Hindawi
Security and Communication Networks
Volume 2021, Article ID 5532410, 13 pages
https://doi.org/10.1155/2021/5532410

Research Article

Reinforcement Learning for Security-Aware Workflow Application Scheduling in Mobile Edge Computing

Binbin Huang,[1] **Yuanyuan Xiang**,[1] **Dongjin Yu**,[1] **Jiaojiao Wang**,[2] **Zhongjin Li**,[1] **and Shangguang Wang**[3]

[1]*School of Computer, Hangzhou Dianzi University, Hangzhou 310018, China*
[2]*Institute of Intelligent Media Technology, Communication University of Zhejiang, Hangzhou 310018, China*
[3]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*

Correspondence should be addressed to Binbin Huang; huangbinbin@hdu.edu.cn and Dongjin Yu; yudj@hdu.edu.cn

Received 2 March 2021; Accepted 15 May 2021; Published 25 May 2021

Academic Editor: Xiaolong Xu

Mobile edge computing as a novel computing paradigm brings remote cloud resource to the edge servers nearby mobile users. Within one-hop communication range of mobile users, a number of edge servers equipped with enormous computation and storage resources are deployed. Mobile users can offload their partial or all computation tasks of a workflow application to the edge servers, thereby significantly reducing the completion time of the workflow application. However, due to the open nature of mobile edge computing environment, these tasks, offloaded to the edge servers, are susceptible to be intentionally overheard or tampered by malicious attackers. In addition, the edge computing environment is dynamical and time-variant, which results in the fact that the existing quasistatic workflow application scheduling scheme cannot be applied to the workflow scheduling problem in dynamical mobile edge computing with malicious attacks. To address these two problems, this paper formulates the workflow scheduling problem with risk probability constraint in the dynamic edge computing environment with malicious attacks to be a Markov Decision Process (MDP). To solve this problem, this paper designs a reinforcement learning-based security-aware workflow scheduling (SAWS) scheme. To demonstrate the effectiveness of our proposed SAWS scheme, this paper compares SAWS with MSAWS, AWM, Greedy, and HEFT baseline algorithms in terms of different performance parameters including risk probability, security service, and risk coefficient. The extensive experiments results show that, compared with the four baseline algorithms in workflows of different scales, the SAWS strategy can achieve better execution efficiency while satisfying the risk probability constraints.

1. Introduction

In recent years, with the explosive growth of smart devices (such as smart cameras, smart glasses, smart bracelets, and smart phones), a large number of advanced mobile applications (such as real-time navigation systems, interactive online games, virtual reality, and augmented reality) are emerging rapidly. In order to efficiently process these mobile applications, mobile devices need to be equipped with abundant computing resources and battery capabilities [1, 2]. However, due to the limited size of mobile devices, they are usually resource-constrained. Therefore, the conflict between the ever-increasing resource requirements of mobile applications and the limited resource capabilities of mobile devices brings great challenges to execute these mobile applications.

Mobile Edge Computing (MEC) as a new computing paradigm brings remote cloud resource to the edge servers nearby mobile users, enabling mobile users to offload partial or all computation tasks of mobile applications to edge servers for collaborative execution, and thereby greatly alleviating the conflict between resource supply and demand, effectively reducing the application completion time and the mobile devices' energy consumption [3–5].

Many mobile applications are typical workflow models, and they consist of a sequence of precedence-constrained

tasks. For example, a video streaming-based face recognition application mainly consists of motion detection and face recognition. The face recognition further consists of face detection, image preprocessing, feature extraction, and classification [3, 6]. In mobile edge computing, workflow application scheduling has a higher complexity in comparison to independent task scheduling [7–9]. In addition, it also faces two challenges for workflow application scheduling in mobile edge computing as follows. One is the edge environment dynamics, such as the time-varying channel quality and workload of edge servers, which can impact the workflow application scheduling decision. The other is the security problem of workflow application scheduling. Due to the open nature of the edge environment, the edge servers that aggregate an amount of user data frequently suffer from malicious attacks such as data leakage and tampering, which pose a serious threat to successfully execute these offloaded tasks [10–13]. Hence, it needs to employ various types of security services to effectively defend against the hostile attacks and protect these offloaded tasks. However, employing security services inevitably incurs additional security overhead, which will increase the completion time of workflow application. Therefore, it is a big challenge to design an efficient security-aware workflow scheduling scheme to reduce the completion time of workflow application while satisfying its security requirement.

To meet the aforementioned challenges, this paper formulates the security-aware workflow scheduling problem in MEC to be a Markov Decision Process (MDP) [14]. The environment state, which consists of the task list on each edge server, the workloads on each edge server, and the channel states between the mobile device and the edge servers, can be observed. Based on the environment state, the task nodes of the workflow are dynamically scheduled to edge servers. The deep reinforcement learning algorithm is suitable to solve decision-making problems with unknown prior knowledge [15–19]. To solve this problem, this paper proposes a deep reinforcement learning-based security-aware workflow scheduling scheme (SAWS). Its main objective is to optimize the completion time of workflow while satisfying its security requirement. To evaluate the effectiveness of the SAWS scheme, this paper implements average workload minimization (AWM), maximum SAWS (MSAWS), Greedy, and HEFT baseline algorithms. We compare the SAWS scheme with these four baseline algorithms under different risk probabilities, different security services, different risk coefficients, different edge server's computing capacities, and different number of edge servers. The experimental results demonstrate that the SAWS strategy can optimize the completion time of workflow application while satisfying the risk probability constraint. The main contributions of this paper can be summarized as follows:

- This paper focuses on the security problem of workflow scheduling in a dynamic edge computing, which is more complex than independent task scheduling.
- This paper formulates the security-aware workflow scheduling problem in mobile edge computing to be a finite Markov decision process, and its main objective is to minimize the completion time of workflow while satisfying the risk probability constraint.
- This paper proposes a deep Q-network-based security-aware workflow scheduling (SAWS) scheme to solve the workflow scheduling problem in a dynamic edge computing environment with malicious attacks. Extensive experimental results demonstrate that the SAWS scheme can greatly reduce the completion time of workflow application while satisfying the risk probability constraint.

