips the probability ratio $pr_t(\gamma) = \frac{\pi_{\gamma_{\text{new}}}(a_{i,t}|o_{i,t})}{\pi_{\gamma_{\text{old}}}(a_{i,t}|o_{i,t})}$ to $[1-\epsilon, 1+\epsilon]$. The objective function is:

$$\begin{aligned}\mathcal{L}_{\text{poli}} = \mathbb{E}_t\Big[&\min\big(pr_t(\gamma)A_t,\ \text{clip}(pr_t(\gamma), 1-\epsilon, 1+\epsilon)A_t\big) \\ &- \beta\, H\big(\pi_\gamma(\cdot \mid o_{i,t})\big)\Big],\end{aligned} \tag{14}$$

where $H(\pi_\gamma(\cdot \mid o_{i,t}))$ is the entropy regularization term, and β is its coefficient.

IV. EXPERIMENTS AND ANALYSIS

A. Experimental Setup

Security tasks are dynamically generated by terminal devices in each time slot. To account for computational heterogeneity, lightweight machine learning models are deployed on edge devices, while more complex deep learning models run on the cloud. We evaluate two model pairings: 1) a Support Vector Machine (SVM) [14] on the edge paired with a CNN-BiLSTM model (CBNID) [15] on the cloud; and 2) a Decision Tree (DT) [16] on the edge paired with a gated self-attention BiGRU model (SAGB) [17] on the cloud. Key simulation parameters are listed in Table I.

TABLE I: Simulation Parameters

Parameters	**Value**
Number of terminal devices	{15, 20, 25, 30, 35, 40}
Number of edge devices	3
Number of cloud devices	1
CPU computational capability of edge device	[10, 20] GHz
CPU computational capability of cloud device	[30, 40] GHz
CPU cycles required for task execution	$[1 \times 10^8, 5 \times 10^8]$
Bandwidth	[30, 60] Mbps
Task arrival rate (per terminal device)	0–5, 5–10, 10–15 tasks/slot
Discount factor γ	0.99
GAE parameter λ	0.95
Temperature parameter τ	1.5

B. Baselines

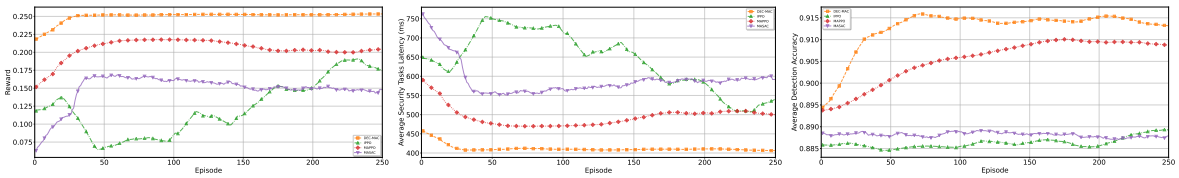
We compare DEC-MAC against the following methods: 1) FCEIDS [7]: A static edge-first strategy that offloads tasks to the cloud only when edge resources are exhausted. 2) VECID [12]: A dynamic heuristic method using NSGA-II to select execution devices by optimizing latency, energy, and survivability. 3) MAPPO [18]: A multi-agent PPO algorithm with clipped gradients for stable coordination. 4) MASAC [19]: A multi-agent soft actor-critic algorithm that incorporates maximum entropy for exploration. 5) IPPO [20]: An independent PPO method where agents learn policies without explicit coordination. 6) Greedy: A heuristic that always schedules tasks to the currently most available resource.

C. Convergence Analysis

As shown in Figure 3, DEC-MAC achieves higher immediate reward, better balances latency and accuracy, and improves policy learning and convergence efficiency via its value decoupling mechanism, which extracts robust, collaboration-aware representations from high-dimensional state space. In contrast, mainstream multi-agent methods like MAPPO and MASAC extract limited robust features in dynamic, complex settings, reducing policy learning efficiency, while IPPO, lacking cooperative mechanisms, struggles with coordination and shows slower convergence and limited performance.

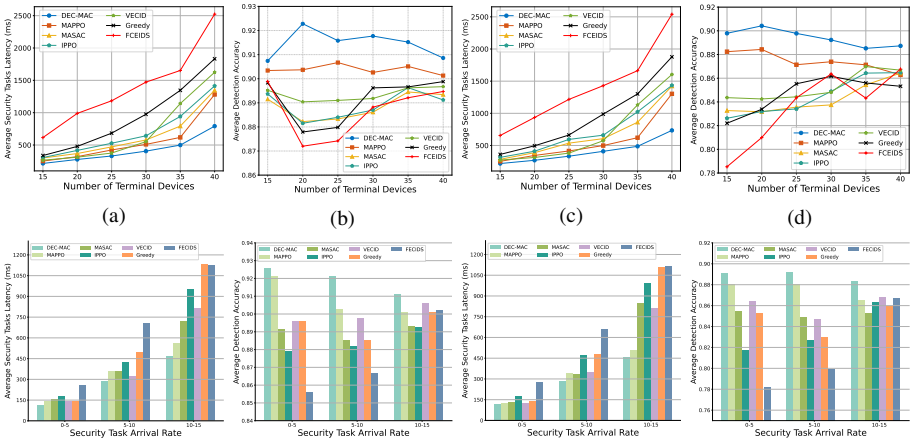
D. Performance Comparison

We examined the effect of environmental complexity on scheduling efficiency for DEC-MAC and baseline algorithms. As shown in Figure 4, DEC-MAC consistently outperforms all baselines. Experimental results indicate that as the number of terminal devices and security task arrival rates increase, the average delay of all scheduling methods rises. This is primarily



