MTCM: Refining Privacy-Aware Task Offloading with HGSA in Multi-Tier Computing System for Emerging Next-Generation Wireless Networks -Based Predictor

R. Udaya Nirmala Mary[1], R. Santhosh[2,*]

[1]Research scholar, Department of Computer Science and Engineering, Faculty of Engineering, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India

[2]Professor and Head, Department of Computer Science and Engineering, Faculty of Engineering, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India

Emails: udaya.7it@gmail.com; santhoshrd@gmail.com

Abstract

Multi-cloud computing is emerging as a transformative solution to meet the extensive computational demands of Internet of Things (IoT) devices. In networks with multiple devices and clouds, factors such as real-time computing requirements, fluctuating wireless channel conditions, and dynamic network scales introduce significant complexity. Addressing these challenges, along with the resource constraints of IoT devices, is essential for effective multi-cloud integration. This paper proposes a hybrid decision-offloading model that integrates continuous and discrete decision-making. IoT devices must learn to make coordinated decisions regarding cloud server selection, task offloading ratios, and local computation capacity. This dual-layer decision-making process involves managing both continuous and discrete variables, along with inter-device coordination, which poses considerable challenges. To address these, we introduce a probabilistic approach that transforms discrete actions, such as selecting a cloud server, into a continuous domain. We further develop a Privacy-Aware Multi-Agent Deep Reinforcement Learning (PA-MADRL) framework that combines centralized training with distributed execution. This framework minimizes overall system costs by considering energy consumption and cloud server rental fees. Each IoT device operates as an agent, autonomously learning efficient policies while alleviating its computational burden. Experimental results demonstrate that the PA-MADRL framework effectively adapts to dynamic network conditions, learning optimal offloading policies. It significantly outperforms four state-of-the-art deep reinforcement-learning models and two heuristic methods, achieving lower system costs and improved resource efficiency.

Keywords: Task Offloading; Privacy; Deep Reinforcement Learning; Wireless Network; Internet of Things (IoT)

1. Introduction

The progression of wireless networks towards next-generation technologies has fundamentally reshaped the landscape of the Internet of Things (IoT) [1]. This transformation is not merely incremental but represents a paradigm shift, giving rise to what is now recognized as Intelligent IoT (iIoT) [2]. As IoT applications become more sophisticated, their requirements for seamless, low-latency services have become more demanding, reflecting a significant departure from traditional IoT models [3]. This shift introduces complex requirements related to delay and jitter, which are critical for maintaining the performance and reliability of intelligent applications [4]. The concept of iIoT highlights the need for real-time, intelligent services that cater to a diverse and growing user base. Unlike traditional IoT, which often operates with more lenient performance criteria, iIoT applications are highly sensitive to delays and fluctuations in service quality. These applications span various critical domains, including autonomous driving, industrial automation, and augmented reality. In these contexts, even minor delays or variations in performance can severely affect user experience and system effectiveness. For instance, autonomous vehicles require instantaneous data processing to make split-second decisions, while augmented reality applications depend on minimal latency to deliver immersive experiences [5].

To address these evolving demands, multi-tier computing has emerged as a crucial solution [6]. Multi-tier computing integrates edge computing and fog computing within the traditional cloud computing framework, creating a more flexible and efficient

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

computational environment [7]. This approach leverages the strengths of each computing layer to enhance overall performance and meet the stringent requirements of iIoT applications [8]. Edge computing involves placing computational resources closer to the source of data generation, which can be sensors, devices, or other endpoints [9]. Edge computing processes data locally or on nearby infrastructure, drastically reducing latency and enhancing system responsiveness [10]. This close proximity to data sources is especially advantageous for applications that demand real-time processing and decision-making. For instance, in smart manufacturing, edge computing facilitates real-time monitoring and control of machinery, boosting operational efficiency and minimizing downtime [11]. Building on this, fog computing expands edge computing by adding intermediate layers of processing and storage between edge devices and the central cloud [12]. This hierarchical approach enables more distributed data management, enhancing scalability and efficiency in complex networks [13]. This distributed architecture provides additional processing power and storage capabilities closer to the edge, thereby improving scalability and resource utilization. Fog computing acts as an intermediary that helps balance the computational load between the edge and the cloud, further reducing latency and improving system resilience [14]. The integration of edge and fog computing within a multi-tier system creates a more nuanced approach to task management and resource allocation [15]. By distributing computing tasks across various nodes, from the cloud to edge devices, multi-tier computing systems can optimize performance and manage workloads more effectively. This distribution helps in balancing the computational load, preventing bottlenecks, and reducing the risks associated with centralized computing models, such as single points of failure. Task offloading is a critical component of multi-tier computing. It involves transferring computational tasks from one layer to other based-on factors such as current load, latency requirements, and resource availability [16]. In a multi-tier computing system, tasks are offloaded from the cloud to edge or fog nodes to optimize processing efficiency and meet the specific needs of iIoT applications.

This dynamic approach to task management ensures that tasks are executed in the most appropriate layer of the computing hierarchy [17]. For instance, tasks requiring immediate processing are offloaded to edge devices, while fog nodes or the cloud [18] can handle more complex or less time-sensitive tasks. This strategy not only enhances performance but also ensures that resources are utilized effectively across the entire system [19]. As iIoT continues to evolve, the importance of multi-tier computing will only grow. The ability to effectively manage and allocate computational resources across diverse nodes will be essential for supporting the increasing sophistication of iIoT applications. Multi-tier computing not only addresses immediate needs for low-latency and high-performance but also provides a scalable framework for future advancements in wireless networks and intelligent systems. The transition towards Intelligent IoT represents a significant shift in how we approach computing and resource management [20]. Multi-tier computing offers a comprehensive solution by integrating edge and fog computing into the traditional cloud framework. This integration provides the necessary infrastructure to meet the rigorous performance requirements of iIoT applications. As we look ahead, the continued development and refinement of multi-tier computing systems will be crucial for ensuring that next-generation wireless networks can support the increasingly sophisticated demands of intelligent, real-time applications. The evolution of multi-tier computing will play a pivotal role in shaping the future of IoT, enabling the delivery of high-performance, low-latency services that are essential for the next generation of wireless technologies. The integration of multi-tier computing into the iIoT framework offers several key benefits:

Enhanced Low-Latency Performance: By processing data closer to its source, multi-tier computing systems can meet stringent low-latency requirements. This proximity reduces the time needed for data transmission and processing, which is crucial for applications demanding real-time responses. Improved Scalability and Flexibility: Multi-tier computing systems can handle varying loads and resource demands more efficiently. The distributed architecture allows for scalable resource management, accommodating the growing complexity and scale of iIoT applications. Increased System Resilience and Reliability: Distributing tasks and resources across multiple layers enhances overall system resilience. This distribution minimizes the impact of potential failures or performance issues in any single component, leading to a more reliable and robust system. Here are the contributions of the paper:

Hybrid Decision Offloading Model: Introduces a novel hybrid decision-offloading model that integrates both continuous and discrete decision-making processes to address the dynamic nature of multi-cloud environments for IoT devices.

Probabilistic Approach for Discrete Actions: Develops a probabilistic method to transform discrete decisions, such as cloud server selection, into a continuous range, thereby simplifying the decision-making process and enhancing flexibility.

Multi-Agent Deep Reinforcement Learning Framework: Proposes a PA-MARL framework that employs centralized training and distributed execution to optimize system cost, balancing energy consumption and cloud server rental fees.

Empirical Validation: Demonstrates through experimental results that the PA-MADRL framework effectively learns dynamic offloading policies, outperforming four state-of-the-art deep reinforcement-learning agents and two heuristic algorithms in minimizing system costs.

