Trustworthy Multi-Hop Cooperative Task Offloading in Device-Edge-Cloud Computing

Jianhua Liu, *Member, IEEE*, Xin Wang, *Senior Member, IEEE*, Shui Yu, *Fellow, IEEE*, Guangtao Xue, *Member, IEEE*, and Minglu Li, *Fellow, IEEE*

***Abstract*—Multi-hop cooperative task offloading (MCTO) allows resource-constrained edge clouds to collaborate and assist each other in completing computation-intensive tasks, such as training machine learning models through device-edge-cloud (DEC) computing. However, internal fake service attacks can pose a threat to the security and reliability of MCTO in DEC computing. In this paper, we propose a trust model based on a directed acyclic graph (DAG) and Proof-of-Work (PoW) to safeguard tasks against potential attacks. The edge node selection for task offloading involves two key steps: offloading confirmation and trust-based node selection. To mitigate the unreliability caused by internal fake service attacks during cooperative offloading, we propose a multi-hop offloading node selection algorithm based on the soft actor-critic (SAC) coalition. This algorithm helps identify trustworthy nodes for constructing secure offloading paths. Our experimental results demonstrate that the proposed algorithm effectively counters internal fake service attacks and significantly reduces cooperative offloading latency compared to existing leading approaches.**

***Index Terms*—Device-edge-cloud computing, task offloading, deep reinforcement learning, fake service attack.**

I. Introduction

DEVICE-EDGE-CLOUD (DEC) computing framework [1] helps IoT devices enhance their computation and storage capacities by offloading their computation tasks to edge or cloud nodes. Since a large amount of computational resources and data are required to handle the offloaded tasks in edge computing, IoT devices, edge nodes, and cloud nodes collaborate through resource and data sharing to complete these tasks. DEC computing can select collaborative nodes and elastically provision resources for offloaded tasks. Multi-hop cooperative task offloading (MCTO), as one of the running formats of DEC computing, enables DEC nodes to process offloaded tasks in a distributed manner [2], [3], [4]. For example, nodes in DEC can collaborate in training a machine learning model in swarm learning (SL) [5]. In this case, a user node can generate a machine learning task, which may be distributed to multiple edge nodes that work collaboratively as service nodes to train the model in SL. Compared with cloud computing and federated learning, SL uses a fully distributed model training framework without a dedicated server, reducing model training costs by leveraging distributed computing resources. It is deployed on open edge nodes, enabling model training with private data on these nodes through multi-hop computation offloading in the SL network based on DEC.

Despite the potential, MCTO not only needs to consider resource efficiency for DEC in processing distributed learning tasks but is also highly vulnerable to internal fake service attacks on DEC nodes in multi-hop paths [6]. It is crucial to guarantee reliability in cooperation for multi-hop task offloading and more desirable to prevent attacks in DEC computing due to the following reasons: Multi-hop task offloading requires a larger number of DEC nodes to help process tasks in a distributed manner, while the selection of multi-hop cooperative nodes provides more opportunities for malicious attackers to degrade task performance. Compared to traditional single-hop offloading cases, MCTO is more susceptible to selfish edge attacks due to increased node vulnerability during training. One such threat is the fake service attack, where malicious nodes falsely claim to offer computing services, ultimately degrading model training efficiency and accuracy. This also makes it difficult for existing task offloading schemes to meet the quality-of-service requirements for DEC computing.

Although some trusted schemes have been proposed to address attack problems in offloading computation tasks, they mainly spotlight on one-hop task offloading or external attacks. A blockchain-based access control scheme is designed in [7] to protect the cloud server from illegal offloading actions, while a multi-objective resource-aware model is proposed in [8] to eliminate malware attacks by making smart decisions on ad-hoc mobile edge cloud devices for task offloading. To ensure the safety of offloaded tasks on edge virtual machines, a semi-Markov decision process framework [9] is designed to minimize the risk of service rejection while meeting the requirements of latency-sensitive applications in mobile edge computing. There has been no study addressing internal fake service attacks through the effective selection of trusted nodes in MCTO.

Although blockchain-based security schemes are designed in [10], [11], [12] to protect the offloading of computation tasks,

Received 26 August 2024; revised 16 August 2025; accepted 24 September 2025. Date of publication 26 September 2025; date of current version 11 December 2025. This work was supported in part by the Basic Public Welfare Research Program of Shaoxing Science and Technology Program under Grant 2025A11008, and in part by NSFC Major International (Regional) Joint Research Project under Grant 62320106006. *(Corresponding authors: Jianhua Liu; Xin Wang.)*

Jianhua Liu is with the Department of Computer Science and Engineering, Shaoxing University, Shaoxing 312000, China, and also with the Network and Mobile Computing Lab, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: ljh_541@163.com).

Xin Wang is with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, NY 11790 USA (e-mail: x.wang@stonybrook.edu).

Shui Yu is with the School of Computer Science, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: shui.yu@uts.edu.au).

Guangtao Xue is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: xuegt@cs.sjtu.edu.cn).

Minglu Li is with the School of Computer Science and Technology, Zhejiang Normal University, Jinhua 321004, China (e-mail: mlli@zjnu.edu.cn).

Digital Object Identifier 10.1109/TSC.2025.3615156

1939-1374 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

the focus is on access control, computation performance, and privacy preservation to secure computation offloading. They did not consider the case where an MCTO lures an internal fake service attack in DEC. A blockchain-based trustworthy access control and computation offloading mechanism is devised in [10] to alleviate computation burdens while ensuring high security by taking into account the offloading decision and the usage of smart contracts in mobile edge cloud. In [11], a blockchain-based secure computation offloading scheduling scheme is proposed to protect content offloading from malicious service provider attacks rather than considering fake service attacks for MCTO. A blockchain-empowered collaborative task offloading mechanism is presented in [12] to encourage cloud-edge-device nodes to honestly offload computation tasks by considering participants' computational capabilities and network situations.

Previous works primarily consider an external attacker who can only launch malicious attacks on single-hop task offloading in edge computing. However, in the multi-hop task offloading DEC scenario, a mixed attacker also intelligently launches internal fake service attacks on multi-hop offloading paths. Existing trust schemes in the literature are less effective at ensuring MCTO's security in DEC, as the attacks have distributed characteristics and are difficult to capture and process using these existing methods. In addition, to fully utilize the processing capacity and ensure trust model training on edge nodes, the input-output dependency of the model in MCTO should be represented as a directed acyclic task graph (DAG) [13]. These concerns motivate us to develop a DAG model and a soft-actor-critic coalition game to select trusted offloading nodes for offloading paths to defend against internal attacks. Although several previous studies use learning-based approaches, they may not be able to achieve low-cost distributed multi-hop offloading to withstand internal attacks.

To prevent internal fake service attacks in MCTO, the task of offloading countermeasures among distributed DEC nodes is modeled as offloading confirmation processes between requesters and responders based on the DAG and Proof-of-Work (PoW). We propose a multi-hop offloading node selection algorithm based on the soft actor-critic (SAC) coalition to identify trustworthy nodes for constructing secure offloading paths. The main contributions of this paper are summarized as follows:

- Designing a DAG model for DEC nodes to perform multi-hop cooperative offloading tasks and robustly resist internal attacks from malicious service providers.
- Developing a trusted offloading scheme based on PoW to defend against fake service attacks. Specifically, to improve computational performance by minimizing latency while ensuring trust in model training and sharing among edge nodes, we formulate the offloading node selection as a CMDP-based coalition game.
- To reduce the complexity of offloading decisions, we present a multi-hop cooperative node selection algorithm based on the soft-actor-critic coalition game to obtain a reliable set of nodes for offloading paths.

Given the dynamic changes in the multi-hop cooperative task offloading network environment that may be affected by fake service attacks, nodes exhibit strong interdependence across time slots, and the associated calculations involve numerous parameters with high complexity. Existing heuristic algorithms, such as decoupling methods and relaxation approximation methods, exhibit weak adaptability in dynamic multi-hop cooperative task offloading environments, resulting in low efficiency [14]. Compared with traditional heuristic methods and deep reinforcement learning approaches, SAC coalition performs better regarding policy generalization ability, environmental adaptability, and stability. By introducing a maximum entropy regularization mechanism, SAC coalition enhances policy diversity, improves global search capabilities, and thus more effectively addresses the optimization problem of multi-hop cooperative task offloading [15]. It can also adaptively adjust its node selection strategy based on proof-of-work and system performance parameters while dynamically adapting to network environments affected by fake service attacks.

The remainder of the paper is organized as follows. Section II discusses related work. In Section III, we present the system model and formulate the trusted node selection problem for MCTO. Section IV proposes a soft-actor-critic coalition game to optimize the node selection policy for defending against fake service attacks. Simulation results are presented in Section V for illustration purposes. Finally, conclusions are drawn in Section VI.

II. Related work

In recent years, a series of approaches have been proposed to enhance the dependability of DEC systems from the perspectives of intelligent resource management [16] and adaptive service [17]. These studies focus on designing learning-based task offloading algorithms and scheduling mechanisms to address resource heterogeneity and network dynamics in DEC. For example, an AI-based cloud-edge-device collaboration framework [16] is designed to adapt to different scenarios, facilitating task offloading decisions under incomplete information. An integrated cloud-edge-device framework [17] is investigated for adaptive deep learning services to achieve lower latency and higher system throughput.

The heterogeneity and resource limitations of DEC nodes, along with the dynamics of collaboration settings, make it challenging to ensure the security of MCTO. To ensure reliable cooperation in DEC, it is necessary not only to consider the adaptability of resource utilization and service provisioning to improve collaborative performance but also to meet reliability and trust requirements. In [18], a cloud-edge-device collaborative reliable digital twin scheme is presented to minimize average communication time costs by optimizing device scheduling and computational resource allocation. In [19], secure device-edge-cloud collaboration is employed to achieve efficient and reliable content delivery, thereby enhancing security performance. Although the above studies exhibit adaptability and reliability, they did not consider multi-hop cooperative offloading security in the presence of internal attacks in DEC computing.