(a) (b) (c)

Fig. 3: Comparative Analysis of Convergence between DEC-MAC and Baseline Methods.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 4: Performance comparison with baseline methods. (a)-(d): Average latency and detection accuracy versus number of terminal devices for SUM+CBNID (a,b) and DT+SAGB (c,d). (e)-(h): Same metrics versus security task arrival rate for SUM+CBNID (e,f) and DT+SAGB (g,h).

because more tasks are processed per unit time at the edge and cloud layers, and limited computational resources lead to longer queuing delays.

Under small-scale scenarios, static scheduling methods such as FCEIDS and Greedy achieve low latency. However, as the number of terminals and tasks increases, these methods experience a significant rise in scheduling delay due to their inability to dynamically perceive environmental changes or adjust strategies accordingly. The heuristic algorithm VECID is constrained by iteration limits under high environmental complexity, making it difficult to obtain optimal solutions and resulting in increased latency. Reinforcement learning approaches such as MAPPO, MASAC, and IPPO enable adaptive scheduling, but their capacity to interpret high-dimensional state vectors diminishes as environmental complexity grows, adversely affecting state representation and agent training effectiveness. In contrast, DEC-MAC employs a value decoupling mechanism to capture complex dependencies between resource dynamics and scheduling decisions in high-dimensional states, enhancing representational capacity and maintaining superior performance in complex environments.

E. Ablation Study

We conduct an ablation study to analyze the contribution of each component by comparing the full DEC-MAC model with the following variants: Three ablation models were designed: 1) DEC-MAC w/o HA: Removes the hierarchy-aware contrastive learning mechanism. 2) DEC-MAC w/o HA & RGW: Removes both the hierarchy-aware mechanism and the reward-guided weighting. 3) DEC-MAC w/o VD: Removes the entire value decoupling mechanism; agents learn directly from raw state vectors. Results are shown in Figure 5. The reward curve of the full DEC-MAC eventually stabilizes at 0.252, while DEC-MAC w/o HA and DEC-MAC w/o HA & RGW stabilize at 0.249 and 0.240, respectively. In contrast, DEC-MAC w/o VD stabilizes at a lower reward of 0.207 compared to the complete DEC-MAC.

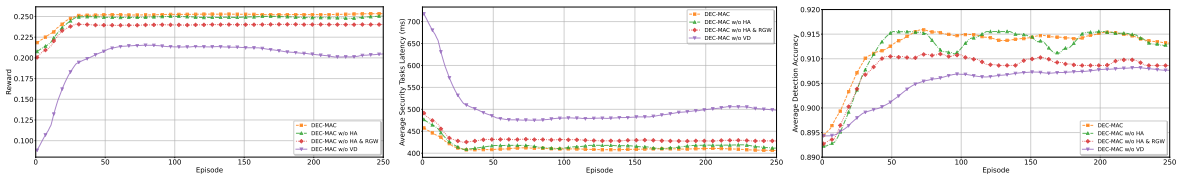
This confirms that the Siamese network is key to extracting decision-relevant features. The reward-guided weighting mechanism improves discriminability by focusing on states with significant value estimation errors. The hierarchy-aware mechanism promotes intra-tier feature coherence and cross-tier correlation for collaboration. Without this representation learning, DEC-MAC w/o VD exhibits lower initial rewards, slower convergence, and worse final performance.

F. Visualization Comparison

To examine the learned representations, we project both raw state vectors and the encoder-processed feature vectors into a 2D space using t-SNE. As illustrated in Figure 6, the feature embeddings, after being processed by the encoder, form a more compact clustering distribution in the latent space. This enables the policy network and value function to learn based on clearer and more discriminative features.

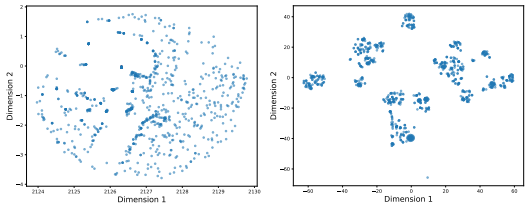
V. CONCLUSION

This work proposes DEC-MAC, a multi-agent collaborative scheduling framework for adaptive security task allocation



(a) (b) (c)

Fig. 5: Performance comparison between DEC-MAC and its variants.



(a) (b)

Fig. 6: Effectiveness Verification of Representation Learning. (a) Raw state vector. (b) Feature vector processed by the Siamese network encoder.

in dynamic device-edge-cloud environments. A novel value decoupling mechanism integrates Siamese networks for robust state representation, reward-guided adaptive weighting for learning on critical states, and hierarchy-aware contrastive learning for capturing cross-layer patterns. Proximal policy optimization enables multi-agent policy learning. Ablation studies validate each component. Future work will explore privacy-preserving techniques for collaborative data security.