The rest of this paper is organized as follows. In Section 2, the related work is summarized. In Section 3, the system model and problem formulation for security-aware workflow scheduling in MEC are presented. In Section 4, the deep reinforcement learning-based security-aware workflow scheduling scheme is described in detail. In Section 5, the simulation parameters are settled, and the experimental performance is analyzed. In Section 6, the work of this paper is concluded.

2. Related Work

The task offloading problem in the MEC has been studied in a lot of works. According to different optimization goals, these works can be classified into three categories. The first one is task offloading with the goal of optimizing the mobile device's energy consumption. For example, Huang et al. [7] propose a security and cost-aware task offloading scheme based on deep reinforcement learning for task offloading in single-user multiserver scenarios. Its main goal is to minimize the task processing delay and mobile device energy consumption while satisfying the security requirement for task. Chen et al. [20] formulate task offloading problem in single-user single-server scenario to be a stochastic optimization problem and decompose this problem into two deterministic optimization subproblems. To solve these two subproblems, a TOFEE algorithm is proposed to optimize the mobile device's energy consumption. Wu et al. [21] propose a Lyapunov optimization-based energy-efficient task offloading scheme to determine the operating position of the application, the objective of which is to minimize the average energy consumption of mobile devices while satisfying the average response time constraint. The second one is task offloading with the goal of optimizing the task processing delay. For example, Chalapathi et al. [22] propose a task scheduling scheme to solve the task offloading problem in multiple cloudlets, aiming at minimizing the task processing delay. Xu et al. [23] design an adaptive task offloading scheme, which leverages decomposition-based multiobjective evolutionary algorithms to generate feasible solutions, to optimize the task processing latency and resource utilization of edge system. The third one is task offloading with the goal of optimizing the weighted sum of the mobile device's energy consumption and the task processing delay. Wu et al. [24] propose a Lyapunov optimization-based energy-efficient task offloading scheme to control the computational and communication overheads

and further choose optimal computational location for the application to minimize energy consumption and task processing time. However, all above works mainly focus on the independent task scheduling in MEC. The task nodes of workflow are precedence-constrained. The above schemes are not suitable for workflow scheduling.

To further study the workflow scheduling problem in MEC, Xu et al. [25] construct a multiresource energy consumption model to solve the unity problem for traditional energy consumption model and propose a particle swarm algorithm-based energy-efficient multiresource workflow scheduling algorithm. Its main objective is to reduce the energy consumption of mobile devices while satisfying the completion time constraint for workflow. Wu et al. [26] construct a weighted resource sum graph based on resource consumption and further design a novel cost-efficient partitioning scheme, the objective of which is to find the optimal partitioning scheme to reduce execution time and energy consumption. Zhu et al. [27] formulate the workflow scheduling problem in MEC to be a joint optimization problem of energy consumption and time delay and adopt the deep Q network algorithm to solve the optimal scheduling scheme. However, the execution order of the workflow is assumed in advance, and how to calculate the execution order of workflow with precedence constraints is not introduced. In addition, this paper does not pay attention to the security problem of workflow scheduling in MEC. Liu [28] proposes a novel maximum probability function and deep Q network-based multiworkflow scheduling scheme to solve the scheduling problem in multiuser edge computing environment, which can find a high-quality workflow scheme in a dynamic environment. However, this paper does not pay attention to the security problem of workflow scheduling in dynamic MEC. Therefore, all the above scheduling schemes are not suitable for security-aware workflow scheduling in dynamic mobile edge computing.

With the escalation of data security threats in mobile edge computing [10–12, 29, 30], a lot of related works have taken some measures to protect security-critical applications and the large amount of data generated in mobile devices from malicious attacks. Huang [6] designs a workflow scheduling scheme based on Genetic Algorithms to minimize the mobile device's energy consumption under the completion time of workflow and risk probability constraints. Elgendy et al. [11] design a multidevice and single-server cooperative task offloading scheme to solve the security-aware multiuser resource allocation and task offloading problem. The goal is to minimize the time delay and energy consumption of the whole system. Jia et al. [31] design an identity-based anonymous authentication key agreement protocol to ensure the security of sensitive data in MEC. He et al. [32] design a security mechanism based on adaptive algorithms to solve the security problem of IoT applications in mobile edge computing. Chen et al. [33] propose a malicious application detection method based on deep learning on mobile devices, which greatly improves the security of mobile edge computing. Xu et al. [34] design a secure service offload approach to promote Internet of vehicles service utility and edge utility while ensuring privacy security in software-defined networks enabled edge computing. Xu et al. [35] adopt a location-sensitive-hash (LSH) method to encrypt the feature information for the offloaded services and further design s LSH-based offloading scheme, the goal of which is to minimize the energy consumption and response time of all services while guaranteeing the service security. All above researches mainly design security strategies from different points to ensure the security of edge computing, and they do not pay attention to the security problem of workflow scheduling in a dynamic edge computing with unknown prior knowledge. Aiming at this problem, this paper mainly focuses on security-aware workflow scheduling problem in dynamic mobile edge computing environment with security threats.