Although some efforts have been made to investigate the multi-hop task offloading problem in collaborative edge computing, existing studies mainly focus on optimizing the QoS for edge users in multi-hop task offloading scenarios [20], [21], [22], rather than addressing internal attacks. For instance, a joint multi-task partial computation offloading and flow scheduling scheme is proposed in [20] to minimize the completion time of all tasks. Because edge cloud infrastructures are unavailable in remote areas, a multi-path and multi-hop task offloading method is introduced in [21] to provide effective edge computing services for users in mobile ad hoc networks. To support resource-intensive computation, a multi-hop computation offloading model is studied in [22] to realize QoS-aware computation offloading based on a game theory approach. To provide edge services in areas with poor server coverage through

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISON OF RELATED WORKS

References	Attack types	Architectures	MCTO	Methodology	Trustworthy model training	Trustworthy scalability
[6]	Fake service attacks	Edge	×	DRL	×	×
[7]	Illegal offloading	VANETs	×	DRL, blockchain	×	×
[8]	Malware attacks	Edge-cloud	×	Multi-objective optimization	×	×
[9]	Service rejection	Edge-cloud	×	MDP	×	✓
[10]	Access control	IoT	×	DRL, blockchain	×	×
[11]	Eavesdrop privacy	Vehicular cloud	×	DRL, blockchain	×	×
[12]	Dishonest offloading	DEC	×	DRL, blockchain	×	✓
[13]	×	Edge	×	DRL	×	×
[16]	×	DEC	✓	Deep actor-critic	×	✓
[17]	×	DEC	✓	DRL	×	×
[18]	×	DEC	✓	Deep actor-critic, DT	×	✓
[19]	Untrusted delivery	DEC	✓	Social trust	×	✓
[20]	×	Edge	✓	Heuristic	×	×
[21]	×	Edge	✓	Heuristic	×	×
[22]	×	DEC	✓	Strategic game	×	×
[23]	×	IoT	✓	Hierarchical minority game	×	×
[24]	×	Edge	✓	DRL	×	×
[25]	×	IoT networks	✓	DDPG	×	×
[26]	×	UAV swarm	✓	Q-learning	×	×
[27]	×	Edge	✓	Online learning	×	×
Our work	Fake service attacks	DEC	✓	SAC coalition game, PoW	✓	✓

multi-hop task forwarding, a novel two-stage method based on a hierarchical minority game (HMG) and a tree-based routing mechanism is proposed in [23] to estimate the offloading costs and schedule the transmission for the offloading nodes. To enable more efficient computing resource utilization, a general multi-hop task offloading framework for vehicle-assisted collaborative edge computing is presented in [24], where deep reinforcement learning is used to address the curse of dimensionality problem caused by vehicular mobility and channel variability. To improve the efficiency of offloading transmission and minimize the total energy consumption, a deep deterministic policy gradient (DDPG) algorithm is proposed in [25] to optimize UAV trajectory and offloading ratio in UAV-assisted maritime IoT networks. A Q-learning framework is proposed in [26] to optimize the candidate UAV selection to achieve intelligent routing in a cognitive UAV swarm for emergency communications. To ensure the timeliness of delay-sensitive tasks, a two-stage optimization method is proposed in [27] for allocating resources between edge servers and offloaded tasks. It also utilizes a dual prediction model based on online learning to reduce energy consumption during the edge server selection process. These methods, however, ignored untrusted computing resources of offloading nodes caused by fake service attacks.

Compared with the above studies, in this paper, we consider a trusted multi-hop task offloading scheme to eliminate internal fake service attacks, thereby improving the QoS of collaborative task offloading and model training in DEC computing. More specifically, we utilize reinforcement learning to protect against fake service attacks in the multi-hop task offloading process, considering the latency constraints of task offloading and training time on DEC nodes. We also explore using PoW to enhance the performance of multi-hop collaborative offloading while meeting the reliability and trust requirements in DEC collaborations. The main attributes and comparisons of these solutions are summarized in Table I.

III. System Model and Problem Formulation

A. Attack Model

Cooperative offloading can be applied to enable a resource-constrained edge node to offload tasks, such as the training of a machine learning model, to neighboring edge nodes to improve learning accuracy in swarm learning based on DEC.

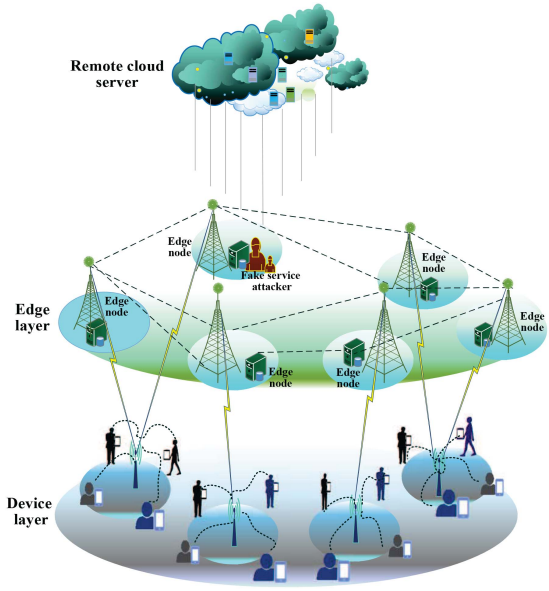


Fig. 1. An attack scenario for multi-hop task offloading in swarm learning.

Because multi-hop computation offloading often requires exploiting the collaborative computing capabilities of edge nodes, a malicious edge node may provide low-quality services through uncooperative strategic behaviors. An attack scenario is depicted in Fig. 1. In this scenario, we consider the case where a fake service attacker uses reduced computation resources to execute the offloaded tasks or outputs fake (training) results, thereby reducing the efficiency of offloading. Therefore, it is essential to identify trusted edge nodes in the cooperative offloading process to resist both selfish and malicious edge attackers. We summarize the main notations used in this paper in Table II. Fig. 2 shows the overall architecture and an application scenario of the proposed trustworthy multi-hop cooperative task offloading framework in the context of the swarm learning network based on DEC.

The operations can be divided into seven steps: ①A user advertises the model training task to its associated edge node. ②After receiving the request, the edge node executes a sub-task to train the model using local data and ③offloads the model to the next edge node until the defined conditions for training time and accuracy are met. ④The next hop node confirms the model

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

TABLE II
LIST OF MAIN NOTATION USED IN THIS PAPER

Notation	Explanation
$\mathcal{E}$,Σ,τ_i	The set of edge nodes, the task, and the subtask
G,e_i	Directed acyclic graph (DAG), edge node
h_{max}	Maximum hop for task offloading
ℓ_{pw}^{t}, ℓ_{pw}^{ac}	Probability of falling into the time and accuracy interval
Φ_k	A compliance tolerance for PoW
O^i	Set of offloading request nodes
o_j	Request node j to offload tasks
$\upsilon_{ji,k}$	Transmission rate
t_k	Transmission time
t_k^{sr}	Sending and receiving times
$x_{ji,k}$	Indication of confirming offloading
k	Index of the offloading transmission
η_k	Average arrival rate of offloading tasks
t_k^q, t_k^e	Time of waiting in queue and local training
$\mathcal{N}$, N	Set of players, number of player agents
v, $s_{e,n}$	Coalition characteristic function, the current node state
$s_{e,n-1}$	Previous hop node state
$\mathcal{C}_{e,n}$	Coalition state in state $s_{e,n}$
$\mathcal{C}_{e,n-1}$	Coalition state in state $s_{e,n-1}$
$\mathbb{C}_{e,n}$	Offloading costs
$a_{e,n}$, $a_{e,n-1}$	Actions in states $s_{e,n}$ and $s_{e,n-1}$
$r_{e,n}$	Reward obtained in state $s_{e,n}$
$\hat{r}_{e,n}$, $\hat{r}_{e,n-1}$	Reward allocated in states $s_{e,n}$ and $s_{e,n-1}$
ϕ_{th}, $\hat{r}_{th}$	Threshold of tolerance probability and allocated rewards
γ, α, λ, β	Discount factor, temperature parameter, and learning rate
$\mathcal{H}(\cdot)$, $\mathbb{E}(\cdot)$	The entropy and expectation
π, ψ	Offloading policy and policy parameters
θ, $\bar{\theta}$	Parameters of soft Q-function and target soft Q-function
$\hat{\nabla} J_Q(\theta)$, $\hat{\nabla} J_\pi(\psi)$	Soft Q-function gradient, policy gradient
κ	Target critic networks' soft update coefficient
$S_{e,n-1}$	$(s_{e,n-1}, \mathcal{C}_{e,n-1})$
S, A	Space of offloading states and actions

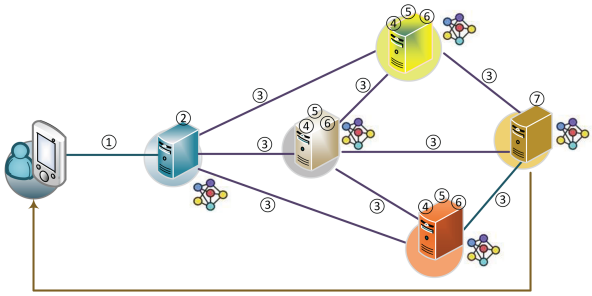


Fig. 2. Overview of the proposed MCTO in the SL network based on DEC.

weights on the training trajectory based on PoW, ⑤selects the previous hop node as a trusted training node, and ⑥merges its model weights with ones from the previous hop node's model to create an updated model before starting new training with local data. ⑦The last node confirms the model weights, selects the previous hop node as a trusted training node, aggregates the model weights to generate a global model, and returns it to the user.

B. System Cost Model for Trusted Cooperation

Due to the limited computational capacity and the prevalence of fake service attacks, it is inefficient for an individual edge node to undertake the offloaded task completely when aiming to improve computational performance in DEC. We consider an offloading system assisted by a set $\mathcal{E} = \{e_1, e_2, ..., e_l\}$ of densely deployed edge nodes, where each node is attached to an access point and within transmission distance of some other edge nodes. An edge node can send requests for task offloading or receive tasks from others. In the example of collaborative training of machine learning models without task segmentation and modularity, we consider a task Σ consisting of K subtasks, i.e., $\Sigma = \{\tau_1, \tau_2, ..., \tau_K\}$. A subtask in execution refers to a

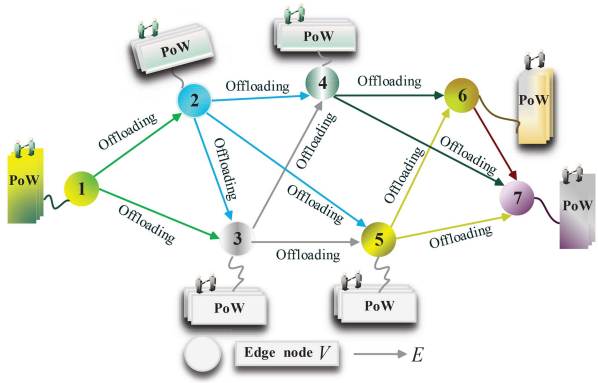


Fig. 3. A directed acyclic graph (DAG) based on PoW.

complete model training process on an edge node using its local data. For the edge node e_i in $\mathcal{E}$, its subtask $\tau_i \subseteq \Sigma$ is described by a tuple $\{d_i, w_i, \varpi_i\}$, where d_i is the size of its local data, w_i denotes the local model output from the subtask τ_i, and ϖ_i indicates the computational resources required to train the model w_i. The task Σ is sequentially executed by every edge node in $\mathcal{E}$ in order to train a large model with rich features until it reaches the converging state. The model w_i output from the subtask τ_i is distributed to the successor of the edge node e_i to continue training the model w_i with its local data and output the model w_{i+1} for the subtask τ_{i+1} until the task Σ finishes. When an edge node receives a request to offload and continue training a model trained by the previous hop node, it evaluates the performance of this model using the PoW mechanism. Multi-hop cooperative task offloading (MCTO) with N edge nodes is represented by a directed acyclic graph (DAG) $G = (V, E)$ as shown in Fig. 3.

Each vertex in G represents an edge node and is described by trust parameters, i.e., training time, queue waiting time, and model training results. Each edge in G represents a transmission between two nodes and is associated with the rate and time requested for task offloading. When an edge node acting as the requester sends offloading task requests to available neighboring nodes, the responding neighbor node confirms whether the offloading is credible according to the PoW. If the responding node finds that the offloaded task fails to meet the trust parameters specified in the PoW, the offloading confirmation is not passed; otherwise, the offloading is confirmed, and the responding node sends rewards to the requesting node for the current offloading task while selecting it as a trusted offloading node.