VI. ACKNOWLEDGMENT

This work was supported in part by the Shandong Provincial Key Research and Development Program (Major Scientific and Technological Innovation Project) under Grant 2025CXGC010106 and 2024CXGC010111, in part by the Taishan Scholars Program under Grant tspd20240814, in part by the Major Project of Quancheng Provincial Laboratory under Grant QCL20250103, in part by the Research Project of Provincial Laboratory of Shandong under Grant SYS202201, and in part by the "20 New Universities" Project of Jinan City under Grant 202333023.

REFERENCES

[1] W. Hu, Q. Cao, M. Darbandi, and N. Jafari Navimipour, "A deep analysis of nature-inspired and meta-heuristic algorithms for designing intrusion detection systems in cloud/edge and iot: state-of-the-art techniques, challenges, and future directions," *Cluster Computing*, vol. 27, no. 7, pp. 8789–8815, 2024.

[2] Y. Wang, C. Yang, S. Lan, L. Zhu, and Y. Zhang, "End-edge-cloud collaborative computing for deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 4, pp. 2647–2683, 2024.

[3] M. Wan, J. Fang, C. Guo, L. Geng, Y. Liu, W. Ma, C. Xu, and M. Su, "A federated learning-based intrusion detection system for satellite-terrestrial integrated networks," in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.

[4] Y. Xu, Y. Li, and Z. Sun, "Iot intrusion detection using sdn cloud-edge collaboration and ensemble deep learning," in *2024 IEEE Cyber Science and Technology Congress (CyberSciTech)*. IEEE, 2024, pp. 195–200.

[5] C. Fan, J. Cui, H. Jin, H. Zhong, I. Bolodurina, and D. He, "Auto-updating intrusion detection system for vehicular network: a deep learning approach based on cloud-edge-vehicle collaboration," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 10, pp. 15 372–15 384, 2024.

[6] R. Yang, H. He, Y. Xu, B. Xin, Y. Wang, Y. Qu, and W. Zhang, "Efficient intrusion detection toward iot networks using cloud–edge collaboration," *Computer Networks*, vol. 228, p. 109724, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138912862300169X

[7] F. Khan, M. A. Jan, R. Alturki, M. D. Alshehri, S. T. Shah, and A. ur Rehman, "A secure ensemble learning-based fog-cloud approach for cyberattack detection in iomt," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 10, pp. 10 125–10 132, 2023.

[8] S. Baimukhanov, H. Ali, and A. Yazici, "Enhancing ML-based anomaly detection in data management for security through integration of IoT, cloud, and edge computing," *Expert Systems with Applications*, vol. 293, p. 128700, Dec. 2025.

[9] X. Deng, H. Tang, X. Pei, D. Li, and K. Xue, "Mdhe: A malware detection system based on trust hybrid user-edge evaluation in iot network," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 5950–5963, 2023.

[10] J. Pei, Y. Hu, L. Tian, X. Pei, and Z. Wang, "Dynamic anomaly detection using in-band network telemetry and gcn for cloud–edge collaborative networks," *Computers & Security*, vol. 154, p. 104422, 2025.

[11] S. Li, Y. Lu, and J. Li, "Cad-ids: A cooperative adaptive distributed intrusion detection system with fog computing," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2022, pp. 635–640.

[12] A. Mourad, H. Tout, O. A. Wahab, H. Otrok, and T. Dbouk, "Ad hoc vehicular fog enabling cooperative low-latency intrusion detection," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 829–843, 2020.

[13] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[14] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE access*, vol. 6, pp. 33 789–33 795, 2018.

[15] J. Sinha and M. Manollas, "Efficient deep cnn-bilstm model for network intrusion detection," in *Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition*, 2020, pp. 223–231.

[16] J. Suganthi, B. Nagarajan, and S. Muhtumari, "Network anomaly detection using hybrid deep learning technique," in *Advances in Parallel Computing Algorithms, Tools and Paradigms*. IOS Press, 2022, pp. 103–109.

[17] Z. Hu, G. Liu, Y. Li, and S. Zhuang, "Sagb: self-attention with gate and bigru network for intrusion detection," *Complex & Intelligent Systems*, vol. 10, no. 6, pp. 8467–8479, 2024.

[18] Q. Wang, S. Zou, Y. Sun, M. Liwang, X. Wang, and W. Ni, "Toward intelligent and adaptive task scheduling for 6g: an intent-driven framework," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 5, pp. 1975–1988, 2024.

[19] Y. Wang, H. Wu, R. H. Jhaveri, and Y. Djenouri, "Drl-based urllc-constraint and energy-efficient task offloading for internet of health things," *IEEE Journal of Biomedical and Health Informatics*, vol. 28, no. 6, pp. 3305–3316, 2024.

[20] L. Qin, H. Lu, Y. Chen, B. Chong, and F. Wu, "Toward decentralized task offloading and resource allocation in user-centric mec," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11 807–11 823, 2024.