2. Related Works

The designated vehicles complete the assigned tasks Author in [21], presents a novel approach to computation offloading in heterogeneous IoT networks using DRL. The authors effectively leverage DRL to dynamically adapt offloading decisions,

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

accommodating the diverse and dynamic nature of IoT networks. The study is well structured, with comprehensive experiments that demonstrate significant improvements in offloading efficiency and performance. However, while the approach shows promise, the complexity of the DRL model and its training requirements may pose practical implementation challenges. Future work could explore simplifying the model and reducing the computational overhead for real-world applications. Author in [22], focuses on reducing both delay and cost in computation offloading within IoT edge computing environments. The paper presents a clear and concise formulation of the problem, followed by an innovative solution that balances these critical metrics. The proposed method is validated through extensive simulations, showing a substantial decrease in both delay and cost compared to traditional methods. One limitation is the assumption of perfect knowledge of network conditions, which may not always be feasible in real-world scenarios. Future research could address this by incorporating predictive models or adaptive mechanisms to handle uncertainty in network conditions. In [23], the authors present an advanced deep learning-based approach for computational offloading in multi-level IoT edge-cloud computing networks. The multi-tier architecture effectively distributes computation tasks between edge and cloud resources, optimizing resource utilization and reducing latency. The use of deep learning models to predict offloading decisions enhances the system's adaptability to varying network conditions. The experimental results are compelling, showing marked improvements over existing methods. However, the reliance on extensive training data for the deep learning model could be a drawback, and further exploration into reducing data dependency would be beneficial. Author in [24], explores partial computation offloading combined with adaptive task scheduling in 5G-enabled IoT networks. The approach allows for more flexible and efficient use of network resources, taking advantage of 5G's high bandwidth and low latency. The authors provide a detailed analysis of the system's performance, demonstrating significant enhancements in both computational efficiency and task completion times. The partial offloading strategy is particularly innovative, offering a balanced trade-off between local computation and offloading. However, the paper could benefit from a deeper discussion on the potential impact of varying IoT mobility patterns on the proposed system's performance. Author in [25], addresses the critical issue of energy efficiency in IoT edge cloud computing through optimized computation offloading. The authors present a comprehensive model that minimizes energy consumption while maintaining computational performance. The proposed solution incorporates both energy and latency considerations, providing a balanced approach to offloading decisions. Experimental results show significant energy savings compared to conventional methods. One area for improvement could be the exploration of real-time implementation challenges, as the current model assumes ideal conditions. Future work could investigate adaptive algorithms that respond to real-time changes in IoT network environments.

Author in [26], presents an intelligent task-offloading framework for IoT edge computing networks, employing advanced algorithms to optimize task distribution. The approach leverages machine learning to predict optimal offloading strategies, enhancing the efficiency and performance of the network. The authors provide a robust experimental evaluation, demonstrating significant improvements in task execution times and resource utilization. However, the reliance on accurate data inputs and the potential complexity of the machine learning models may limit scalability. Future work could focus on improving the model's adaptability to varying network conditions and reducing computational overhead. Author in [27], addresses the crucial issue of energy efficiency in computation offloading for IoT edge computing networks. The authors propose a novel approach that minimizes energy consumption while maintaining computational performance. Their methodology integrates energy-aware algorithms that dynamically adjust offloading decisions based on real-time energy metrics. The results indicate substantial energy savings and improved system efficiency. Nonetheless, the paper could benefit from a deeper exploration of the trade-offs between energy savings and potential increases in computational latency. Further research could also examine the impact of different IoT mobility patterns on energy consumption. In this paper [28], the authors propose a blockchain-based framework to enhance the security of computation offloading in IoT networks. The approach leverages the decentralized and immutable nature of blockchain to ensure secure task offloading and data integrity. The experimental results highlight the framework's ability to prevent common security threats, such as data tampering and unauthorized access, while maintaining efficient offloading performance. However, the computational and storage overhead associated with blockchain integration could be a potential drawback. Future work might explore lightweight blockchain solutions or hybrid models to mitigate these overheads. Author in [29], introduces a multiagent DRL framework for IoT computation offloading in IoT environments. The multiagent approach allows for cooperative decision-making among vehicles, enhancing overall network efficiency. The authors demonstrate significant improvements in task offloading performance and network resource utilization through extensive simulations. However, the complexity of the multiagent DRL model and the need for substantial computational resources for training may pose practical challenges. Future research could investigate more scalable training methods and the potential of transfer learning to reduce training time and resource requirements. Author in [30], presents an energy and service-level agreement (SLA)-aware GA for edge-cloud combined computation offloading in IoT networks. The proposed algorithm balances energy consumption with SLA requirements, optimizing offloading decisions across both edge and cloud resources. The authors provide a thorough analysis and extensive simulations, demonstrating the algorithm's effectiveness in reducing energy consumption while meeting SLA constraints. However, the computational complexity of genetic algorithms and their convergence time might be concerns for real-time applications. Future work could focus on enhancing the algorithm's efficiency and exploring hybrid approaches that combine genetic algorithms with other optimization techniques.

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

3. System model and Problem formulation

The proposed model, Refining Privacy-Aware Task Offloading with HGSA in Multi-Tier Computing System, is designed for next-generation wireless networks, addressing privacy concerns and task offloading challenges in dynamic, multi-tier environments. It integrates edge, fog, and cloud computing layers to optimize task allocation, minimize latency, and ensure data security.

3.1 Hybrid Optimization Approach

The core of the model is the HGSA algorithm, combining the exploration capabilities of GA with the exploitation strengths of SA. This hybrid strategy balances local and global optimization, enabling effective resource utilization across multiple computing tiers. By incorporating privacy constraints as a key factor in the optimization process, the model ensures secure task allocation while maintaining system efficiency.

A. Privacy-Aware Multi-Tier Design

The model adopts a privacy-aware framework, embedding secure encryption protocols and data isolation mechanisms to protect sensitive information during task offloading. It dynamically adapts to the heterogeneous nature of multi-tier systems, accounting for factors like network bandwidth, device capabilities, and task priority. This ensures scalable and secure performance in real-time applications, making the model suitable for emerging wireless networks.

C. Adversary Model

We create the statement, which every IoT devices is consistent in this learning. Though its manners the policy updating procedure honestly, the cloud is seen as an inquisitive but honest entity which is attentive in concluding offloading preference from the accumulated experiences. It appears sense that the current MADRL-based solution shares the experiences of transferring them to a powerful unit, like the cloud, and keeping them in a replay buffer for policy updates. The attacker will regulate an arbitrary offloading preference of IoT $\hat{\mu}$ based on the data, which is currently reachable. In particular, the adversary indicates a sampled mini-batch of the target IoT devices g from the replay buffer by maximizing the log likelihood of g's actions to carry out a single gradient step, as realized below;

$$\Im^{g} = -\mathbb{W}\log\hat{\mu}^{g}\left(Y_{g}\middle|\mathbb{C}_{g}\right) + \mathfrak{H}\mathrm{En}(\hat{\mu}^{g}) \quad (1)$$

Here $\log\hat{\mu}^{g}\left(Y_{g,n}\middle|\mathbb{C}_{g,n}\right)$ signifies the log probability to choosing the real action of g-th IoT devices $Y_{g,n}$ below the observed action of the adversary $\mathbb{C}_{g,n}$. The En signifies the entropy of $\hat{\mu}^{g}$ distribution. Existing research has established an interpretation procedure based on inverse reinforcement learning that empowers attackers to improve the reward function by exploiting experience leaks as a resource of performing the above-mentioned attacks.