1) PoW for MCTO: The edge node in G sends an offloading task request as $\Theta_j = (D, Y_j)$ to its neighbor nodes, where D is the model size and Y_j is the resource consumed to complete model training. When the training task request node sets the maximum hop to $h_{\max}$, there should be at least $h_{\max} + 1$ nodes participating in model training in MCTO, and $h_{\max}$ is a configurable constant that depends on the SL network size. However, due to the destructive behavior of attackers (e.g., increased computation time delay, deliberate modification of the model training schedule, etc.), the offloading transmission, computation behavior, and training results become untrustworthy in MCTO, which can cause offloading failure and thus reduce the trust degree between edge nodes.

The trust degree of edge nodes and their decays can significantly shorten the lifespan of MCTO. To prolong the lifetime of MCTO as much as possible, a PoW mechanism is

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

introduced to ensure the trustworthiness in cooperative offloading and model training between edge nodes. The PoW utilizes the training period to prevent the abuse of decentralized computing capacity in MCTO. Thus, it is defined as $pw = \{\ell_{pw}^{t}, \ell_{pw}^{ac} | t \in [t_{\min}, t_{\max}], ac \in [ac_{\min}, ac_{\max}]\}$, where the random variables t and ac follow the standard normal distribution. ℓ_{pw}^{t} is the probability that the model training time t falls into the confidence interval of $[t_{\min}, t_{\max}]$ and $\ell_{pw}^{t} = \frac{1}{\sqrt{2\pi}} \int_{t_{\min}}^{t_{\max}} e^{-t^2/2} dt$. ℓ_{pw}^{ac} is the probability that the model training accuracy ac falls into the confidence interval of $[ac_{\min}, ac_{\max}]$ and $\ell_{pw}^{ac} = \frac{1}{\sqrt{2\pi}} \int_{ac_{\min}}^{ac_{\max}} e^{-ac^2/2} dac$. The higher the values of ℓ_{pw}^{t} and ℓ_{pw}^{ac}, the higher the PoW compliance of task offloading. In Fig. 3, a compliance tolerance for PoW in the offloading transmission k from a requester k is denoted as

$$\Phi_k = \nu \ell_{pw}^{t} + (1 - \nu) \ell_{pw}^{ac}, \tag{1}$$

where $0 \leq \nu \leq 1$ is the normalized weight to balance the training time and training accuracy. When receiving task offloading requests for model training, the edge node determines the model training time and accuracy defined by PoW. If the model training time and accuracy do not fall within the confidence interval, the edge node refuses to accept the task offloading. The set of affirmative request nodes for task offloading is defined as $O^i = \{o_j\}$. In the DAG G, each edge node needs to confirm the offloading between itself and the previous hop node. In other words, a user only needs to send a request to the first edge node, which will forward the request to the next edge node with the model w_i output from its completed subtask. Each receiving node determines whether to accept based on the time requirements and PoW and responds to the requester. Task offloading must be processed and confirmed by the edge nodes sequentially based on the time requirements and PoW. If a node does not have enough computational capacity, it cannot accept PoW. Constrained by the computational capacity of edge nodes, the offloaded task from one edge node to multi-hop edge nodes should be completed before the capacity of edge nodes and the links between edge nodes become unavailable.

If a task cannot be completed by an edge in the set $\mathcal{E}$, it sends a request to other edge nodes. If this node is the last one in $\mathcal{E}$, it can reconfigure the offloading path by adding new nodes into $\mathcal{E}$. Due to fake service attacks on edge nodes, this will cause the offloading latency to increase between the request node and the response node in the DAG G. The offloading latency varies across different paths in DAG G and mainly consists of three parts: time for a transmission between edge nodes, wait time in the task processing queue, and training time at edge nodes.

2) Offloading Transmission Time: In the DAG G, the response node confirms the offloading and receives offloading tasks. Due to offloading congestion caused by large payloads from request nodes, the available offloading transmission rate for each request node differs in the offloading transmission k between request node j and the response node i. This rate can be expressed as

$$\upsilon_{ji,k} = \zeta_{j,k} B_{j,k} \log_2 \left(1 + \frac{p_{j,k} g_{j,k}}{\sigma^2}\right). \tag{2}$$

At each transmission k, each request node j will be assigned to a response node i and obtain bandwidth $B_{j,k}$ from it. Time division multiple access (TDMA) is adopted for spectrum reuse among these request nodes, where $\zeta_{j,k}$ is the fraction of timeslots allocated to request node j. $p_{j,k}$, $g_{j,k}$, and σ^2 respectively denote the transmission power, channel gain, and background noise on the channel. In this case, the transmission time among edge nodes can be calculated as

$$t_k = \max_{j \in O^i} \left\{ \frac{x_{ji,k} D}{\upsilon_{ji,k}} \right\}, \tag{3}$$

where $j \in O^i$ indicates that response node i can choose request node j to confirm the offloading and offloading tasks. Specifically, if $x_{ji,k} = 1$ for an offloading transmission k from request node $j \in O^i$ in the previous hop, the response node returns an acknowledgment and receives the offloaded tasks; otherwise, $x_{ji,k} = 0$. For the model size D of the offloaded task, the required time to send and receive over the offloading transmission k is calculated as $t_k^{sr} = t_k$. Therefore, the total transmission time in a DAG path is $T = \sum_{k=1}^{h_{\max}} t_k^{sr}$.

3) Waiting Time in Offloaded Task Queue: In the DAG G, the offloading latency between the request node and the response node not only hinges on the offloading transmission time but also depends on the tasks arriving at the current node. The z_k tasks arriving at the current node during duration χ follow the Poisson distribution with parameter δ_k [28], that is, $P_j(\Gamma_k = z_k) = \frac{(\delta_k \chi)^{z_k}}{(z_k)!} e^{-\chi \delta_k}$, where Γ_k is the number of tasks arriving at the current node during duration χ. Because the service rate is constant, the offloaded task can be modeled as an M/D/1 queue. Thus, considering the number of tasks distributed in current transmission k, the average arrival rate of offloaded tasks is expressed as

$$\eta_k = \frac{1}{M} \sum_{j=1}^{M} P_j(\Gamma_k = z_k) \cdot \vartheta_k \cdot I_{\{x_{ji,k}=1\}}. \tag{4}$$

Where M represents the number of request nodes in the offloading task, ϑ_k is the computational resources required for finishing the training task, and $I_{\{*\}}$ is the indicator function, which equals 1 if the response node returns an acknowledgment and receives offloaded tasks, and 0 otherwise. The arrival of offloaded tasks needs to wait until previous tasks are processed by the response node due to the requirement of large CPU cycles to process the training task with the local data in the response node. Therefore, considering the resource requirements to complete training tasks in the offloading task queue, the waiting time in the offloading task queue is expressed as $t_k^{q} = \frac{\eta_k}{2 f_c (f_c - \eta_k)}$, where f_c is the service rate of each CPU core.

4) Local Training Time for Offloaded Tasks: Each response node performs distributed stochastic gradient descent (SGD) to train the currently offloaded model with its local data. If edge nodes perform local training up to ε_i iterations, then the local training time is $t_k^{e} = \frac{\varepsilon_i D}{f_c}$. Therefore, the total latency of offloading in transmission k is denoted as

$$T_o = \sum_{k=1}^{h_{\max}} \left(t_k^{sr} + t_k^{q} + t_k^{e}\right). \tag{5}$$

C. Offloading Node Selection in MCTO

As seen from (5), the total latency of the offloading in transmission k increases as the training time increases. A fake service attacker will endeavor to prolong the training time to consume computing power, thereby degrading the performance of the training models and ultimately diminishing the reward of the next hop edge node as it is unable to complete the training task on time in the DAG G. To enhance the service performance for training models, request nodes within the DAG can transfer the

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

training models to other response nodes if their own DAG nodes belong to a group that complies with PoW. This group of DAG nodes is commonly referred to as a coalition, and these DAG nodes are known as collaborative offloading nodes. The key to MCTO is to determine: (1) optimal cooperative offloading node selection policies and (2) methods for finding stable coalitions of DAG nodes to form a trust model training path. On the other hand, for DAG nodes, the selection of offloading nodes and the rewards for offloading work to train models in transmission k depend on the selection of offloading nodes and the rewards for offloading work in transmission $k-1$. Single-hop offloading methods and cooperative offloading models that do not consider these dependencies and associated rewards are not suitable for ensuring trusted task offloading and protecting against fake service attackers.

In addition, although reinforcement learning can maximize the accumulated reward of agents by taking joint actions and considering the dependency relationship and related rewards among agents, it cannot guarantee that the original action value Q-function will learn a policy that motivates agents to cooperate. To improve the overall reward in policy learning, we formulate and analyze a game between DAG nodes by combining cooperative coalition games with the constrained Markov decision process (CMDP). Therefore, the problem of offloading node selection in MCTO can be transformed into a coalition game based on CMDP to obtain optimal offloading node selection policies and offloading paths by considering the constraint of PoW and the latency cost in the offloading task. The CMDP-based coalition game can be represented by a seven-tuple $\Theta_M = \langle \mathcal{N}, S, A, \Pr, C_{pw}, R, v \rangle$, where $\mathcal{N}$ is the set of players in the game, S is the state space, A denotes the action space, $\Pr$ represents the state transition probability, C_{pw} is the constraint on proof of work, $R : S \times A \times S \mapsto \mathbb{R}$ denotes the reward function, and v is the characteristic function of the coalition. The detailed definitions of coalition games with CMDP are as follows:

- *Player:* Consider a coalition game with a set of N agents of players (e.g., DAG nodes) denoted by $\mathcal{N} = \{1, ..., N\}$. Each player's agent i is associated with an action and a state.
- *State:* As the DAG assumes that the offloading state of the current node e_n is only related to the state of the previous node e_{n-1}, we design the offloading state based on the PoW rule between e_n and e_{n-1}. The state $s_{e,n-1}$ in node e_{n-1} consists of two parts, represented as $s_{e,n-1} = (\ell^t_{pw}, \ell^{ac}_{pw})$, where ℓ^t_{pw} and ℓ^{ac}_{pw} are the probabilities abiding by PoW for training time and accuracy. $\mathcal{C}_{e,n}$ represents the observation of the evolutionary coalition state, describing a set of coalition players. We denote the offloading state as $S = \{(s_{e,n}, \mathcal{C}_{e,n}) | n = 1, 2, ..., h_{\max} + 1\}$, which consists of the local offloading states and the evolutionary coalition states of all nodes participating in the coalition.
- *Action:* In the MCTO based on the DAG G, the previous node offloads the task to the next hop node, which then confirms receipt. For the next hop node e_n, the offloading action $a_{e,n}$ determines which previous hop neighbor node should be selected to receive the offloaded tasks from, denoted as $a_{e,n} = \{a^0_{e,n}, a^1_{e,n}\}$. Specifically, when the current node e_n takes action $a^0_{e,n}$, it implies that the previous hop node e_{n-1} is not selected for the offloading. Otherwise, the node e_n takes the action $a^1_{e,n}$ to select the previous hop node e_{n-1} as a coalition member to receive its offloaded tasks. Correspondingly, we denote the joint offloading action as $A = \{a_{e,n} | n = 1, 2, ..., h_{\max} + 1\}$, which consists of the local offloading actions of all nodes participating in the collaborative offloading in MCTO.
- *State Transition Probability:* The state transition probability $\Pr(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, \mathcal{C}_{e,n-1}, a_{e,n-1})$ is defined as the probability of tasks offloading from the previous node e_{n-1} to the current node e_n, after taking an action $a_{e,n-1}$, satisfying the condition $\sum_{s_{e,n}, \mathcal{C}_{e,n} \in S} \Pr(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, \mathcal{C}_{e,n-1}, a_{e,n-1}) = 1$.
- *Constraint:* A compliance tolerance for PoW in the offloading transmission k should be within the available range. We define the compliance tolerance as C_{pw}, and express it as $\Phi_k \geq \phi_{th}$, where ϕ_{th} is the probability threshold for the compliance tolerance of PoW.
- *Reward:* Once the player agent in the node e_n takes an offloading action under the current offloading state, the agent will obtain an instant reward for the task, and all agents will share a common reward $r_{e,n}$. The objective of the offloading node selection is to minimize the total offloading latency cost under the constraints of PoW. Therefore, we define the offloading cost on each transmission as follows

$$\mathbb{C}_{e,n} = \ln\left[1 + \lambda_d (t^{sr}_k + t^q_k + t^e_k) + \lambda_s \Phi_k\right], \tag{6}$$

where the second component represents the latency associated with offloading models. The third component represents the PoW compliance tolerance in the offloading transmission k. λ_d and λ_s are non-negative coefficients used to control the weight of latency and trust risk. To minimize offloading costs, the reward function is designed as follows

$$r_{e,n} = R(s_{e,n}, \mathcal{C}_{e,n}, s_{e,n-1}, \mathcal{C}_{e,n-1}, a_{e,n-1}) = -\mathbb{C}_{e,n}. \tag{7}$$