3. System Model and Problem Formulation

In this section, we first introduce the mobile edge computing model, security cost model, communication model, and risk probability model in mobile edge computing environment, respectively, and then describe the security-aware workflow scheduling problem in detail.

3.1. Mobile Edge Computing Model. As illustrated in Figure 1, we consider a mobile edge computing system, which consists of a mobile device U and n edge servers $\text{eNB} = \{\text{eNB}_1, \ldots, \text{eNB}_i, \ldots, \text{eNB}_n\}$. The mobile device U can be denoted by a two-tuple $U = \{f_u, N_u\}$, where f_u denotes the CPU frequency of the mobile device, and N_u denotes the number of CPU cores of the mobile device. Due to the limited computing resources and battery capacity of mobile device, the workflow applications (such as a video streaming-based face recognition application) running on mobile device can be scheduled to edge servers through wireless network. Each edge server can be denoted by a two-tuple $\text{eNB}_i = \langle f_{c,i}, N_{c,i} \rangle$, where $f_{c,i}$ denotes the CPU frequency of the ith edge server, and $N_{c,i}$ denotes the number of CPU cores of the i th edge server. Each edge server has an execution queue $Q_{c,i}$ that is used to store the tasks scheduled to the i th edge server.

Each mobile application can be abstracted into a workflow model, which can be denoted by a directed acyclic graph (DAG) $G = \langle V, E \rangle$, in which $V = \{v_1, \ldots, v_k, \ldots, v_K\}$ denotes a set of task nodes, and $E = \{e_{kl} | v_k \in V, v_l \in V\}$ denotes a set of edges between task nodes. Each task node v_k can be characterized by a three-tuple $v_k = \langle W_k, D_k^{tx}, D_k^{rx} \rangle$, in which W_k denotes the workload (CPU Cycles) of task node v_k, D_k^{tx} denotes the input data size (MB) of task node v_k, and D_k^{rx} denotes the output data size (MB) of task node v_k. The edge e_{kl} represents the precedence constraint between task nodes. This means that task v_l can be executed only after task v_k is executed. The system time is logically divided to equal length time slots, and the time slot duration is T_{slot}. The index sets of time slots can be denoted by $\mathcal{T} = \{0, 1, \ldots, \tau, \ldots\}$. At the beginning of each time slot, a task node in workflow is scheduled to the edge server.

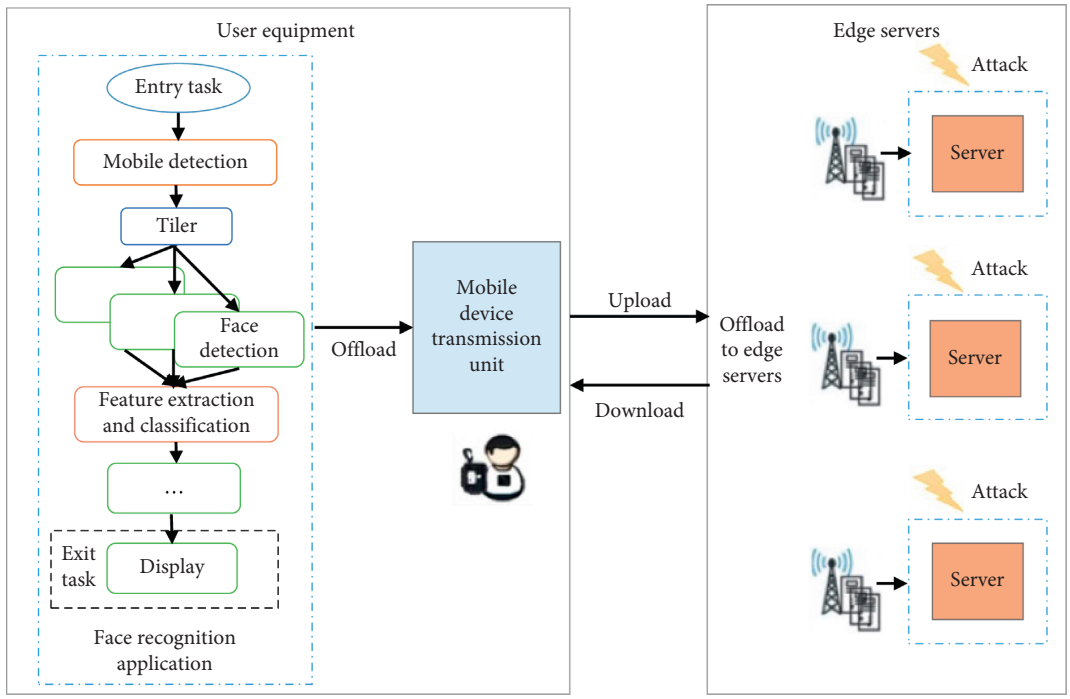

Figure 1: The architecture of workflow scheduling in a dynamic MEC with security threats.

3.2. Security Cost Model. The task nodes scheduled to edge servers are vulnerable to suffer from stealing and tampering security threats. In order to guard against these security threats, these task nodes need to employ encryption service cf and integrity service ig [36–38], respectively. Referring to the literature [38], encryption services cf mainly include IDEA, DES, Blowfish, AES, and RC4 algorithms. Each encryption algorithm has its own security level and encryption speed, which can be found in Table 1. The different encryption algorithms with different security levels can be flexibly selected to protect data from being stolen. Integrity services ig mainly include TIGER, RipeMD160, SHA-1, RipeMD128, and MD5 hash functions. Each hash function has its own security level and hash speed, which can be found in Table 2. The different hash algorithms with different security levels can be flexibly selected to protect data from being tampered. By flexibly selecting different encryption and hash algorithms with different security levels, an integrated security protection is formed to protect against security threats.