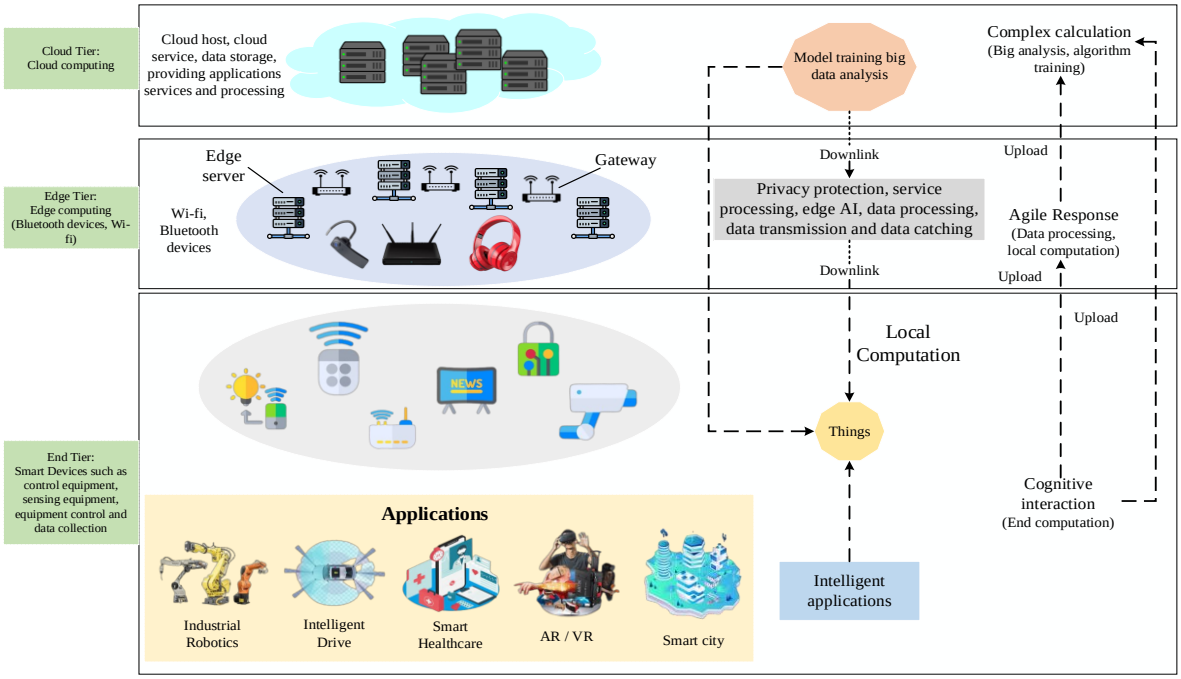

Figure 1. Overall Proposed Architecture

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

3.2 MTCM Framework

In this portion, we initially change the issues from a system cost minimization problem to a reward maximization problem by relating it as a multi-agent version of the Markov Decision Process (ϵ-MDP). We succeeding to drive into depth around the presented PA-MADRL method as shown in figure 2.

Multi Agent ϵ-MDP Formulation

A joint optimization problem may be renovated from a system cost reduction challenge to a reward maximization one by modifying it as a multi-agent variant of the generic Markov Decision Process (MDP). In our setup, the transition function $\mathfrak{M} = \mathbb{M}(X_{n+1}|X_n, Y_n)$ of generic MA-MDP $\langle \mathcal{T}, \mathfrak{B}, \mathfrak{M}, \mathfrak{S} \rangle$ is dynamic and vacillates beneath the same pair of (X_n, Y_n), $X_n \in \mathcal{T}$, $Y_n \in \mathfrak{B}$. This is because of the (δ, Φ)-differential privacy mechanisms that will affects action selection and transmission power. The changing transition function $\mathfrak{M}'$ is demarcated as $\mathfrak{M}' = \mathbb{M}(X_{n+1}|X_n, Y_n(\xi) = \{D^i_{g,h,n}(\xi), EF^i_{g,h,n}(\xi)\})$. The changing transition function $\mathfrak{M}'$ is confined by (δ, Φ)- differential privacy-based noise. We implement a multi-agent version of ϵ-Markov Decision Process (ϵ-MDP) to modify the issue from system cost reduction to reward maximization, which represented as a tuple $\langle \mathcal{T}, \mathfrak{B}, \mathfrak{M}', \mathfrak{S} \rangle$. The characterizations for every component in the tuple are offered below.

State

In task offloading through VANET, the state $X_n \epsilon \mathcal{T}$ is a united state of every IoT devices $X_{g,n}$ with $X_n = \{X_{g,n} | g \epsilon G\}$. The system encompasses of three parts such as the offloading target Edge, the transmission power in the sub-channel ($X^{power}_{g,n}$), the location of the g-th edge and the distance among the IoT devices and other IoT devices and Edge. The state is well-defined as specified below.

$$X_{g,n} = \{X^{off}_{g,n}, X^{power}_{g,n}, pos_{g,n}, pos_{g,g',n}, pos_{g,p,n}\} \quad (2)$$

Here $X^{off}_{g,n} \in [0,1,2 \dots, |L|]$, both $X^{off}_{g,n} = 0$ and $X^{power}_{g,n} = 0$ designate which the job can be processed locally. $pos_{g,n}$ signifies a two-dimensional coordinate, $pos_{g,g',n}$ signifies the distance between the g-th IoT devices and all other IoT devices and $pos_{g,p,n}$ signifies the distance between the g-th IoT devices and all Edge.

Action

Task offloading concluded VANET that encompasses every IoT devices which creating a supportive optimal $Y_n = \{Y_{g,n} | g \in G\}$. The behaviour of the g-th IoT devices is demarcated. The stages are as follows:

$$Y_{g,n} = D^i_{g,n}, EF^i_{g,n} \quad (3)$$

Here $D^i_{g,n} \in \{0,1\}$ and $EF^i_{g,n}$ signifies the continuous within $(0, EF_{max})$.

Transition Function

The transition function $\mathfrak{M}'$ is demarcated as $\mathbb{M}(X_{n+1}|X_n, Y_n(\xi)$ and is inadequate by the (δ, Φ)-differential privacy-related noise. $Y_n(\xi) = D^i_{g,h,n} + \mathfrak{X}(0, \xi^2 R), EF^i_{g,h,n} + \mathfrak{X}(0, \xi^2 R)$, here the kernel is set to $R(s,t) = v^{-\theta|s-t|}$ and ψ signify the Manhattan distance.

Reward function

The reward function $O_n \epsilon \mathfrak{S}$ is intended to meet the problem's optimization goal by turning the minimization problem to a maximization problem. The difference among two prizes should be less than one to deliver privacy protection. So, the reward function is demarcated as follows:

$$O_n = \beta(d^S_n + d^r_n) \quad (4)$$

Here,

$$d^S_n = \sum_{g \in G}(\beta(q^S_{g,n}) + \beta(y^S_{g,n})) \quad (5)$$

And,

$$d^r_n = \{\sum_3 g \in G\, (\beta(q^r_{g,n}) + \beta(y^r_{g,n})\} \quad (6)$$

If task is not failed, if task is failed. The $penalty = 3$ signifies the penalty value for task failure.