For the empty coalition $\mathcal{C}_{e,n} = \mathcal{C}_{e,n-1} = \emptyset$, $R(s_{e,n}, \mathcal{C}_{e,n}, s_{e,n-1}, \mathcal{C}_{e,n-1}, a_{e,n-1}) = 0$. This implies that all nodes leave a coalition based on their individual rewards.
- *Characteristic function:* $v : 2^{\mathcal{N}} \to \mathbb{R}$ is a real-valued function. For any coalition $\mathcal{C} \subseteq \mathcal{N}$, $v(\mathcal{C})$ represents the total reward for the set $\mathcal{C}$ of players when they cooperate, and $v(\emptyset) = 0$. The characteristic function value of any coalition $\mathcal{C}_{e,n}$ given the current state and action is expressed as

$$\begin{aligned} &v(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, a_{e,n-1}) \\ &= \mathbb{E}\left[\sum_{i=k}^{h_{\max}} \gamma^{i-k} R^i(s_{e,n}, \mathcal{C}_{e,n}, s_{e,n-1}, \mathcal{C}_{e,n-1}, a_{e,n-1}) | \pi\right]. \end{aligned} \tag{8}$$

D. Problem Formulation

In this paper, we focus on facilitating multi-hop cooperative task offloading to improve computational performance by minimizing latency costs while ensuring trust in model training and sharing among edge nodes. In the CMDP-based coalition game, each player agent in the DAG G needs to find a cooperative trust offloading policy that results in a joint action A given any state S, which maximizes the expected cumulative multi-hop cooperative node reward. The policy π of player agents in the DAG G is a mapping from their states to actions. To select trusted offloading nodes and obtain the available offloading path set under fake service attacks, our objective is to learn the optimal policy π^* by maximizing the expected cumulative multi-hop

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

cooperative node discounted reward, expressed as

$$\mathcal{P}1: \max_{\pi} \mathbb{E}\left[\sum_{i=k}^{h_{\max}} \gamma^{i-k} r_{e,n}^{i} | \pi\right], \tag{9a}$$

$$s.t.\ \ a_{e,n} = \{a_{e,1}, a_{e,2}, ..., a_{e,h_{\max}+1}\}, \tag{9b}$$

$$\left\{\ell_{pw}^{t} | t \in [t_{\min}, t_{\max}]\right\}, \tag{9c}$$

$$\left\{\ell_{pw}^{ac} | ac \in [ac_{\min}, ac_{\max}]\right\}, \tag{9d}$$

$$\Phi_k \geq \phi_{th}, \tag{9e}$$

$$a_{e,n} \in \{0,1\}, \forall n \in N. \tag{9f}$$

In (9a), $\mathbb{E}[\cdot]$ denotes the expectation over the varying parameters of the offloading node in the offloading transmission. $\gamma \in (0,1]$ represents the discount factor, which indicates the importance of the future node selection relative to the current node selection. The constraint (9b) guarantees that the number of nodes selected to confirm and receive offloading tasks meets the requirement. The constraints (9c) and (9d) ensure that the training time and accuracy of the model fall into the confidence interval. The constraint (9e) ensures that the probability of the compliance tolerance is greater than the predefined threshold ϕ_{th}. (9f) denotes the constraint on the offloading decision variables. The problem $\mathcal{P}1$ is a mixed-integer nonlinear programming problem (MINLP) and NP-hard due to its nonlinear objective function in $r_{e,n}$. The constraint conditions contain mixed integer variables, making it difficult to obtain the optimal solution for problem $\mathcal{P}1$.

IV. Soft-Actor-Critic Coalition Game

Based on the above definition of the CMDP coalition game problem $\mathcal{P}1$, this paper proposes a trusted node selection algorithm for multi-hop task offloading based on the soft-actor-critic (SAC) coalition game. Unlike traditional RL methods that focus solely on maximizing cumulative rewards, the SAC method not only maximizes cumulative rewards but also maximizes policy entropy, offering advantages such as higher policy sampling efficiency and greater stability. SAC is applicable to CMDP with multi-modal rewards (latency and compliance tolerance for PoW), further enhancing the exploration and stability.

A. Shapley Value Reward for SAC Coalition Game

By interacting with the environment and utilizing deep neural network (DNN) models as well as the deterministic policy gradient (DPG), deep reinforcement learning (DRL) algorithms are capable of discovering an optimal policy and approximating the value function. However, function approximation errors can lead to overestimated values and suboptimal policies. To avoid potential suboptimal policies, SAC explores the stochastic policy gradient (SPG) and the maximum entropy framework of the policy to induce a probability distribution over actions for a given state, encouraging exploration of the action space [15]. However, SAC is unfair in the division of rewards among player agents because it does not take into account the contribution of each player agent to the coalition. In the following, we present using Shapley value to fairly divide the total reward among all player agents. It provides an unbiased assignment of values to player agents to encourage them to participate in trusted cooperation for model training while countering fake service attack behaviors. The Shapley value of player agent i is represented as a weighted mean over all $\mathcal{C} \subseteq \mathcal{N}$ of the coalition game's participants $\mathcal{N}$, and it can be expressed as follows

$$\begin{aligned}&\hat{\varphi}_i(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, a_{e,n-1}) \\ &= \sum_{\mathcal{C}_{e,n} \in \mathcal{N}} \frac{|\mathcal{C}_{e,n}|!(N - |\mathcal{C}_{e,n}| - 1)!}{N!} \Delta v(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, a_{e,n-1}),\end{aligned} \tag{10}$$

where N is the number of player agents in the coalition $\mathcal{C}_{e,n}$. $\Delta v(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, a_{e,n-1})$ represents the marginal contribution of player agent i to the coalition $\mathcal{C}_{e,n}$, which can be calculated as

$$\begin{aligned}&\Delta v(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, a_{e,n-1}) \\ &\quad = v(s_{e,n}, \mathcal{C}_{e,n} \cup \{i\} | s_{e,n-1}, a_{e,n-1}) \\ &\qquad - v(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, a_{e,n-1}).\end{aligned} \tag{11}$$

Equation (11) is used to determine whether the player agent i can increase the coalition $\mathcal{C}_{e,n}$ reward when it takes an action $a_{e,n-1}$ to join the coalition. Because the Shapley value has symmetry and additivity [29], we further normalize the Shapley value of each member (player agent) to

$$\begin{aligned}&\varphi_i(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, a_{e,n-1}) \\ &\qquad = \frac{\hat{\varphi}_i(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, a_{e,n-1})}{\sum_{i \in \mathcal{N}} \hat{\varphi}_i(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, a_{e,n-1})}.\end{aligned} \tag{12}$$

Thus, we can obtain each player agent's reward on each offloading transmission. It can be expressed as

$$\hat{r}_{e,n} = \varphi_i(s_{e,n}, \mathcal{C}_{e,n} | s_{e,n-1}, a_{e,n-1}). \tag{13}$$

B. Proposed Multi-Agent SAC Coalition Game

The traditional RL algorithm only maximizes the expected sum of rewards. By combining the coalition reward with the expected entropy of the policy $\pi(s_{e,n-1}, \mathcal{C}_{e,n-1})$, the SAC coalition game can favor stochastic exploration of state and action spaces. In other words, at each offloading decision, the player agent (response node e_n) will stochastically select a set of offloading actions when multiple request nodes' offloading actions need to be chosen, ensuring that no trusted node is missed while strengthening learning robustness. Thus, in the SAC coalition game, we consider a general maximum entropy objective to find an optimal policy π^* that maximizes both the coalition reward and entropy. For convenience of representation, we let $S_{e,n-1} = (s_{e,n-1}, \mathcal{C}_{e,n-1})$. The optimal policy is expressed as

$$\begin{aligned}\pi^* &= \arg\max_{\pi} \mathbb{E}\left[\sum_{n=2}^{h_{\max}} \hat{r}_{e,n-1} + \alpha \mathcal{H}\left(\pi(\cdot | S_{e,n-1})\right)\right] \\ &= \arg\max_{\pi} \mathbb{E}\left[\sum_{n=2}^{h_{\max}} \hat{r}_{e,n-1} - \alpha\left(\log \pi\left(a_{e,n-1} | S_{e,n-1}\right)\right)\right],\end{aligned} \tag{14}$$

where α is the temperature parameter that can control the weight of entropy relative to the coalition reward. $\pi(\cdot | S_{e,n-1})$ is the action distribution that measures the probability of taking any action under policy π in state $S_{e,n-1}$. The entropy $\mathcal{H}(\pi(\cdot | S_{e,n-1}))$ indicates the uncertainty of the action distribution of policy π in state $S_{e,n-1}$ and is defined as $\mathcal{H}(\pi(\cdot | S_{e,n-1})) = -\log(\pi(a_{e,n-1} | S_{e,n-1}))$. The soft policy iteration converges to the maximum coalition reward and entropy policy π^* in (14).