To ensure the security of task nodes scheduled to edge servers, the integrated security protection consisting of encryption and hash algorithms with different security levels needs to be employed. However, different security protection leads to different security cost. When the task node in the workflow is scheduled to the i th edge server, the total encryption cost on the mobile device can be calculated by [6]

$$T_{c,i}^{E} = \sum_{\text{type}\in\{cf,ig\}} \frac{\left(\varphi \cdot D_k^{tx}\right)}{f_u \cdot N_u \cdot \text{sp}\left(\text{sl}_{c,i}^{\text{type}}\right)}, \tag{1}$$

where $\varphi = 2.2$. When the task node is scheduled to the i th edge server, $\text{sl}_{c,i}^{cf}$ and $\text{sl}_{c,i}^{ig}$ denote the security levels of the encryption service and integrity service, respectively. $\text{sp}(\text{sl}_{c,i}^{cf})$ denotes the encryption speed of encryption service with encryption level $\text{sl}_{c,i}^{cf}$. $\text{sp}(\text{sl}_{c,i}^{ig})$ denotes the hash speed of integrity service with security level $\text{sl}_{c,i}^{ig}$. When the edge server eNB_i receives the task v_k, it first decrypts the task v_k and the total decryption cost can be calculated by

$$T_{c,i}^{\text{DE}} = \frac{f_u \cdot N_u \cdot T_{c,i}^{E}}{f_{c,i} \cdot N_{c,i}}. \tag{2}$$

3.3. Communication Model. Due to the user's mobility, the channel state between the mobile device and different edge servers is dynamically changing. We assume that the channel state between the mobile device and the edge servers is constant in each time slot τ and is dynamically changing in different time slots. In each time slot τ, the transmission rate $R_{c,i}^{u}(\tau)$ between the mobile device and the i th edge server can be calculated by

$$R_{c,i}^{u}(\tau) = B_{c,i}\log_2\left(1 + \frac{P_u G_{c,i}^{u}}{\sigma^2}\right), \tag{3}$$

where $B_{c,i}$ denotes the transmission bandwidth between the mobile device and the i th edge server, P_u denotes the transmission power of the mobile device, $G_{c,i}^{u}$ denotes the wireless channel gain between the mobile device and the i th edge server, and σ^2 denotes the Gaussian white noise power.

3.4. Risk Probability Model. To measure the risk degree of the task nodes scheduled to edge servers, it is necessary to establish a risk probability model to quantify the risk probability of these tasks.

Table 1: The encryption algorithms for confidential service.

Encryption algorithms	sl_{cf}: security level	$V(\mathrm{sl}_{cf})$: processing rate (Mb/s)
IDEA	1.0	11.76
DES	0.85	13.83
Blowfish	0.56	20.87
AES	0.53	22.03
RC4	0.32	37.17

Table 2: The hash functions for integrity service.

Hash functions	sl_{ig}: security level	$V(\mathrm{sl}_{ig})$: processing rate (Mb/s)
TIGER	1.0	75.76
RipeMD160	0.75	101.01
SHA-1	0.69	109.89
RipeMD128	0.63	119.05
MD5	0.44	172.41

Without loss of generality, referring to the literatures [36–38], the malicious attacks of data leakage and data tampering on the i th edge server are assumed to follow Poisson's distribution with parameters λ_i^{cf} and λ_i^{ig}. Therefore, the task node v_k in the workflow is scheduled to the edge server eNB_i, and the risk probability of data leakage or data tampering can be calculated by [6, 38]

$$P\left(\mathrm{sl}_{c,i}^{\mathrm{type}}\right) = 1 - \exp\left(-\lambda_i^{\mathrm{type}}\left(1 - \mathrm{sl}_{c,i}^{\mathrm{type}}\right)\right), \quad \mathrm{type} \in \{cf, ig\}. \tag{4}$$

Based on the above the description, when the task v_k in the workflow is scheduled to the edge server eNB_i, the risk probability of the task v_k suffering from these two malicious attacks can be calculated by

$$P(v_k) = 1 - \prod_{\mathrm{type} \in \{cf, ig\}} \left(1 - P\left(\mathrm{sl}_{c,i}^{\mathrm{type}}\right)\right). \tag{5}$$

When the risk probability of each task v_k scheduled to the edge server does not exceed $P_{\max}$, the risk probability of task execution must meet the following risk constraint:

$$P(v_k) \le P_{\max}. \tag{6}$$

3.5. Problem Formulation. In this section, we formulate the security-aware workflow scheduling problem in the mobile edge computing to be a Markov Decision Process. We first introduce the sorting strategy of workflow nodes and then define the state space, action space, and reward function of this problem. Finally, the objective function and constraints of this problem are defined.

3.5.1. Sorting of Workflow Nodes. In order to sort all the task nodes in the workflow, we assign a weight $\Pr(v_k)$ to each task node v_k [39]. The value of $\Pr(v_k)$ can be calculated by

$$\Pr(v_k) = \overline{\mathrm{ET}(v_k)} + \max_{v_l \in \mathrm{succ}(v_k)} \left\{ \frac{D_k^{rx}}{R_{kl}} + \Pr(v_l) \right\}, \tag{7}$$

where $\overline{\mathrm{ET}(v_k)}$ denotes the average time of the task node v_k executing on all edge services; R_{kl} denotes the transmission rate between edge servers, where the task node v_k and its successor node v_l are located; $\mathrm{succ}(v_k)$ denotes the set of all successor nodes of the task node v_k. Since the edge server each task node v_k is scheduled to is not known in advance, the priority of the task node can be calculated by the average time of the task node v_k executing on all edge servers. The priorities of all task nodes in workflow can be calculated by equation (7). According to the priorities of all task nodes, these task nodes can be sorted in descending order.