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

3.3 Overview of the PA-MADRL

We presented a privacy-aware multi-agent DRL (PA-MADRL) approach to report the problem of multi-agent ϵ-MDP that is an upgraded version of MADDPG with (δ, Φ)-differential privacy. The presented PA-MADRL model integrates functional noise formed by the (δ, Φ)-differential privacy mechanisms into policy selection and apprising during policy learning. Specifically, presenting noise to action selection decreases the observing accuracy on $u_{g,n}$, but adding noise to policy updating decreases the entropy regularized performance. Figure represents an overview of our PA-MADRL that encompasses of two key processes such as centralized learning and distributed offloading. In the centralized learning process, the involvements formed by interactions among every IoT devices and every Edge as well as the state X_n, action Y_n, reward O_n and new state X_{n+1} are uploaded to the cloud to learn cost-effective task offloading policy. Though a global image of the communications among every IoT devices and every Edge can alleviate the environment and later that improve convergence performance, it also intensifications the danger of preference leakage. The (δ, Φ)-differential privacy technique is employed to safeguard offloading preferences in the centralized learning process. Following centralized learning, every IoT devices downloads the job offloading strategy from the cloud and decides whether to offload reliant on its own state $X_{g,n}$.

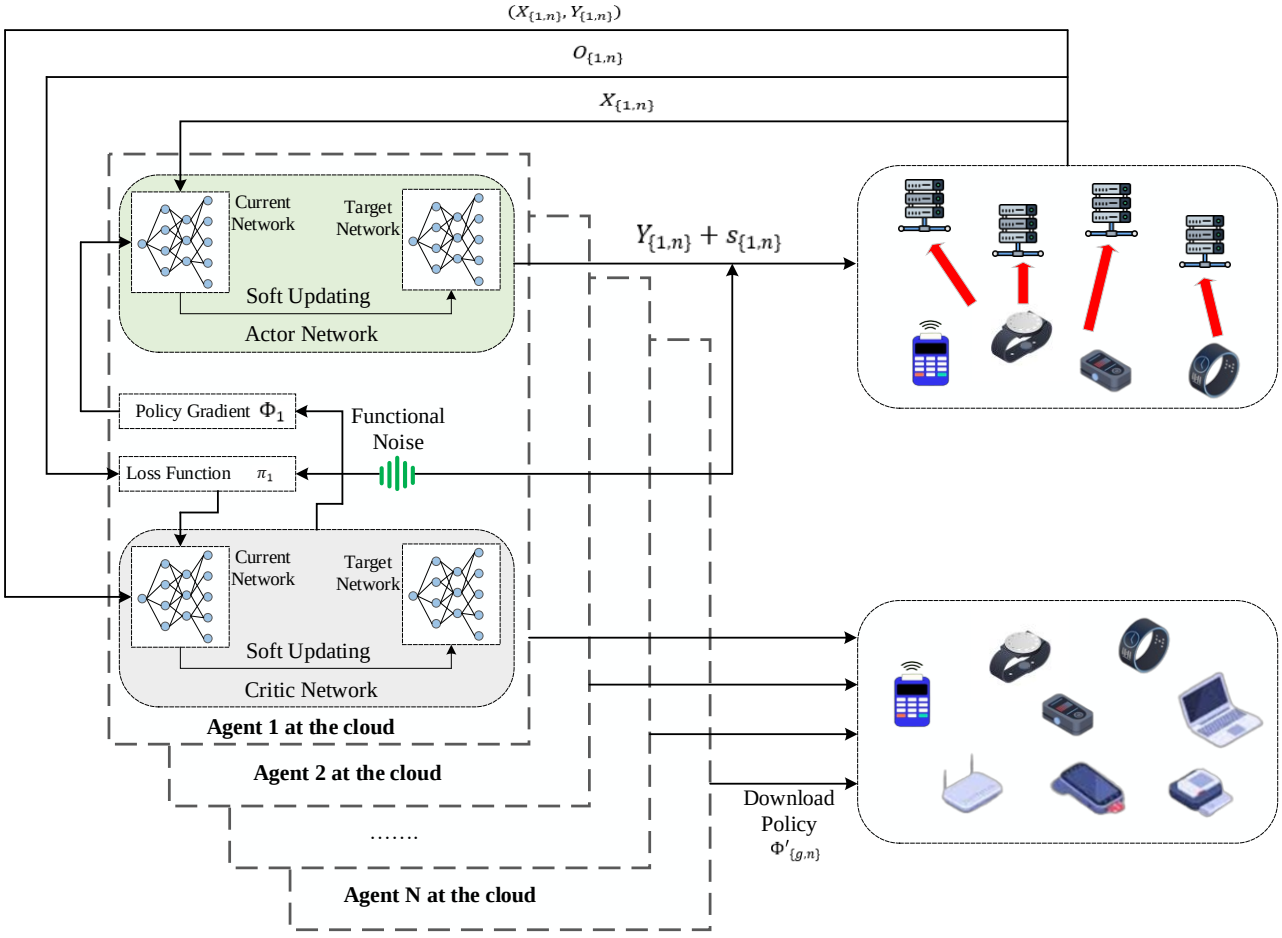

Figure 2. Illustration of PA-MADRL

Details of the PA-MADRL

Centralized Learning Process: The centralized learning process is portrayed in Algorithms 1 and 2. First algorithm represents the weights $\mu_g(.), \mu'_g(.), \pi_g(.), \pi'_g(.)$ is to adjust the actor and critic networks at the initial stage of the learning process. Parameters in the environment are also initialized. After that, for every θg ∈ G, the differential perturbation function $f_{g,n}(.,.)$ is prepared to zero. The offloading decision in every IoT devices is made based on the current actor network μ_g under the (δ, Φ)-differential privacy-based noise $f_{g,n}(X_{g,n}, f_g(X_{g,n}))$ that will avoid from leaking the precise offloading decision during the exploring and policy

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

updating stages. Following the completion of the offloading action, all IoT devices obtain a reward O_n and the task offloading system transfers to a new state X_n, Y_n, O_n and X_{n+1} is employed to form an experience that is later uploaded to the cloud for policy updates. When a minibatch's length is equal to or more than the size of the replay buffer, a minibatch will be randomly sampled in the cloud. Subsequently, Algorithm 2 apprises the differential perturbation function before updating the actor and critic networks as of right now. Following that, equation is implemented to regulate the goal value $\mho_g$.

$$\mho_g = O_g + \mathfrak{H}\pi'_g\left(X_{g,a+1}, Y'_{g,a+1}\right) \quad (7)$$

Here $Y'_{g,a+1} = \mu'_g(X_{g,a+1})$, $a \in Z$. By reducing the loss $\mathfrak{I}^{\pi g}$ that is provided as follows, the weight w_g of the current critic network $\pi_g(\cdot)$ is updated based on the goal value $\mho$.

$$\mathfrak{I}^{\pi g} = \frac{1}{|Z|}\sum_{a\in Z}(\pi_g(X_a, Y_a) + f_{g,n}(X_a, Y_a) - \mho_g)^2 \quad (8)$$

Equation recounts the weight $\aleph_g$ of the current actor network $\mu_g(\cdot)$ to the sampled policy gradient $\mathfrak{I}^{\mu g}$.

$$\mathfrak{I}^{\mu g} \approx \frac{1}{|Z|}\sum_{a\in Z}\left(\triangle_{\aleph_g}(\mu_{g,a}) + f(X_a, Y_a)\right) \triangle_{Y_a} \pi_g(X_a, Y_a) \quad (9)$$

Following a cycle of policy updates for every IoT devices, the weights of the target actor and critic networks $\aleph'_g$ and w'_g are gently modified using equations in that order:

$$\aleph'_g = (1 - e)\aleph'_g + e\aleph_g \quad (10)$$

And,

$$w'_g = (1 - e)w'_g + ew_g \quad (11)$$

Ultimately, each IoT devices transfers the target actor network μ'_g as part of the job offloading approach at the assumption of the learning process.