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

Correspondingly, the cumulative entropy-integrated reward is called the soft reward, which is expressed as

$$r_{soft} = \hat{r}_{e,n-1} + \gamma\alpha\mathbb{E}\left[\mathcal{H}\left(\pi(\cdot|S_{e,n})\right)\right]. \tag{15}$$

For policy π, the Q-value is computed iteratively by repeatedly applying the Bellman backup equation. Its Q-function is expressed as

$$Q(S_{e,n-1}, a_{e,n-1}) = \hat{r}_{e,n-1} + \gamma\mathbb{E}\left[Q(S_{e,n}, a_{e,n})\right]. \tag{16}$$

Furthermore, by replacing $\hat{r}_{e,n-1}$ in (16) with r_{soft} in (15), and using $\mathcal{H}(\pi(\cdot|S_{e,n})) = -\log(\pi(a_{e,n}|S_{e,n}))$ according to (14), the soft Q-function can be written as

$$\begin{aligned} &Q_{soft}(S_{e,n-1}, a_{e,n-1}) \\ &\quad= \hat{r}_{e,n-1} + \gamma\mathbb{E}\left[Q_{soft}(S_{e,n}, a_{e,n})\right] \\ &\qquad\qquad + \gamma\alpha\mathbb{E}\left[\mathcal{H}\left(\pi(\cdot|S_{e,n})\right)\right] \\ &\quad= \hat{r}_{e,n-1} + \gamma\mathbb{E}\left[Q_{soft}(S_{e,n}, a_{e,n})\right. \\ &\qquad\qquad \left. - \alpha\log\left(\pi(a_{e,n}|S_{e,n})\right)\right]. \end{aligned} \tag{17}$$

Therefore, according to (17), we can derive that the Q-function with the bounded augmented-entropy will converge to the soft Q-value of the offloading policy $\pi \in \Pi$. To maximize the coalition reward and the entropy of the offloading policy, it is necessary not only to iteratively evaluate the policy but also to ensure that the selected policy leads to an increase in the reward. The improved offloading policy ensures that a new offloading policy π_{new} has a higher reward than the current offloading policy π_{old} under a fake service attack. Since both the offloading policy π and the exponential of the Q-function are probability distributions, we can use Kullback-Leibler (KL) divergence to update the offloading policy towards the exponential of the soft Q-function. Given the state $S_{e,n-1}$, in each policy improvement step, the offloading policy can be updated according to

$$\pi_{new} = \arg\min_{\pi\in\Pi} D_{KL}\left(\pi(\cdot|S_{e,n-1}) \,\middle\|\, \frac{\exp\left(\mathrm{Q}^{\pi_{\mathrm{old}}}(S_{e,n-1}, \cdot)\right)}{\mathrm{Z}^{\pi_{\mathrm{old}}}(S_{e,n-1})}\right), \tag{18}$$

where $D_{KL}(\cdot\|\cdot)$ denotes the KL divergence that projects the improved policy into the desired set of offloading policies by using the information projection, $\mathrm{Z}^{\pi_{\mathrm{old}}}(S_{e,n-1})$ is utilized to normalize the exponential of the Q-function as a probability distribution. According to (18), the goal of the soft policy evaluation and improvement is to find a new offloading policy in the set of policies such that $Q^{\pi_{new}}(S_{e,n-1}, a_{e,n-1}) > Q^{\pi_{old}}(S_{e,n-1}, a_{e,n-1})$ holds for all $(S_{e,n-1}, a_{e,n-1}) \in S \times A$ while minimizing the KL divergence.

The multi-hop task offloading with continuous multiple node selection and cooperation needs to approximate the soft offloading policy under a fake service attack. To avoid overestimating the Q-value, we use DNNs as function approximators for the soft Q-function $Q(S_{e,n-1}, a_{e,n-1})$ and the offloading policy $\pi(a_{e,n-1}|S_{e,n-1})$. The DNN parameters θ of the soft Q-function can be trained to minimize the soft Bellman residual

$$J_Q(\theta) = \left[\frac{1}{2}\left(Q_\theta(S_{e,n-1}, a_{e,n-1}) - \hat{Q}(S_{e,n-1}, a_{e,n-1})\right)^2\right], \tag{19}$$

where

$$\hat{Q}(S_{e,n-1}, a_{e,n-1}) = \hat{r}_{e,n-1} + \gamma\mathbb{E}\left[V(S_{e,n})\right], \tag{20}$$

and

$$V(S_{e,n}) = \mathbb{E}\left[Q(S_{e,n}, a_{e,n}) - \alpha\log\pi(a_{e,n}|S_{e,n})\right]. \tag{21}$$

Here, $\hat{Q}(S_{e,n-1}, a_{e,n-1})$ is the target soft Q-function with parameters $\bar{\theta}$, which can be obtained by calculating the exponentially moving average of the soft Q-function weights. The soft Q-function can be optimized by the stochastic gradient

$$\begin{aligned} \hat{\nabla}J_Q(\theta) = \nabla Q_\theta(S_{e,n-1}, a_{e,n-1})\,&\big(Q_\theta(S_{e,n-1}, a_{e,n-1}) \\ &- \left(\hat{r}_{e,n-1} + \gamma(Q_{\bar{\theta}}(S_{e,n}, a_{e,n}) - \alpha\log\pi_\psi(a_{e,n}|S_{e,n}))\right)\big), \end{aligned} \tag{22}$$

Given two distributions π_1, π_2, the expected KL divergence between them is expressed as $D_{KL}(\pi_1||\pi_2) = \mathbb{E}[\log\frac{\pi_1}{\pi_2}]$. Parameters ψ of the offloading policy can be updated by minimizing the expected KL divergence in (18):

$$J_\pi(\psi) = \mathbb{E}\left[\log\frac{\pi_\psi(\cdot|S_{e,n-1})}{\left(\frac{\exp(\mathrm{Q}_\theta(S_{e,n-1},\cdot))}{\mathrm{Z}_\theta(S_{e,n-1})}\right)}\right]. \tag{23}$$

If we immediately apply the likelihood ratio gradient to (23) to sample policy gradients, the variance of the probability distribution of the offloading policy π will increase. In particular, because the output of the policy network is a Gaussian distribution, to reduce this variance, a random variable ξ following a Gaussian distribution with zero mean and unit variance is used to reparameterize the action as $a_{e,n-1} = \hat{a}_\psi(S_{e,n-1}, \xi) = \tanh(\mu_\psi + \xi\sigma_\psi)$, where μ_ψ is the mean and σ_ψ denotes the standard deviation. $\hat{a}_\psi(S_{e,n-1}, \xi)$ represents a sample from the probability distribution of the player agent's action. The $\tanh(\cdot)$ function compresses the action space into the [-1,1] range. Additionally, $Z(\cdot)$ in (23) can be ignored since it is not related to the parameters ψ. According to the definition of KL divergence, $J_\pi(\psi)$ in (23) is changed to

$$\begin{aligned} J_\pi(\psi) = \mathbb{E}\big[&\log\pi_\psi\left(\hat{a}_\psi(S_{e,n-1}, \xi)|S_{e,n-1}\right) \\ &-Q_\theta\left(S_{e,n-1}, \hat{a}_\psi(S_{e,n-1}, \xi)\right)\big]. \end{aligned} \tag{24}$$

The offloading policy in (24) can be optimized using stochastic gradient

$$\begin{aligned} \hat{\nabla}J_\pi(\psi) = &\nabla_\psi\log\pi_\psi(a_{e,n-1}|S_{e,n-1}) \\ &+ \big(\nabla_{a_{e,n-1}}\log\pi_\psi(a_{e,n-1}|S_{e,n-1}) \\ &-\nabla_{a_{e,n-1}}Q(S_{e,n-1}, a_{e,n-1})\big)\nabla_\psi\hat{a}_\psi(S_{e,n-1}, \xi). \end{aligned} \tag{25}$$

In (14), the parameter α affects the trade-off between the coalition reward and entropy. If its value is fixed, the training process may become unstable in some states. Therefore, we need to automatically adjust α to maximize the expected cumulative reward while ensuring a higher entropy to accelerate learning with more exploration under uncertain states. The entropy of the current policy must satisfy a minimum expected entropy constraint, that is: $\mathcal{H}(\pi(\cdot|S_{e,n-1})) \geq \mathcal{H}_{\min}$, where $\mathcal{H}_{\min}$ represents the predetermined minimum expected entropy. We apply canonical primal-dual optimization to iteratively update the parameter α as follows

$$\alpha = \alpha - \lambda\mathbb{E}\left[\left(\mathcal{H}_{\min} - \mathcal{H}\left(\pi(\cdot|S_{e,n-1})\right)\right)\right]^+, \tag{26}$$

where λ is the learning rate and $[x]^+ = \max\{0, x\}$.

C. SAC Coalition Game Algorithm for MCTO

In this section, we propose a distributed SAC coalition game algorithm to select multi-hop task offloading nodes in DEC computing. As shown in Fig. 4, each edge node e_n in the DAG acts as a player agent that observes the state of the preceding

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

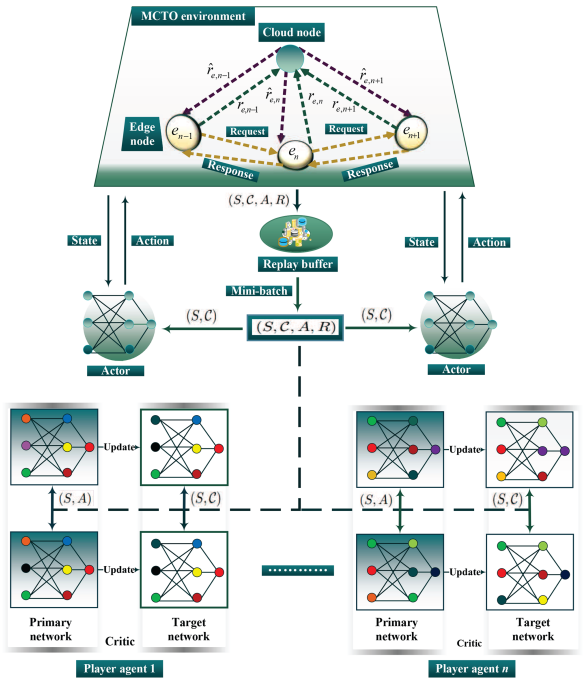


Fig. 4. The architecture of the SAC coalition game.

hop node e_{n-1} and takes actions to confirm requests and offload models from node e_{n-1} to the current node e_n. The offloading actions and states are stored in the replay buffer of each edge node without being uploaded to the cloud node. The cloud node collects and computes the reward $r_{e,n}$ for the player agents and uses the Shapley value to allocate rewards $\hat{r}_{e,n}$ to the player agents in the coalition.

1) Soft Actor-Critic Architecture for MCTO: As each player agent incurs offloading requests to maximize its reward, the current node selects previous-hop nodes to join the coalition and form a cooperative team for model training. Each player agent chooses an action to receive trusted offloaded tasks to combat fake service attacks while minimizing model training latency. As illustrated in Fig. 4, the player agent comprises two primary critic networks, two target critic networks, and an actor network. The actor network is responsible for learning the policy $\pi_\psi(a_{e,n-1}|S_{e,n-1})$ based on the stochastic gradient $\hat{\nabla} J_\pi(\psi)$ and outputs the means and standard deviations to determine the action based on the input state $S_{e,n-1}$. By executing this action to interact with the MCTO environment, the player agent obtains the reward and the next state $S_{e,n}$, then stores $(S_{e,n-1}, a_{e,n-1}, \hat{r}_{e,n-1}, S_{e,n})$ in the replay buffer $\mathcal{Z}$ as experience.

To overcome the problem of overestimation and improve the offloading policy, two critic networks are used to evaluate the actions generated by the actor network based on the soft Q-function. The primary critic network is responsible for estimating $Q_\theta(S_{e,n-1}, a_{e,n-1})$, while the target critic network, parameterized by $\bar{\theta}$, outputs a target Q-value based on the input action and state to update $Q_\theta(S_{e,n-1}, a_{e,n-1})$.