3.5.2. State Space. In each time slot τ, the sorted task nodes are scheduled in turn. The edge server each task node v_k is scheduled to is dependent on the system state. The system state $s(\tau)$ in time slot τ can be denoted by

$$s(\tau) = \left(W_c(\tau), Q_c(\tau), G_c^u(\tau)\right), \tag{8}$$

where $W_c(\tau) = (W_{c,1}(\tau), \ldots, W_{c,i}(\tau), \ldots, W_{c,n}(\tau))$ is an n-dimensional vector, denoting the workload states of n edge server; $Q_c(\tau) = \left\{Q_{c,1}(\tau), \ldots, Q_{c,i}(\tau), \ldots, Q_{c,n}(\tau)\right\}$ denotes the state of the scheduled tasks in n edge servers; $G_c^u(\tau) = \left\{G_{c,1}^u(\tau), \ldots, G_{c,i}^u(\tau), \ldots, G_{c,n}^u(\tau)\right\}$ denotes the channel state between the mobile device and n edge servers. Specifically, $W_{c,i}(\tau)$ denotes the workload of the edge server eNB_i in time slot τ; $Q_{c,i}(\tau)$ denotes a set of all task nodes scheduled to the edge server eNB_i in time slot τ; $G_{c,i}^u(\tau)$ denotes the channel state between the mobile device and the edge server eNB_i in time slot τ.

3.5.3. Action Space. In each time slot τ, the system action $a(\tau)$ can be denoted by

$$a(\tau) = \left(a_c(\tau), \mathrm{sl}_c^{cf}(\tau), \mathrm{sl}_c^{ig}(\tau)\right), \tag{9}$$

where $a(\tau) = (a_{c,1}(\tau), \ldots, a_{c,i}(\tau), \ldots, a_{c,n}(\tau))$ is a n-dimensional vector, denoting the edge server the current task node is scheduled to. Specifically, $a_{c,i}(\tau)$ denotes whether the current task node is scheduled to the edge server eNB_i. If the value of $a_{c,i}(\tau)$ is 1, it denotes that the current task node is scheduled to the edge server eNB_i; otherwise, it is the opposite. Note that, in each time slot τ, the current task node can only be scheduled to a single edge server. Therefore, the system action needs to meet the constraint condition $\sum_{i=1}^{n} a_{c,i}(\tau) = 1$. $\mathrm{sl}_c^{cf}(\tau) = (\mathrm{sl}_{c,1}^{cf}(\tau), \ldots, \mathrm{sl}_{c,i}^{cf}(\tau), \ldots, \mathrm{sl}_{c,n}^{cf}(\tau))$ denotes the security level of the encryption service employed by the task nodes scheduled to n edge servers. $\mathrm{sl}_{c,i}^{cf}(\tau) \in \{0, 1.0, 0.85, 0.56, 0.53, 0.32\}$ denotes the security level of the encryption service employed by task node scheduled to the i th edge server. $\mathrm{sl}_c^{ig}(\tau) = (\mathrm{sl}_{c,1}^{ig}(\tau), \ldots,$

$\mathrm{sl}_{c,i}^{ig}(\tau), \ldots, \mathrm{sl}_{c,n}^{ig}(\tau))$ denotes the security level of the integrity service employed by the task nodes scheduled to n edge servers. $\mathrm{sl}_{c,i}^{ig}(\tau) \in \{0, 1.0, 0.75, 0.69, 0.63, 0.44\}$ denotes the security level of the integrity service employed by the task node scheduled to the i th edge server.

3.5.4. Reward Function. In each time slot τ, given the system state $s(\tau)$, after taking an action $a(\tau)$, the immediate reward obtained by system is $R(\tau)$. The immediate reward $R(\tau)$ is defined as

$$R(\tau) = \begin{cases} -\left(\max T_{\mathrm{end}}(M(a(\tau))) - \max T_{\mathrm{end}}(M(a(\tau-1)))\right), & \text{if } \tau \neq 0, \\ -T_{\mathrm{end}}(M(a(0))), & \text{if } \tau = 0, \end{cases} \tag{10}$$

where $M(a(\tau)) = v_k$ denotes that, in time slot, the task node scheduled by taking the action $a(\tau)$ is v_k. $\max T_{\mathrm{end}}(M(a(\tau)))$ denotes the execution delay of the workflow until the τ th time slot, and $R(\tau)$ denotes the increment of the workflow execution delay after scheduling the task in time slice τ.