Algorithm 2: The differential perturbation function $f_{g,n}(\cdot, \cdot)$ reorganized frequency is measured at the outset of the procedure by a reset factor $\Re$. Reset aims to adjust the adversary's difficulty in interpreting policy. The following interest the reset factor such as an adversary will more eagerly gather more experience to gather policy when there are more training steps in a training episode. Even though the adversary will only attain minimal information per inference thanks to the (δ, Φ)-differential privacy mechanisms, if the noise's generation parameters stay constant for an extended period of time, the adversary may still be able to attain more information about the policy. For illustration, the accurate policy with bounded noise. Following the reset step, a Gaussian process $\Psi_{g,n}(X_{n+1}, Y) \simeq \mathfrak{X}(\varpi_{g,n}(X_{n+1}), \varrho_{g,n}(X_{n+1})$ through equation, $\theta_Y \in \mathfrak{B}$, would update the value of $f_{g,n+1}(X_{n+1}, Y)$. It must adjust the operation dimension, which the kernel function provisions as the state space's size cultivates. The RKHS kernel in this learning is calculated to $R(s,t) = v^{-\theta|s-t|}$, where the Manhattan distance is characterized by the symbol $|\psi|$. This strategy is based on the notion of state space.

$$\varpi_{g,n} = \frac{(t^{\tau\lambda} - t^{-\tau\lambda})f(X_g^-) + (t^{\tau H} - t^{-\tau H})f(X_g^+)}{t^{\tau f} - t^{-\tau f}} \quad (12)$$

And

$$\varrho_{g,n} = \frac{(t^{\tau\lambda} - t^{-\tau\lambda})t^{\tau\lambda} + (t^{\tau H} - t^{-\tau H})t^{\tau H}}{t^{\tau f} - t^{-\tau f}} \quad (13)$$

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

Here $= (4\eta(m+1)/|Z|)^{-1}$, $\lambda = \left|\left|X_{g,n+1} - X_{g,n}\right|\right|$, $f = ||X_{g,n+2} - X_{g,n}||2$, $H = ||X_{g,n+2} - X_{g,n+1}||2$ and η signifies the balance factor.

Distributed Offloading Process: Following the centralized learning process, every IoT devices downloads the target actor network $\mu'(\cdot)$ from the cloud in accordance with the task offloading policy, and its usages its own state $X_{g,n}$ to indicate which tasks to offload. Next, the transition function $\mathfrak{M} = \mathbb{M}(X_{n+1}|X_n, Y_n)$ transforms the current state of the system X_n into a new state X_{n+1}. It should be distinguished that the transition function is autonomous of the privacy level ξ because our distributed offloading process abolishes the risk of offloading preference leakage, and our PA-MADRL only suggests a proactive protection mechanism to safeguard offloading preference during the learning process. This makes sense because, in a distributed offloading process, no entity cloud, IoT devices, or Edge can continuously access a specific IoT's offloading decision due to the mobility of IoT. As a result, the adversary taken into consideration in this paper is unable to deduce the offloading preference.

4. Result and Analysis

We placed our TBBS-TO prototype into follow. Initially, we employed our block chain framework by utilizing the Hangzhou Blockchain Technology Research Institute's BROP stage. A block chain related stage known as BROP assists efficient and genuine relationship among many stakeholders. Moreover, BROP helps assorted consensus approaches and creating the perfect material for appraising POO's efficiency. 4 workers create our blockchain prototype and they are employed on a server called Intel i7-11700k, 8 cores, 1.7GHz, 32GB. We added the block ID and the time trample, which was produced, the number of transaction and the transaction information are explored in section 1V in every block. Every block surround 10,000 transactions unless otherwise specified. The consensus protocol can be utilized to concur on a block and include to the block chain in every 3 seconds. A CNN technique is implemented as the model in the RL approach. A max-pooling layer pursues the original convolutional layer. Convolutional layers contain out channels that are 25,481, 64, 256 and 16 in order. There are nine, two, seven, three, four kernel size strata. The most recent outputs prospect distributions for the action and fully connected. The hyper-parameter that contains the weight parameters for the efficacy function and learning rate is changed to imitate the difference among the workers. We also carried out into 5 distinct RL approaches such as DQN, AC, Asynchronous A3C, MADQN with proposed PA-MADRL.

Deep Q network (DQN): Every client has dual networks installed. $\text{Target}_{\text{net}}$ Remains the pathway of a copy of Eval_{net}, that is employed to evaluate the state action assessment. A collection of memory barriers provisions the final 200 behaviors. There is a batch size of 32. There are 5 iterations dividing $\text{Target}_{\text{net}}$ and Eval_{net}.

Actor-Critic Network (AC): An AC network is employed. The detractor outputs the current state value function by employing the TD approach. By implementing the detractor output as a source, reduces the benefits of the actions to offer the action allocation.

Asynchronous Advantage Actor-Critic (A3C): It enlarged the AC functionality by including the parallel processing. The learning agent appeals to four threads that work collectively to modernize a sharded approach, which combines the training data.

Multi-agent DQN (MADQN): Every four learning agents in this approach separately produce the policy.

Then, our formerly created mutual video Trans coding technology can utilize the offloading approach produced from the blockchain to constrain the Trans coding work allowance to additional test offloading concert. On this proposal, each connected device will create resources for video Trans coding. Mobile devices conclude that CU to the offload and create suggestions for video Trans coding. Then, video information will be passed to the intended CU. MD will accept the arrival pleasant leading end of the Trans coding method. This proposal includes 4 laptops such as Intel i7-7700k, 4 cores 4.25GHz/16GB, AMD Ryzen 5 Quad Core 3.2GHz/16GB, Intel i7-10750H, 6 cores 2.6GHz/16GB, Intel i7-10510U, 4 cores 1.8 Ghz/16GB performing as the MD and subscribers, and one server is performing as the cloud as Intel Xeon Platinum 8163, 8 cores 2.5Ghz/64GB.

Trans coding material is FFmpeg when the operating system is CentOS 7. Each node updates a resource monitor to identify changes in resources. The wired link has a 100Mbps bandwidth. The Trans coded video substance extent is signified by the random variable$U[2, 10]$s. The real video Trans coded bitrate set is 1080p (60fps), 720p (60fps), 1080p (30fps), 720p (30fps), and 480p (30fps). At every time slot, the subscriber offers a common amount of behavior with unreliable computational resource necessities and video sizes. The amount of tasks produced a Poisson distribution with $\lambda 1$ values. An identical distribution with $U[5, 20]$ is experimental for the amount of time slots which divided to dual consecutive tasks that produce while $U[a, b]$ signifies an identical distribution across the range$[a, b]$.