2) Algorithm Detail for MCTO: The details of the SAC coalition game for MCTO are shown in Algorithm 1. It includes the model training process and the offloading execution process. Steps 5-11 perform the model training process; at each hop transmission, the player agent needs to observe the state of the request nodes and take an offloading action. The cloud node allocates rewards to the player agents after they execute actions, and the player agents store these experiences in the replay buffer. Furthermore, steps 12-19 optimize the actor network parameters ψ and the two critic network parameters $\theta_i, \bar{\theta}_i, i = 1, 2$ by performing mini-batch stochastic gradient descent iterations with learning rates to approximate the expectations $J_\pi(\psi)$ and $J_Q(\theta_i)$. The target critic networks are updated according to parameters $\bar{\theta}_i$. $\kappa \in (0, 1)$ is the soft update coefficient for the target critic networks, which is kept small to ensure that the target critic networks update slowly to output accurate values. Steps 20-25 conduct the offloading execution process, during which the player agent takes an action to confirm the offloading and receives a model to train from the request nodes. These request nodes, confirmed by the player agent, are selected as trusted

Algorithm 1: SAC Coalition Game for MCTO.

Input: The task request node and the maximum hop h_{max}.
Output: Nodes selected by taking action and merging these nodes into coalition $\mathcal{C}$.

1 **Model training process**:
2 **Initialization**: For each player agent, actor network parameters ψ for offloading policy, primary critic network parameters θ_i, where $i = 1, 2$, target critic network parameters $\bar{\theta}_i$, and replay buffer $\mathcal{Z}$.
3 **foreach** *epoch* **do**
4 Initialize the node e_{n-1} as the request node and e_n as the response node, with the environment state $S_{e,n-1}$.
5 **foreach** *hop* **do**
6 Each player agent observes node $s_{e,n-1}$'s state $S_{e,n-1}$.
7 Each player agent selects an offloading action according to $a_{e,n-1} \sim \pi_\psi(\cdot|S_{e,n-1})$.
8 Execute the action $a_{e,n-1}$ and obtain reward $r_{e,n-1}$, then reach a next hop state $S_{e,n}$.
9 Upload the reward $r_{e,n-1}$ to the cloud node and receive allocated reward $\hat{r}_{e,n-1}$ from the cloud node.
10 Store experience $(S_{e,n-1}, a_{e,n-1}, \hat{r}_{e,n-1}, S_{e,n})$ in the replay buffer $\mathcal{Z}$.
11 **end**
12 **foreach** *player agent* **do**
13 Sample a mini-batch randomly from $\mathcal{Z}$.
14 Update the actor network parameters ψ to learn the policy using Eq. (25).
15 Update primary critic networks parameters θ_i using Eq. (22) for $i = 1, 2$.
16 Update target critic networks parameters $\bar{\theta}_i \leftarrow \kappa\theta_i + (1-\kappa)\bar{\theta}_i$ for $i = 1, 2$.
17 Optimize temperature parameter α using Eq. (26).
18 **end**
19 **end**
20 **Offloading execution process**:
21 All player agents load the trained models of critic networks and actor networks.
22 **foreach** *hop* **do**
23 Each player agent selects an action based on the output of the actor networks.
24 Execute actions to confirm and receive the models.
25 **end**

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

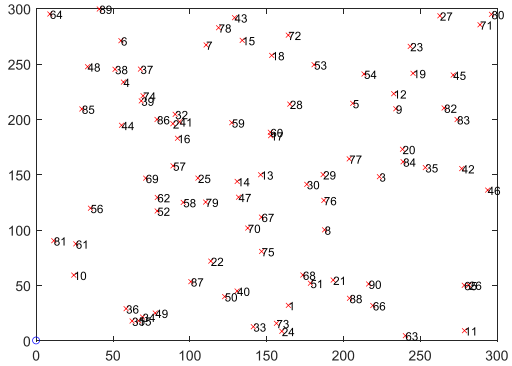


Fig. 5. The simulation scenario and the node deployment diagram.

cooperative nodes for training the model in the next round. The cooperative coalition members are updated by merging the selected nodes after they are determined by the offloading action.

D. Coalition Stability Analysis

We define the initial and h-hop iteration coalition structures as $\mathcal{C}^{(0)}$ and $\mathcal{C}^{(h)}$, respectively. Each player participating in the SAC coalition game is rational and aims to obtain its maximum total reward. A trusted SAC coalition $\mathcal{C}^{(*)}$ is an h-hop stable MCTO coalition if the following condition is met: for each $e_n \in V$, there does not exist any MCTO coalition $\mathcal{C}$ such that $v(\mathcal{C}) > v(\mathcal{C}^{(*)})$ and $h < h_{\max}$. Therefore, the initial coalition structure $\mathcal{C}^{(0)}$ reaches the stability at $\mathcal{C}^{(*)}$ after the h-hop iterations, where no player has an incentive to take further action to select nodes. The h-hop stable MCTO coalition structure represents a Nash equilibrium. In the process of the MCTO coalition resisting fake service attacks, each game player uses SAC to obtain the optimal node selection strategy, thereby improving the reward of the MCTO coalition and accelerating convergence to an h-hop stable solution.

E. Complexity Analysis

Algorithm 1 includes five DNNs to approximate an actor network, two primary critic networks, and two target critic networks. Thus, the computational complexity is $\mathcal{O}(5\hat{a}\Upsilon\hat{b}\varpi)$, where $\hat{a}$ denotes the number of layers, Υ represents the number of neurons in each layer, $\hat{b}$ is the number of iteration rounds, and ϖ represents the mini-batch size. Because each player agent needs to upload rewards to the cloud node and receive rewards from the cloud node at each training round, the communication overhead of Algorithm 1 is $\mathcal{O}(\hat{b}\sum_{e_i\in\mathcal{E}} 2\omega)$, where ω represents the data size. For Algorithm 1, the state and action spaces are stored in memory for each player agent to train networks or select actions, and the space complexity is $\mathcal{O}(|S| \cdot |A|)$, where $|S|$ and $|A|$ denote the size of the state space and the action space, respectively.

V. Performance Evaluation and Analysis

In this section, we conduct simulations to evaluate the performance of the proposed node selection scheme for MCTO against fake service attacks. Simulation experiments consider a multi-hop offloading scenario and a node deployment diagram as shown in Fig. 5, where edge nodes and cloud nodes are randomly distributed within a 300m×300 m region with 90 edge nodes and 10 cloud nodes. For the offloading task, model sizes for the training task are distributed in the range [413, 2090] KB. The number of CPU cycles per byte for the training task is set to 0.8 cycles/byte. The computation capacity for each edge node is set to 2.41 GHz. For the wireless transmission of the offloading task, we set the offloading power for each edge node to 0.58 W and its channel bandwidth to 6 MHz. We set the noise power for the offloading environment to −70 dBm and the channel gain to 1.5 dB. We implement the proposed algorithm using PyTorch, which is based on Python 3.7, on a computer with an Intel 4-Core i5-7200U processor and 32 GB RAM.

TABLE III
Simulation Hyperparameters in our Algorithms

Parameters	Value
Discounted factor	0.99
Actor and critic optimizer	Adam
Actor, critic, and Q-learning rate	1e-3,1e-2,1e-2
Buffer size	6000
Buffer minimal size	300
Batch size	90
Num episodes	100
Soft coefficient	0.006

A. Results Analysis

To evaluate the performance of our proposed algorithm, we consider four peer cooperative offloading policies: (1) Deep Deterministic Policy Gradient (DDPG) [25]: This method consists of an actor and a critic. The actor network outputs a deterministic action based on the state of the current nodes, while the critic network evaluates the value using the Q-function. DDPG utilizes a deterministic policy but is not suitable for environments under attack. (2) Intelligent Path Selection without considering PoW (IPS) [26]: It uses the Q-learning algorithm to select a path and update the Q-value at each time step. This scheme does not apply the stochastic policy gradient or reward allocation based on Shapley values to optimize the action space of the player agent. (3) Real-time Task Allocation Strategy (RTTAS) [27]: This scheme divides subtasks and allocates them among mobile edge computing servers, optimizing the trade-off between time delay and energy consumption, without considering fake service attacks on edge nodes. (4) Random Node Selection (RNS): This is a random node selection scheme for multi-hop cooperative task offloading in DEC.

We deploy the actor and critic on each player agent, which encompasses an input layer, an output layer, and a hidden layer. The hidden layer is set to 180 neurons. The simulation parameters and scenarios are derived from the literature [30], [31]. We set other hyperparameters in our model as shown in Table III.

Attackers utilize fake service attacks to achieve sophisticated attacks. The target region in Fig. 5 is divided into eight subareas. The player agents in these subareas use our proposed algorithm to form eight coalitions based on assigned rewards computed by Shapley values. Fig. 6 shows the average observed reward using Shapley values for different coalitions, which is lower than the average allocated reward using Shapley values for different coalitions in Fig. 7. We can see that the average rewards allocated in coalitions 2 and 3 are greater than those in other coalitions. This indicates that coalitions 2 and 3 are the optimal coalitions for providing reconfiguration paths for multi-hop cooperative task offloading in DAG.

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

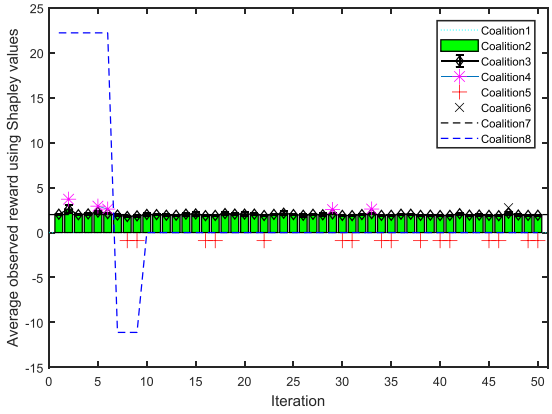


Fig. 6. Average observed reward using Shapley values for different coalitions.

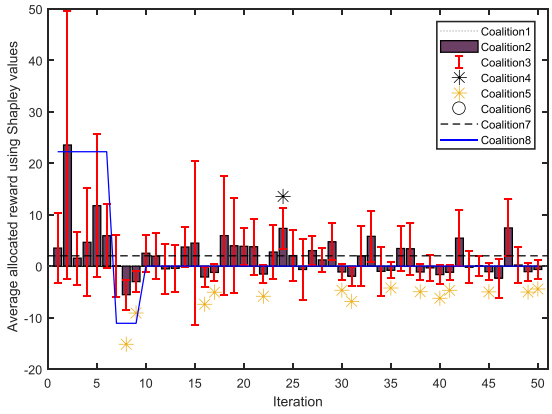


Fig. 7. Average allocated reward using Shapley values for different coalitions.

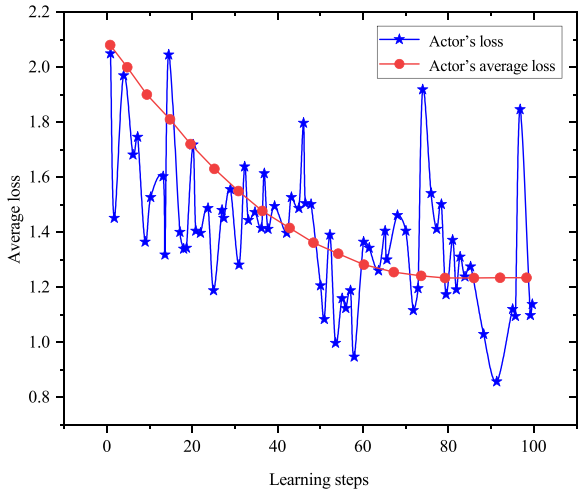


Fig. 8. Actor's average loss with varying learning steps.