When the task node v_k is scheduled to the edge server eNB_i, the latest completion time $T_{\mathrm{end}}(v_k)$ is needed to be calculated. In order to calculate $T_{\mathrm{end}}(v_k)$, it is necessary to calculate the start time $T_{\mathrm{start}}(v_k)$ of the task node v_k, the encryption time $T_{c,i}^{E}$ of the task node v_k on the mobile device, the transmission time $T_{c,i}^{\mathrm{trans}}$ of the task node v_k transmitted from the mobile device to the edge server eNB_i, the waiting time $T_{c,i}^{\mathrm{wait}}$ of the task node v_k on the edge server eNB_i, the decryption time $T_{c,i}^{\mathrm{DE}}$ of the task node v_k on the edge server eNB_i, and the execution time $T_{c,i}^{\mathrm{exec}}$ of the task node v_k on the edge server eNB_i. In general, there may be multiple predecessor nodes for a task node v_k. Therefore, in order to calculate the start time $T_{\mathrm{start}}(v_k)$ of task node v_k, it needs to calculate the maximum sum of the completion time $T_{\mathrm{end}}(v_h)$ and the transmission time $T_{h,k}^{tr}$ for all the predecessor nodes v_h of the task node v_k. $T_{\mathrm{start}}(v_k)$ and $T_{\mathrm{end}}(v_k)$ can be calculated by equations (11) and (12), respectively:

$$T_{\mathrm{start}}(v_k) = \max_{v_h \in \mathrm{pre}(v_k)} \left\{ T_{\mathrm{end}}(v_h) + T_{h,k}^{tr} \right\}, \tag{11}$$

$$T_{\mathrm{end}}(v_k) = T_{\mathrm{start}}(v_k) + T_{c,i}^{E} + T_{c,i}^{\mathrm{trans}} + T_{c,i}^{\mathrm{wait}} + T_{c,i}^{\mathrm{DE}} + T_{c,i}^{\mathrm{exec}}, \tag{12}$$

where $\mathrm{pre}(v_k)$ denotes the set of all predecessor nodes of the task node v_k; v_h is a predecessor node of v_k. $T_{\mathrm{end}}(v_h)$ is the completion time of the task node v_h; $T_{h,k}^{tr}$ is the transmission time between the scheduled node v_k and its predecessor node v_h.

When the task nodes are scheduled to different edge servers, they will be exposed to different risk probabilities, thereby incurring different start time and different completion time. Therefore, this paper needs to find an optimal scheduling strategy π^* in a dynamic MEC with security threats, the main goal of which is to minimize the completion time of the workflow while satisfying the risk probability of the task nodes.

$$\text{Maximize: } R(\tau), \tag{13}$$

$$\text{Subject to: } P(v_k) \leq P_{\max}, \quad \forall_{v_k} \in V. \tag{14}$$

The objective of this paper can be denoted by equation (13). The risk probability constraint of the task node can be denoted by equation (14).

Due to the fact that the MEC environment is dynamical, and its state change is unknown (such as the gain state of the wireless channel), it is difficult for traditional optimization methods to solve the security-aware workflow scheduling problem in a dynamic MEC with security threats. However, the deep reinforcement learning algorithm, as a model-free machine learning approach, is good at solving such dynamic stochastic optimization problems. In the next section, the deep reinforcement learning-based security-aware workflow scheduling scheme is introduced in detail.

4. Deep Reinforcement Learning-Based Security-Aware Workflow Scheduling Scheme

The security-aware workflow scheduling problem in a dynamic MEC with security threats is formulated to be a finite Markov Decision Process. The action space of this problem is discrete. To solve the optimal workflow scheduling scheme, this paper proposes a SAWS scheme based on deep Q network (DQN).

As shown in Figure 2, the DQN framework consists of three main functional components: (1) the evaluated Q network: the evaluated Q network is consisting of one input layer, one hidden layer, and one output layer. The number of neurons in the input layer is equal to the number of dimensions of the state, the number of neurons in the hidden layer is taken as 2048 in this paper, and the number of neurons in the output layer is equal to the number of dimensions of the action. (2) The target Q network: the structure of the target Q network is the same as that of the evaluated Q network. To continuously approach the Q function, the parameters of the target Q network are periodically updated by the parameters of the evaluated Q network. (3) The replay memory: the function of replay memory is to store these state transition experiences $\langle s(\tau), a(\tau), R(s(\tau), a(\tau), s(\tau'+1)) \rangle$. A minibatch of state transition experiences are randomly chosen from the replay memory to train the Q network in the direction of minimizing a sequence of the loss function. The detailed processes of deep Q-network-based SAWS scheme are described in Algorithm 1.

During the training stage, the system state $s(\tau)$ in each time slot τ is first observed and fed into the evaluated Q network. Then, the evaluated Q network computes the evaluated Q values $Q(s(\tau), \cdot)$ for all possible actions $a(\tau)$ corresponding to the system state $s(\tau)$. The action with the largest Q value is chosen with $(1-\varepsilon)$ probability, and the action is chosen randomly with ε probability, and the immediate reward $R(s(\tau), a(\tau))$ can be calculated. Next, the system state $s'(\tau+1)$ in the next time slot $(\tau+1)$ can be