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

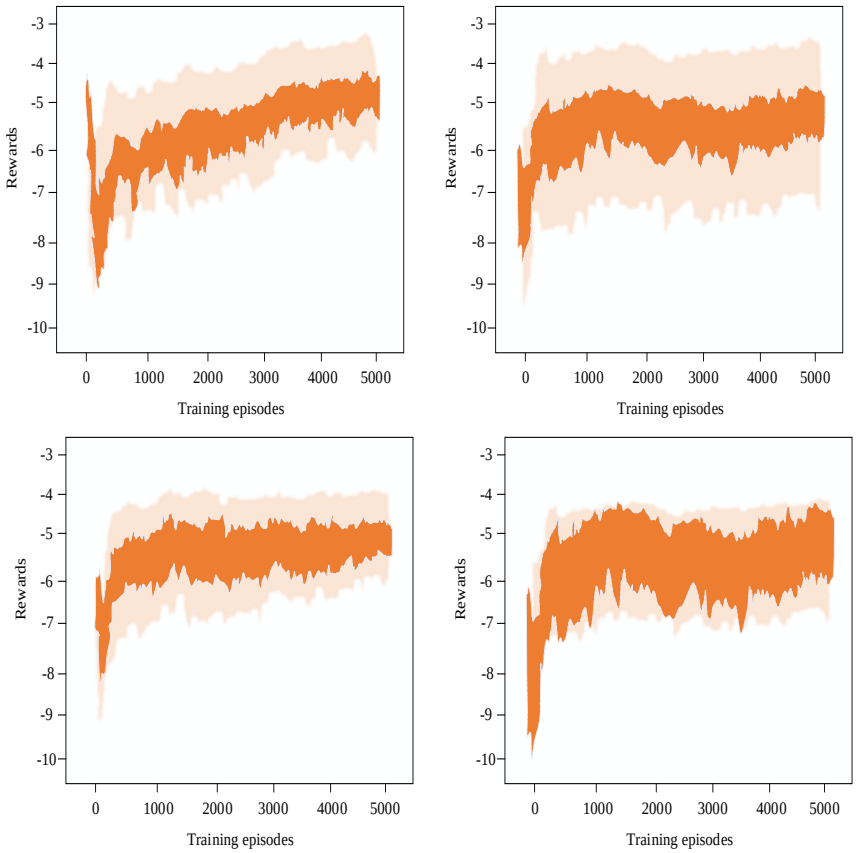

Figure 3. Performance Analysis of Training Episodes

Performance Analysis of RL Models

To validate the overview of our introduced task offloading mechanism, we initially determine the performance of five RL models including as DQN, AC, A3C, and MADQN, while solving this issue. The evaluated metrics involve loss variance via training and delay of unit processing during testing. Loss Variance: Figure 3 (a) illustrates the value of loss function while training. As per the figures, entire existing models experienced slow trend minimizing whereas rapid minimizing trend and then arrived the constant stage, depicting that the algorithm is congregated. This effect arises from the learning agent's initial lack of knowledge about the system dynamics, prompting it to estimate the values of actions in various states to improve policies. The curves stabilize once the agent's policy converges after learning the value function, or Q-value. The convergence of all curves around the 2500s indicates that their performance in work offloading issues is comparable. Notably, the AC-based approach initially shows an increasing trend before declining around the late 900s, while DQN and MADQN continue to increase as training progresses. Therefore, DQN and MADQN are far better than other models for job offloading when training time is limited. However, due to insufficient parameter tuning effectiveness of task offloading is minimized. In contrast, our proposed TBBS-TO adapts MA-DDQN with optimization algorithm which aids for enhancing task offloading performance.

Unit Delay: The latency of processing a task's unit of raw data is referred to as the UD. We use datasets of 600s to assess the UD of various solutions, and the results are displayed in Fig. 3(b). As shown in the image, A3C outperforms the other options in most cases due to its minimal delay. A3C's advantage lies in the AC model's ability to consistently seek the activity more effectively than the average. Additionally, A3C explores the environment using four distinct methods simultaneously, achieving greater performance. The performance of the other three solutions is comparable. The DQN curves start higher than the AC and MADQN curves but converge closely after 300s. Anyhow, the AC network may exhibit higher unit delay metrics in task offloading due to the complexity of simultaneously learning both policy and value functions.

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

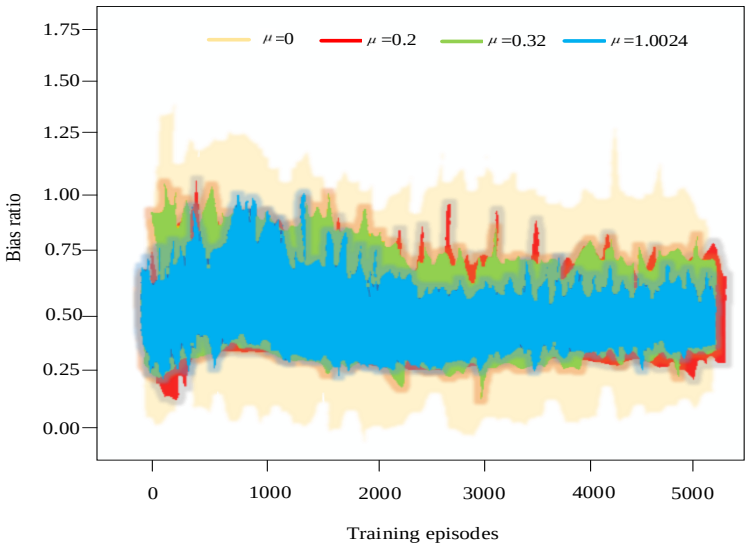

Figure 4. Analysis of Bias Ratio

Comparative Analysis

In this study, a comparative analysis is conducted between five distinct RL approaches for task offloading: DQN, AC, A3C, Multi-Agent DQN (MADQN), and the proposed PA-MADRL. These approaches are evaluated based on three key performance metrics: energy consumption, average system cost, and latency. Each method offers a unique approach to task offloading in multi-agent, multi-tier systems, with varying trade-offs between computational efficiency and performance.

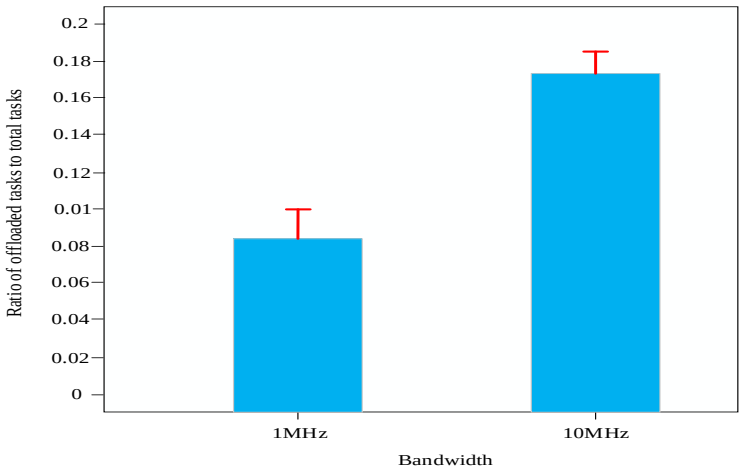

Figure 5. Bandwidth Analysis

(i) Energy Consumption Comparison

Energy consumption is a crucial factor in task offloading, especially for mobile devices operating in wireless networks where battery life is limited. DQN performs reasonably well in terms of energy consumption as it utilizes two neural networks (target and evaluation) for decision-making, but this dual-network structure increases computation overhead, thus slightly raising energy costs. Moreover, DQN relies on a memory replay mechanism and a batch size of 32, which could lead to higher energy consumption during the training phase due to continuous updates and evaluations of network states. On the other hand, AC is more efficient in terms of energy consumption because it uses a single network to evaluate both the value function and policy. However, it lacks the

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

parallelism and scalability seen in more advanced methods like A3C and MADQN, which can increase computational requirements in large-scale systems. A3C approach, which enhances AC by using multiple agents running in parallel, shows improved energy efficiency compared to DQN and AC. The parallel execution of threads allows A3C to share the computational load, reducing the time each agent spends on learning, and thus, reducing energy consumption. MADQN, where multiple agents independently generate their policies, consumes more energy compared to A3C due to the need for separate network training for each agent. However, PA-MADRL, the proposed approach, improves on MADQN by incorporating Double Q-learning into the multi-agent framework. This reduction in overestimation bias leads to more stable and efficient learning, ultimately lowering energy consumption by avoiding excessive exploratory actions that consume energy unnecessarily. The collective training approach used in PA-MADRL strikes an optimal balance between task offloading performance and energy efficiency. Figure 5 illustrates the bandwidth measures for task offloading.