To verify the learning performance of our proposed algorithm, we plot the training loss for the actor agent in Fig. 8. The average training loss of the actor agent decreases rapidly and reaches the lowest steady point as the learning step increases. Besides, we can see that when the learning step is 100, the average training loss of the critic agent also decreases and reaches a steady state (Fig. 9). This is because the actor and critic agents at each node

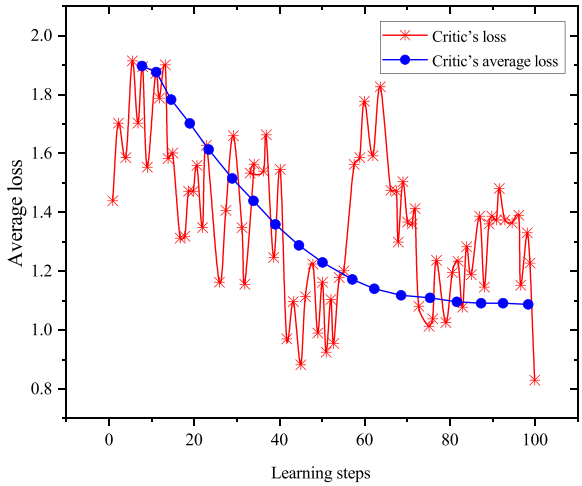


Fig. 9. Critic's average loss with different learning steps.

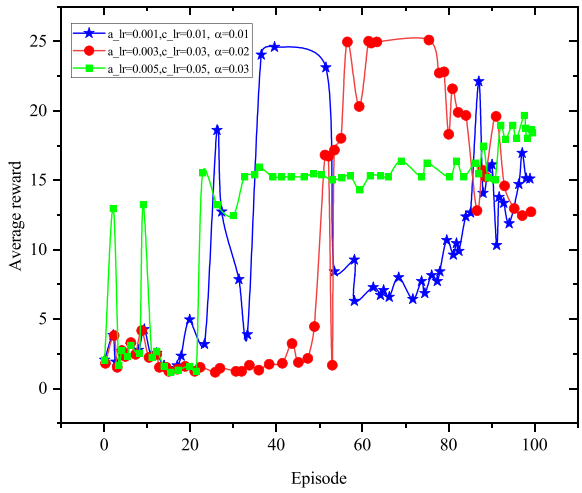


Fig. 10. Convergence of the proposed algorithm with different learning rates.

observe more local states and receive more rewards, allowing them to learn an optimal offloading action at each iteration step.

We evaluate the impact of the learning rate on our proposed algorithm in Fig. 10, where the cumulative reward under our proposed algorithm is shown with different actor network learning rates a_lr, different critic network learning rates c_lr, and different temperature parameters α. We can see that the training process eventually converges, although the learning rate has an impact on the algorithm's convergence speed. It is clearly observed that the convergence speed increases as the learning rate increases. Specifically, when $a\_lr = 0.005$, $c\_lr = 0.05$, and $\alpha = 0.03$, the convergence speed is faster compared to other learning rates. Therefore, setting a suitable learning rate is crucial for adjusting the weights of the actor and critic networks according to the gradient of the loss.

Fig. 11 shows the impact of the number of nodes under fake service attacks on the average multi-hop offloading latency. As expected, the DDPG-based offloading method has a higher average multi-hop offloading latency compared to our proposed algorithm. Furthermore, as the number of nodes increases, RTTAS and RNS exhibit faster upward spikes in latency compared

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

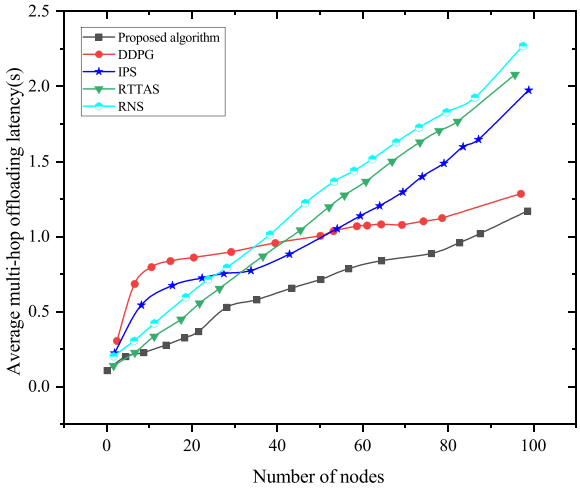


Fig. 11. Multi-hop offloading latency of five schemes with different numbers of nodes.

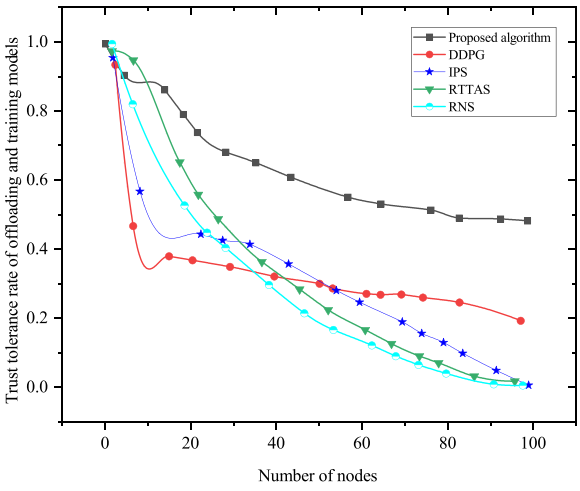


Fig. 12. Trust tolerance rate of offloading and training model with different the number of nodes.

to the DDPG-based offloading scheme, reaching approximately 2.2 seconds when the number of nodes is 98. In contrast, our proposed algorithm maintains an average multi-hop offloading latency of about 1 second. We also observe that when the number of nodes is less than 20, the average multi-hop offloading latency of DDPG and IPS is greater than that of the other three methods. This is because DDPG and IPS can select fewer nodes to offload tasks under attack, but their average multi-hop offloading latency is still lower than that of RTTAS and RNS when the number of nodes reaches 50. This phenomenon implies that the number of nodes significantly affects the average multi-hop offloading latency. RTTAS and RNS generate suboptimal offloading policies under fake service attacks. Additionally, our proposed algorithm achieves the lowest average multi-hop offloading latency under these attacks.

In Fig. 12, the trust tolerance rate of offloading and training models versus the number of nodes is illustrated. The trust tolerance rate decreases with an increase in the number of nodes under fake service attacks. When the number of nodes is small, all methods are shown to be ineffective in increasing the trust tolerance rate. Conversely, when the number of nodes exceeds 70, our proposed algorithm demonstrates significant

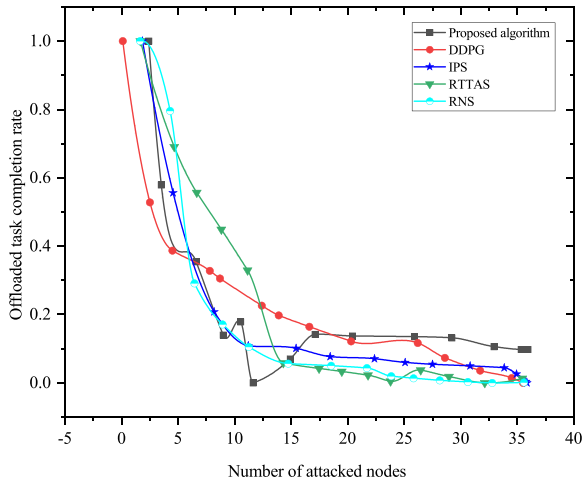


Fig. 13. Offloaded task completion with different numbers of attacked nodes.

effectiveness in enhancing the trust tolerance rate compared to the reference scheme, which does not perform intelligent node confirmations or path reconfiguration for offloading and training models. As a result, when the number of nodes reaches 100, the trust tolerance rates of DDPG, IPS, RTTAS, and RNS are 60%, 90%, 98%, and 92% lower than that achieved by our proposed algorithm.

Fig. 13 shows the impact of the number of attacked nodes on the offloaded task completion rate for different offloading approaches, where the number of attacked nodes is associated with the node selection for the offloading path in DAG. When we set the proportion of attacked nodes to 40%, the offloaded task completion rate of our proposed algorithm is the highest and is 1.5x, 1x, 4x, and 2.33x higher than that of DDPG, IPS, RTTAS, and RNS, respectively. DDPG is applied in a centralized manner and has low protection ability against attacks on offloading paths. The IPS method cannot ensure cooperative nodes follow PoW for offloading tasks. RTTAS considers the offloading performance of unattacked nodes. RNS may select fake service nodes as collaborators, thus decreasing the completion rate for offloading tasks and training models. Therefore, as the number of attacked nodes increases, the offloaded task completion rate of our proposed algorithm remains higher compared to the above methods.

Fig. 14 shows the efficiency of resource utilization compared to task offloading schemes. We can see that our proposed algorithm achieves higher resource utilization than other methods. Our proposed algorithm saves communication resources by uploading rewards to the cloud node without transmitting local states, which not only eliminates the impact of fake service attacks on edge nodes but also improves the performance of the DAG network. Other methods have lower resource utilization efficiency compared to our proposed algorithm, as DDPG, IPS, and RTTAS do not focus on optimizing resource utilization. The RNS method has the lowest efficiency because it randomly selects nodes for task offloading, leading to significant redundancy in offloading tasks.

B. Discussion

1) Fairness and Efficiency of PoW: In edge computing environments, the computational capabilities of different edge nodes exhibit significant variation. To ensure both efficiency and

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

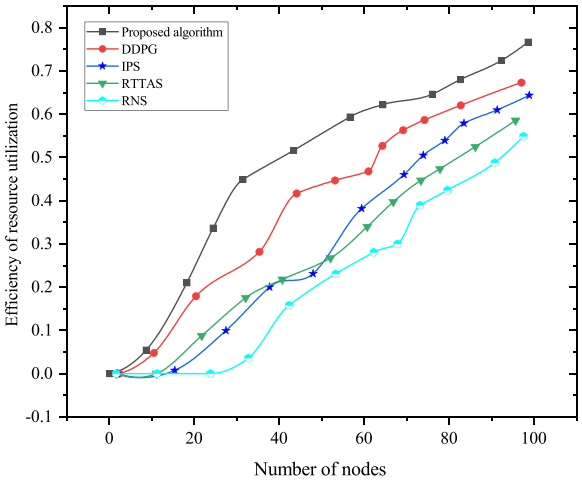


Fig. 14. Efficiency of resource utilization with different numbers of nodes.

fairness in training tasks, it is essential to adjust the difficulty of workload proof using dynamic strategies. This study employs an edge node selection strategy that prioritizes nodes with higher computational power for participation in training tasks. Participation eligibility is dynamically adjusted based on real-time workload performance metrics (e.g., training duration and model accuracy) to prevent underperforming nodes from compromising the overall system efficiency. In addition, through multiple rounds of training, weaker nodes are permitted to increase the number of local training iterations in order to compensate for limited computational resources and prevent failure caused by time-outs.