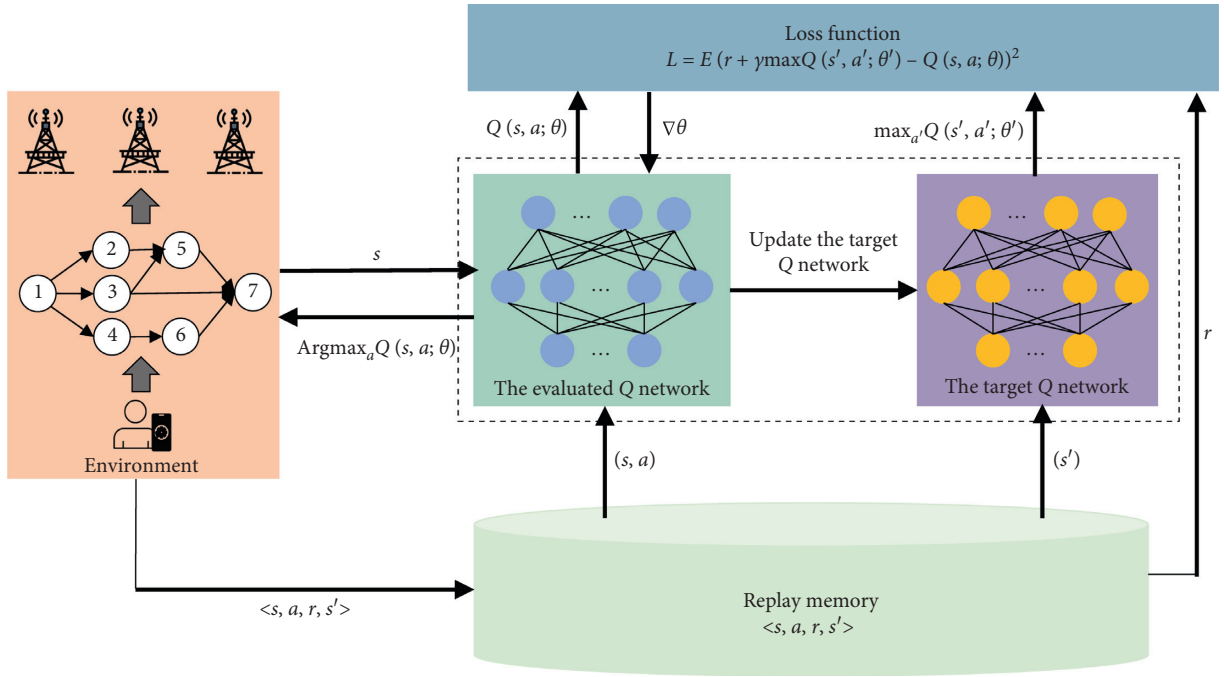

Figure 2: Deep Q-network-based security-aware workflow scheduling strategy.

BEGIN
(1) Initialize the replay memory with the size of buf_size, and a minibatch of the state transition experiences with the size of mini_batch;
(2) **for** episode = 1: MAX_EPI **do**
(3) Resetting the system state $s(\tau)$;
(4) for step = 1: K **do**
(5) At the beginning of each time slot τ, the current state $s(\tau)$ of the system is observed;
(6) Based on the current state $s(\tau)$, randomly select an action with ε probability and select the action $a(\tau)$ with the largest Q value with $(1 - \varepsilon)$ probability;
(7) The immediate reward $R(s(\tau), a(\tau))$ can be calculated and the system state $s'(\tau)$ in the next time slot $(\tau + 1)$ can be observed;
(8) The state transition experience $\langle s(\tau), a(\tau), R(s(\tau), a(\tau), s(\tau' + 1))\rangle$ can be obtained and stored into the replay memory;
(9) The immediate rewards $R(s(\tau), a(\tau))$ at each step are accumulatively summed;
(10) Randomly sample mini_batch state transition experiences from the replay memory to train the Q network;
(11) Calculate the expectation of the mean-squared error between the current evaluated Q value $Q(s(\tau), a(\tau); \theta_\tau)$ and the target Q value $R(s(\tau), a(\tau)) + \gamma \max_{a(\tau+1)} Q(s(\tau + 1), a(\tau + 1); \theta_{\tau+1})$:
(12) **end for**
(13) **end for**

Algorithm 1: Deep Q network-based security-aware workflow scheduling scheme.

observed, and the state transition experience $\langle s(\tau), a(\tau), R(s(\tau), a(\tau), s'(\tau + 1))\rangle$ can be obtained and stored into the replay memory with size buf_size. Finally, a minibatch of samples are randomly selected from the replay memory to train the Q network in the direction of minimizing the loss function $L(\theta_\tau)$ and the corresponding network parameters θ_τ are saved.

The loss function $L(\theta_\tau)$ is defined as the expectation of the mean-squared error between the current evaluated Q value $Q(s(\tau), a(\tau); \theta_\tau)$ and the target Q value $R(s(\tau), a(\tau)) + \gamma \max_{a(\tau+1)} Q(s(\tau + 1), a(\tau + 1); \theta_{\tau+1})$:

$$L(\theta_\tau) = E\left[\left(R(s(\tau), a(\tau)) + \gamma \max_{a(\tau+1)} Q(s(\tau + 1), a(\tau + 1); \theta_{\tau+1}) - Q(s(\tau), a(\tau); \theta_\tau)\right)^2\right]. \quad (15)$$

During the testing stage, the system state is first reset, and the learned network parameters are loaded. Then, at the beginning of each time slot, the current system state $s(\tau)$ is observed and fed into the trained neural network. Next, the neural network selects an optimal action $a(\tau)$ for the system state $s(\tau)$ and the corresponding reward is calculated.

5. Experimental Evaluation

To demonstrate the effectiveness of the proposed SAWS scheme in this paper, a lot of comparative experiments can be conducted. In this section, the simulation parameters are first set. Then, MSAWS, AWM, Greedy, and HEFT baseline algorithms are introduced. Finally, the performance of the SAWS scheme in comparison with these four baseline algorithms is analyzed under different simulation parameters.

5.1. Parameter Settings. This paper mainly considers a mobile edge computing system consisting of a mobile user U and n edge servers. Different workflow applications generated on the mobile device need to be scheduled in a dynamic MEC with security threats. Referring to the literatures [6, 7], the detailed parameter settings in experiment are introduced as follows:

(1) The parameter settings of the mobile device: the CPU frequency f_u and the CPU core number N_u of the mobile device are set to $f_u = 2.5\,\text{GHz}$ and $N_u = 4$, respectively.