(ii) Average System Cost Comparison

Average system cost typically refers to the total cost involved in completing a task, which includes the computation cost, communication cost, and the cost of decision-making under various constraints. In the case of DQN, the average system cost is relatively higher due to the intensive use of both target and evaluation networks, which require substantial computational resources and memory management. The batch processing mechanism in DQN, while effective for convergence increases the overall cost as it necessitates frequent updates and state evaluations, which can lead to slower decision-making times. In contrast, AC exhibits a more efficient approach to system cost by reducing the number of computations needed for policy evaluation, thanks to its single-network architecture. However, it still faces challenges in terms of scalability, as it is not as adaptable to dynamic environments as more parallelized systems like A3C. A3C stands out for its ability to scale better in multi-agent scenarios. The parallel processing threads allow A3C to distribute computational resources more effectively, thereby reducing the overall cost of system operation. Despite the increased number of agents, the cooperative approach to learning ensures that the system can handle larger and more complex environments without a proportional increase in cost. MADQN shows a notable improvement over A3C by enabling independent learning among agents while still using centralized decision-making, which strikes a balance between cooperation and independence. However, it faces higher costs due to the need for each agent to maintain its own policy. In contrast, PA-MADRL reduces system costs significantly by mitigating the overestimation bias typically seen in DQN-based algorithms. By improving the stability of learning and convergence, PA-MADRL leads to more accurate offloading decisions with fewer computational resources, thus lowering the overall system cost as shown in figure 6. This results in an efficient task offloading solution for large-scale, distributed networks.

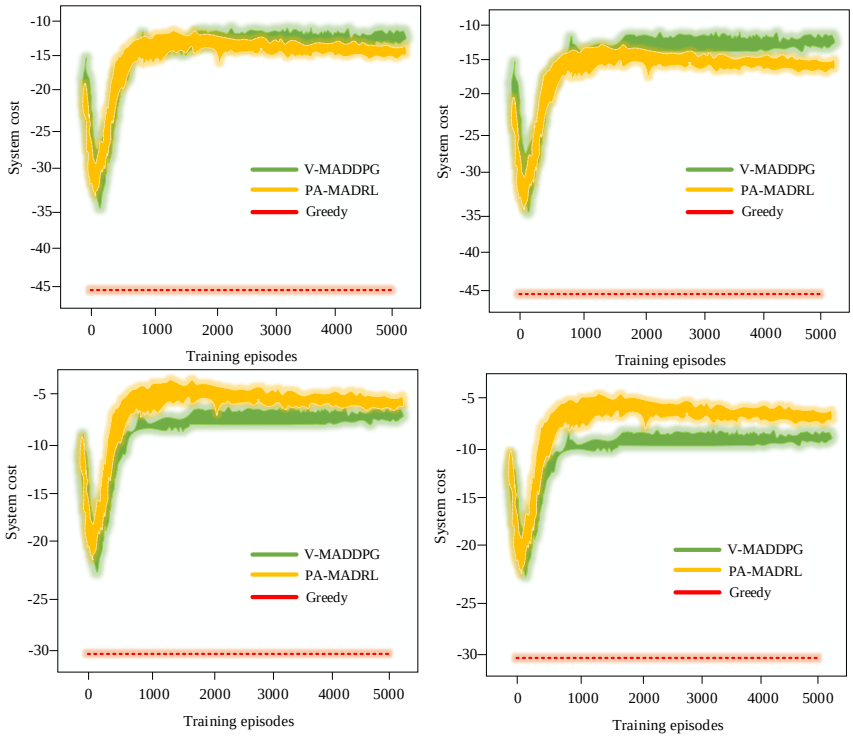

Figure 6. Comparison of System Cost

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

(iii) Latency Comparison

In terms of latency, which measures the time it takes for tasks to be offloaded and processed, DQN exhibits relatively high latency due to the process of updating the target and evaluation networks. The batch-based training process inherently introduces delays, as the system waits for a sufficient number of experiences to be accumulated in memory before performing updates. This can be particularly problematic in real-time applications where low latency is crucial. AC, with its single network setup, reduces the latency when compared to DQN because it does not require as much computational overhead for each decision. However, its performance is still limited by the sequential nature of the updates and the inability to handle large, distributed tasks efficiently. A3C significantly reduces latency by running multiple threads concurrently. Each agent can process different parts of the task-offloading problem simultaneously, which accelerates the decision-making process and reduces waiting times, particularly in large-scale networks with complex task distribution requirements. MADQN, while improving over A3C in terms of energy efficiency, introduces some latency due to the need for each agent to independently generate a policy. However, it benefits from quicker convergence compared to DQN and AC, resulting in relatively lower latency in decision-making. Finally, PA-MADRL offers the best latency performance by addressing the overestimation bias present in DQN-based systems. This leads to faster convergence, reducing the number of iterations needed to reach an optimal decision and minimizing the latency in task offloading. Overall, while DQN offers a strong baseline in terms of task offloading performance, it is outpaced by more advanced approaches like A3C, MADQN, and PA-MADRL in terms of energy consumption, system cost, and latency. PA-MADRL, the proposed method, combines the strengths of multi-agent learning with the stability of Double Q-learning, making it the most efficient and effective approach for task offloading in dynamic, multi-tier wireless networks. It outperforms the other models by balancing computational resources, reducing energy consumption, and minimizing system cost while ensuring low-latency performance.

5. Conclusion

The advent of multi-cloud computing represents a significant advancement in providing scalable and abundant computational resources for IoT devices. However, managing a multi-device, multi-cloud network presents numerous challenges due to the dynamic nature of real-time computing demands, fluctuating wireless channel conditions, and variable network scale. To address these challenges effectively, our research has proposed a novel approach through the development and implementation of a PA-MADRL framework. The PA-MADRL framework stands out by combining both continuous and discrete decision-making processes within a hybrid-offloading model. This model is designed to address the complexities inherent in multi-cloud environments by allowing IoT devices to make coordinated decisions regarding cloud server selection, offloading ratios, and local computation capacities. By integrating these decision-making aspects, the framework not only simplifies the management of dynamic network conditions but also optimizes the overall system performance. One of the key contributions of this research is the development of a probabilistic approach to handle discrete actions, such as the selection of cloud servers, which are traditionally challenging to incorporate into continuous decision models. This innovative approach enables the conversion of discrete actions into a continuous range, thereby improving the flexibility and adaptability of the decision-making process. This transformation is crucial for addressing the variability and dynamic nature of cloud resource management in real-time.

PA-MADRL framework further enhances system efficiency through its cooperative multi-agent structure. By employing a centralized training mechanism coupled with distributed execution, the framework effectively coordinates the actions of multiple IoT devices, optimizing the total system cost. This cost is primarily measured in terms of energy consumption by IoT devices and the rental fees of cloud servers. The centralized training allows for the learning of optimal policies, while distributed execution ensures that these policies are implemented efficiently across all devices. Experimental results from our study provide compelling evidence of the PA-MADRL framework's effectiveness. The results show that our framework not only successfully learns dynamic offloading policies tailored to the specific needs of each IoT device but also significantly outperforms existing solutions. Specifically, the PA-MADRL framework has demonstrated superior performance compared to four state-of-the-art deep reinforcement-learning agents and two heuristic algorithms. This performance improvement is evident in terms of reduced system costs, which encompasses both energy consumption and cloud server rental fees. The experimental results validate the effectiveness of the framework, highlighting its potential to deliver superior performance and cost efficiency compared to existing methods. Future research could explore further enhancements to the PA-MADRL framework, including its application to larger and more complex networks, as well as its integration with emerging technologies in cloud computing and IoT. Overall, the PA-MADRL framework stands as a promising contribution to the optimization of multi-cloud resource management, paving the way for more efficient and cost-effective solutions in the evolving landscape of IoT computing.