2) Adjustment of Flexible SAC Coalition: Multi-hop cooperative task offloading based on DEC constitutes a chained service architecture. This study draws on the fundamental principle from public blockchain systems that the proportion of malicious nodes must remain strictly below 50%. In simulation settings, this threshold is typically configured within the range of 30% to 45% to evaluate network resilience. When the proportion of fake nodes exceeds the threshold set in this paper, the SAC coalition formation mechanism and the dynamic adjustment of the compliance tolerance threshold for PoW are used to enable 2/3 of the high-computing-power nodes to form a service chain coalition to participate in the training tasks.

3) Flexible DAG Task Processing: When a new task arrives, the current node verifies the submitted model through PoW. After verification, the new node is merged into the existing DAG through the SAC coalition formation algorithm, and the training task execution sequence is dynamically adjusted. The PoW of the edge nodes is continuously monitored. If the probability of the compliance tolerance of PoW exceeds the threshold, it is marked as unavailable, and the coalition members are dynamically adjusted, and removed from the sequence. After each successful completion of a trusted model training task by an edge node, the node is merged into the SAC coalition in real-time.

4) Deployment Limitations and Scalability of MCTO: In large-scale deployment scenarios, the increase in the number of nodes in multi-hop paths leads to exponential growth in communication hops, intensifies the strategy conflict problem in distributed reinforcement learning, and causes a sharp increase in the complexity of solving Nash equilibrium due to the independent decision-making of multiple agents. The model convergence time extends linearly with the increase in the number of nodes. For system scalability, response delays from cross-domain collaboration and heterogeneous resource scheduling require urgent optimization.

VI. Conclusion

In this paper, we propose an intelligent coalition game scheme to defend against fake service attacks for MCTO in device-edge-cloud computing. We consider two multi-hop task offloading approaches based on the DAG G and PoW technologies. The objective of this scheme is to minimize the time and costs of distributed model training under attack. To defend against fake service attacks from nodes inside MCTO, we formulate the node selection process for trusted cooperative training as a CMDP-based coalition game model. To allow the node selection decision to be made through multiple agents, we present a soft-actor-critic coalition game method with Shapley value reward allocation. This approach encourages player agents to participate in trusted cooperation while forming an effective cooperative coalition. Finally, the simulation results show that our proposed algorithm outperforms other approaches for MCTO under the constraints of cooperative time costs and internal attacks.

References

[1] B. Lin, Y. Huang, J. Zhang, J. Hu, X. Chen, and J. Li, "Cost-driven offloading for DNN-based applications over cloud, edge, and end devices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5456–5466, Aug. 2020.

[2] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2833–2849, Dec. 2020.

[3] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2169–2182, Feb. 2023.

[4] C. Chen, Y. Zeng, H. Li, Y. Liu, and S. Wan, "A multihop task offloading decision model in MEC-enabled Internet of Vehicles," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3215–3230, Feb. 2023.

[5] S. Warnat-Herresthal et al., "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, 2021.

[6] L. Xiao et al., "A reinforcement learning and blockchain-based trust mechanism for edge networks," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5460–5470, Sep. 2020.

[7] X. Zheng, M. Li, Y. Chen, J. Guo, M. Alam, and W. Hu, "Blockchain-based secure computation offloading in vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4073–4087, Jul. 2021.

[8] T. Dbouk, A. Mourad, H. Otrok, H. Tout, and C. Talhi, "A novel ad-hoc mobile edge cloud offering security services through intelligent resource-aware offloading," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1665–1680, Dec. 2019.

[9] G. H. S. Carvalho, I. Woungang, A. Anpalagan, and I. Traore, "Optimal security-aware virtual machine management for mobile edge computing over 5 G networks," *IEEE Syst. J.*, vol. 15, no. 3, pp. 3403–3414, Sep. 2021.

[10] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Secure computation offloading in blockchain based IoT networks with deep reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 4, pp. 3192–3208, Oct.–Dec. 2021.

[11] S. Xu, C. Guo, R. Q. Hu, and Y. Qian, "Blockchain-inspired secure computation offloading in a vehicular cloud network," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14723–14740, Aug. 2022.

[12] S. Yao et al., "Blockchain-empowered collaborative task offloading for cloud-edge-device computing," *J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3485–3500, Dec. 2022.

[13] J. Yan, S. Bi, and Y. J. A. Zhang, "Offloading and resource allocation with general task graph in mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5404–5419, Aug. 2020.

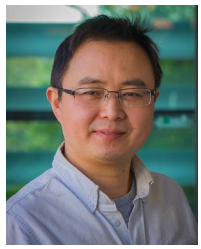
[14] T. Li, Y. Liu, T. Ouyang, H. Zhang, K. Yang, and X. Zhang, "Multi-hop task offloading and relay selection for IoT devices in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 24, no. 1, pp. 466–481, Jan. 2025.

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic:off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[16] Z. Wang, Z. Zhou, H. Zhang, G. Zhang, H. Ding, and A. Farouk, "AI-based cloud-edge-device collaboration in 6 G space-air-ground integrated power IoT," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 16–23, Feb. 2022.
[17] Y. Huang et al., "An integrated cloud-edge-device adaptive deep learning service for cross-platform web," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 1950–1967, Apr. 2023.
[18] H. Liao et al., "Cloud-edge-device collaborative reliable and communication-efficient digital twin for low-carbon electrical equipment management," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1715–1724, Feb. 2023.
[19] R. Wang, B. Yang, P. Zhang, H. Wang, Z. Yang, and D. Wu, "SEC-DEC: A social trust based video centric network leveraging secure device-edge-cloud collaborations," *IEEE Wireless Commun.*, vol. 29, no. 5, pp. 52–59, Oct. 2022.
[20] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1133–1145, May 2021.
[21] G. Feng, X. Li, Z. Gao, C. Wang, H. Lv, and Q. Zhao, "Multi-path and multi-hop task offloading in mobile ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5347–5361, Jun. 2021.
[22] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT-Edge-Cloud computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2759–2774, Dec. 2019.
[23] W. Huang, Z. Zhao, G. Min, and J. Chen, "Distributed multihop task offloading in massive heterogeneous IoT systems," *IEEE Trans. Comput.*, vol. 73, no. 4, pp. 1126–1137, Apr. 2024.
[24] Y. Deng, H. Zhang, X. Chen, and Y. Fang, "Multi-hop task routing in vehicle-assisted collaborative edge computing," *IEEE Trans. Veh. Technol.*, vol. 73, no. 2, pp. 2444–2455, Feb. 2024.
[25] L. P. Qian, H. Zhang, Q. Wang, Y. Wu, and B. Lin, "Joint multi-domain resource allocation and trajectory optimization in UAV-assisted maritime IoT networks," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 539–552, Jan. 2023.
[26] L. Zhang, X. Ma, Z. Zhuang, H. Xu, V. Sharma, and Z. Han, "Q-learning aided intelligent routing with maximum utility in cognitive UAV swarm for emergency communications," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3707–3723, Mar. 2023.
[27] J. Tang, T. Qin, Y. Xiang, Z. Zhou, and J. Gu, "Optimization search strategy for task offloading from collaborative edge computing," *IEEE Trans. Serv. Comput.*, vol. 16, no. 3, pp. 2044–2058, May/Jun. 2023.
[28] Z. Ji et al., "Multiple-task coded computing for distributed computation framework: Modeling and delay analysis," *IEEE Trans. Commun.*, vol. 72, no. 8, pp. 5032–5046, Aug. 2024.
[29] S. Han, H. Wang, S. Su, Y. Shi, and F. Miao, "Stable and efficient shapley value-based reward reallocation for multi-agent reinforcement learning of autonomous vehicles," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 8765–8771.
[30] X. Wu, X. Li, J. Li, P. C. Ching, V. C. M. Leung, and H. V. Poor, "Caching transient content for IoT sensing: Multi-agent soft actor-critic," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5886–5901, Sep. 2021.
[31] D. Wu, T. Liu, Z. Li, T. Tang, and R. Wang, "Delay-aware edge-terminal collaboration in green Internet of Vehicles: A multiagent soft actor-critic approach," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 1090–1102, Jun. 2023.

Jianhua Liu (Member, IEEE) received the PhD degree in computer application technology from Shanghai University, Shanghai, China, in 2012. He was a visiting scholar with the State University of New York at Buffalo State in 2014. From 2022 to 2023, he was a visiting scholar with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. He is currently an associate professor with the Department of Computer Science and Engineering, Shaoxing University, Shaoxing, China and research assistant with Network and Mobile Computing Lab, Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include distributed computing, mobile computing, Internet of Things, and dependable networking.

Xin Wang (Senior Member, IEEE) received the BS degree in telecommunications engineering and the MS degree in wireless communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 1990 and 1993, respectively, and the PhD degree in electrical and computer engineering from Columbia University, New York, NY, USA, in 2001. She was a member of Technical Staff, in the area of mobile and wireless networking with Bell Labs Research, Lucent Technologies, NJ and assistant professor with the Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY. She is currently an associate professor with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, NY. Her research interests include algorithm and protocol design in wireless networks and communications, mobile and distributed computing, and networked sensing and detection. She was with executive committee and technical committee of numerous conferences and funding review panels and an associate editor for *IEEE Transactions on Mobile Computing*. She was the recipient of the US National Science Foundation (NSF) Career Award in 2005 and ONR challenge Award in 2010.



Shui Yu (Fellow, IEEE) is currently a professor with the School of Computer Science, University of Technology Sydney, Australia. He initiated the research field of networking for Big Data in 2013 and his research outputs have been adopted by industrial systems. He has authored or coauthored three monographs and edited two books, more than 350 technical papers, including top journals and top conferences, such as *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Emerging Topics in Computing*, *IEEE/ACM Transactions on Networking*, and INFOCOM. His h-index is 48. His research interests includes Big Data, security and privacy, networking, and mathematical modelling. He is currently with a number of prestigious editorial boards, including *IEEE Communications Surveys and Tutorials* (Area Editor), *IEEE Communications Magazine*, and *IEEE Internet of Things Journal*. He is a member of AAAS and ACM and distinguished lecturer of IEEE Communication Society.

Guangtao Xue (Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2004. He is currently a professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include vehicular ad hoc networks, wireless networks, mobile computing, and distributed computing. He is a member of IEEE Communication Society.

Minglu Li (Fellow, IEEE) received the PhD degree in computer software from Shanghai Jiao Tong University, in 1996. He is currently a full professor and director with the Artificial Intelligence Internet of Things Center, Zhejiang Normal University. He is also the director with Network Computing Center, Shanghai Jiao Tong University. He has authored or coauthored more than 400 papers in academic journals and international conferences. His research interests include vehicular networks, Big Data, cloud computing, and wireless sensor networks. He was the chairperson of Technical Committee on Services Computing from 2004 to 2016 and Technical Committee on Distributed Processing from 2005 to 2017 of IEEE Computer Society in Great China region. He was a PC member of more than 50 international conferences, including IEEE INFOCOM from 2009 to 2016 and IEEE CCGrid in 2008. He was the General co-chair of IEEE SCC, IEEE CCGrid, IEEE ICPADS, and IEEE IPDPS and vice chair of IEEE INFOCOM.

Authorized licensed use limited to: SUNY AT STONY BROOK. Downloaded on January 09,2026 at 08:07:25 UTC from IEEE Xplore. Restrictions apply.