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

References

[1] P. Viktor and M. Fodor, "Examining Internet of Things (IoT) Devices: A Comprehensive Analysis," in *2024 IEEE 22nd World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Jan. 2024, pp. 000115-000120.

[2] J. Chen, J. He, F. Chen, Z. Lv, J. Tang, W. Li, and G. Han, "Towards General Industrial Intelligence: A Survey on IIoT-Enhanced Continual Large Models," *arXiv preprint arXiv:2409.01207*, 2024.

[3] F. Firouzi, B. Farahani, and A. Marinšek, "The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT)," *Information Systems*, vol. 107, p. 101840, 2022.

[4] P. Popovski, F. Chiariotti, K. Huang, A. E. Kalør, M. Kountouris, N. Pappas, and B. Soret, "A perspective on time toward wireless 6G," *Proceedings of the IEEE*, vol. 110, no. 8, pp. 1116-1146, 2022.

[5] K. Thakkar, A. Patel, S. Patel, K. Patel, A. Nayak, and A. Budhrani, "A Complete virtual Edge Computing Extrapolation Architectural style, Uses, and Implications," in *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*, Jul. 2023, pp. 63-70.

[6] K. Wang, J. Jin, Y. Yang, T. Zhang, A. Nallanathan, C. Tellambura, and B. Jabbari, "Task offloading with multi-tier computing resources in next generation wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 306-319, 2022.

[7] P. Zhang, N. Chen, G. Xu, N. Kumar, A. Barnawi, M. Guizani, and K. Yu, "Multi-target-aware dynamic resource scheduling for cloud-fog-edge multi-tier computing network," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[8] M. Alabadi, A. Habbal, and X. Wei, "Industrial internet of things: Requirements, architecture, challenges, and future research directions," *Expert Systems*, vol. 10, pp. 66374-66400, 2022.

[9] S. S. Gill, M. Xu, C. Ottaviani, P. Patros, R. Bahsoon, A. Shaghaghi, and S. Uhlig, "AI for next generation computing: Emerging trends and future directions," *Internet of Things*, vol. 19, p. 100514, 2022.

[10] M. Yazdi, "Integration of IoT and Edge Computing in Industrial Systems," in *Advances in Computational Mathematics for Industrial System Reliability and Maintainability*, Cham: Springer Nature Switzerland, 2024, pp. 121-137.

[11] G. Nain, K. K. Pattanaik, and G. K. Sharma, "Towards edge computing in intelligent manufacturing: Past, present and future," *Journal of Manufacturing Systems*, vol. 62, pp. 588-611, 2022.

[12] R. Das and M. M. Inuwa, "A review on fog computing: issues, characteristics, challenges, and potential applications," *Telematics and Informatics Reports*, vol. 10, p. 100049, 2023.

[13] M. Nikpour, P. B. Yousefi, H. Jafarzadeh, K. Danesh, R. Shomali, A. G. Lonbar, and M. Ahmadi, "Intelligent energy management with IoT framework in smart cities using intelligent analysis: An application of machine learning methods for complex networks and systems," *Networks*, 2023.

[14] A. Hazra, P. Rana, M. Adhikari, and T. Amgoth, "Fog computing for next-generation internet of things: fundamental, state-of-the-art and research challenges," *Computer Science Review*, vol. 48, p. 100549, 2023.

[15] D. Alsadie, "A Comprehensive Review of AI Techniques for Resource Management in Fog Computing: Trends, Challenges and Future Directions," *International Journal of Information Technology*, 2024.

[16] W. Almuseelem, "Energy-efficient and security-aware task offloading for multi-tier edge-cloud computing systems," *Soft Computing*, 2023.

[17] M. Hassan, A. A. Al-Awady, A. Ali, M. M. Iqbal, M. Akram, J. Khan, and A. A. AbuOdeh, "An efficient dynamic decision-based task optimization and scheduling approach for microservice-based cost management in mobile cloud computing applications," *Pervasive and Mobile Computing*, vol. 92, p. 101785, 2023.

[18] A. B. Kanbar and K. Faraj, "Region aware dynamic task scheduling and resource virtualization for load balancing in IoT–fog multi-cloud environment," *Future Generation Computer Systems*, vol. 137, pp. 70-86, 2022.

[19] N. Mungoli, "Scalable, Distributed AI Frameworks: Leveraging Cloud Computing for Enhanced Deep Learning Performance and Efficiency," *Sensors*, 2023.

[20] G. K. Walia, M. Kumar, and S. S. Gill, "AI-empowered fog/edge resource management for IoT applications: A comprehensive review, research challenges and future perspectives," *IEEE Communications Surveys & Tutorials*, 2023.

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

[21] H. Zhao, G. Lu, Y. Liu, Z. Chang, L. Wang, and T. Hämäläinen, "Safe DQN-based AOI-minimal task offloading for UAV-aided edge computing system," *IEEE Internet of Things Journal*, 2024.

[22] C. Hou and Q. Zhao, "Optimal task-offloading control for edge computing system with tasks offloaded and computed in sequence," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1378-1392, 2022.

[23] Mohanasundaram, "Graph Based Event Measurement for Analyzing Distributed Anomalies in Sensor Networks," *Sådhanå*, vol. 45, p. 212, 2020.

[24] H. Jiang, X. Dai, Z. Xiao, and A. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 4000-4015, 2022.

[25] S. Long, Y. Zhang, Q. Deng, T. Pei, J. Ouyang, and Z. Xia, "An efficient task offloading approach based on multi-objective evolutionary algorithm in cloud-edge collaborative environment," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 2, pp. 645-657, 2022.

[26] L. Zang, X. Zhang, and B. Guo, "Federated deep reinforcement learning for online task offloading and resource allocation in WPC-MEC networks," *Soft Computing*, vol. 10, pp. 9856-9867, 2022.

[27] J. Almutairi, M. Aldossary, H. A. Alharbi, B. A. Yosuf, and J. M. Elmirghani, "Delay-optimal task offloading for UAV-enabled edge-cloud computing systems," *IEEE Access*, vol. 10, pp. 51575-51586, 2022.

[28] C. Fang, X. Meng, Z. Hu, F. Xu, D. Zeng, M. Dong, and W. Ni, "AI-driven energy-efficient content task offloading in cloud-edge-end cooperation networks," *IEEE Open Journal of the Computer Society*, vol. 3, pp. 162-171, 2022.

[29] T. H. Nguyen and L. Park, "A survey on deep reinforcement learning-driven task offloading in aerial access networks," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, Oct. 2022, pp. 822-827.

[30] Y. Liu, Y. Mao, Z. Liu, F. Ye, and Y. Yang, "Joint task offloading and resource allocation in heterogeneous edge environments," *IEEE Transactions on Mobile Computing*, 2023.

DOI: https://doi.org/10.54216/JCIM.160204
Received: December 09, 2024 Revised: January 31, 2025 Accepted: March 01, 2025

Doi : https://doi.org/10.54216/IJNS.xxxxxx
Received: July 20, 2021 Accepted: Jan 04, 2022