(2) The parameter settings of edge servers: the number of edge servers is set to $n = 4$. The CPU frequencies of five edge servers are set to $f_{c,1} = \cdots = f_{c,5} = 2.5\,\text{GHz}$. The numbers of CPU cores are $N_{c,1} = 6$, $N_{c,2} = 7, N_{c,3} = 8$ and $N_{c,4} = 9$, respectively. The risk coefficients of confidentiality service for these five edge servers are $\lambda_{c,1}^{cf} = 1.8, \lambda_{c,2}^{cf} = 2.1, \lambda_{c,3}^{cf} = 2.4$ and $\lambda_{c,4}^{cf} = 2.7$, respectively. And the risk coefficients of integrity service for these five edge servers are $\lambda_{c,1}^{ig} = 1.2, \lambda_{c,2}^{ig} = 1.5, \lambda_{c,3}^{ig} = 1.8$ and $\lambda_{c,4}^{ig} = 2.1$, respectively.

(3) The communication parameter settings: the transmission power of each edge server is $P_{c,i} = 40\,\text{W}$, the maximum bandwidth is $B_{c,i} = 100\,\text{MHz}$, the white Gaussian noise power is $\sigma^2 = -174\,\text{dBm/Hz}$, the path loss constant is $\partial = 2$, the path loss exponent is $\theta = 4$, and the reference distance is $d_o = 1\,\text{m}$ [6, 7]. The distance between the mobile device and each edge server is $d_i \in (0, 350]\,\text{m}$.

(4) The parameter settings of workflow: the number of task nodes in different workflows is set to 50, 100, and 150, respectively. The out degree or in degree of each intermediate task node is less than 5, and every two task nodes can be connected with 10% probability to form an edge. The workload W_k of each task node v_k is in the range of $1\text{GHz} \cdot \text{s}$ to $10\,\text{GHz} \cdot \text{s}$. The input data size D_k^{tr} of each task node v_k is in the range of 10 MB to 100 MB, and its output data size D_k^{rx} is set from 1 MB to 10 MB. The maximum risk probability of each task node v_k is $P_{\max} = 0.4$.

(5) The parameter settings of the neural network: the evaluated Q network is consisting of one input layer, one hidden layer, and one output layer, and the number of neurons in the hidden layer is 2048. The learning rate is 0.003, and the learning discount factor γ is 0.9. The size buf_size of the replay memory is 3000, and the size mini_batch of the state transition experiences randomly sampled from the replay memory is 64. The maximum value of episodes is set to MAX_ EPI = 1000. The maximum value K of steps in each episode is equal to the number of task nodes in workflow.

5.2. Performance Analysis. To demonstrate the effectiveness of the proposed SAWS scheme, this paper implements MSAWS, AWM, Greedy, and HEFT baseline algorithms and compares the SAWS scheme with these four baseline algorithms under different experimental parameters.

Average Workload Minimization (AWM): In each time slot, the AWM strategy chooses the edge server with the smallest average workload to schedule the task node.

SAWS: This abbreviation represents a security-aware workflow scheduling scheme. Its main goal is to minimize the completion time of workflow while satisfying the risk probability constraint.

MSAWS: Based on the SAWS scheme, the security service with the security level 1 is chosen for these scheduled task nodes.

Greedy: In each time slot, the Greedy algorithm selects the edge server that enables each scheduled task node to complete at the earliest based on the current environment.

HEFT [40]: This abbreviation represents heterogeneous earliest finish time. This algorithm is a workflow scheduling strategy based on list and is widely used in workflow scheduling. It first needs to calculate the priority of task nodes based on their computational and communication costs. Then, the task node is scheduled to the server that can complete it at the earliest.

5.2.1. The Convergence Analysis of SAWS. Three different types of workflows with 50, 100, and 150 task nodes are scheduled by the SAWS scheme. Figure 3 shows their learning curves, respectively. It can be observed that the completion time gradually decreases and tends to be stabilized with the increasing of learning time (i.e., the number of Episodes). This result indicates that the proposed SAWS scheme can learn an optimal policy to schedule workflow applications with different task nodes. The optimal policy can minimize the completion time of workflow while satisfying risk probability constraint. Moreover, as shown in Figure 3, it can be further observed that the completion time

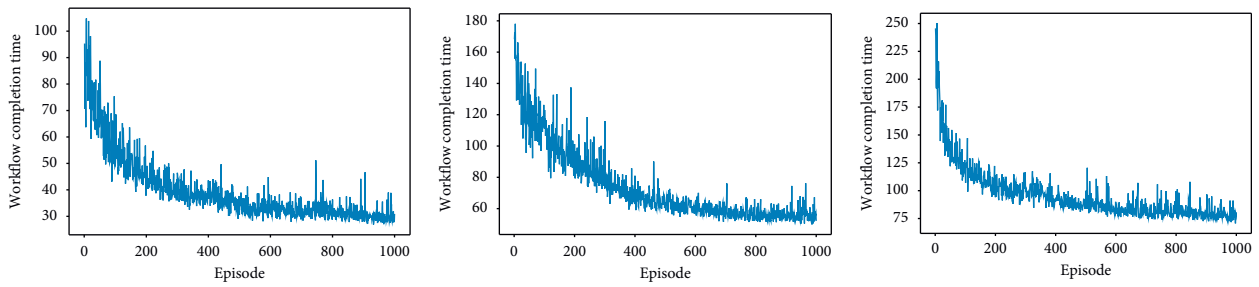

Figure 3: Learning curves with respect to different workflows. (a) The workflow with 50 nodes. (b) The workflow with 100 nodes. (c) The workflow with 150 nodes.

of workflow application with 50 task nodes is smallest, that of workflow application with 100 task nodes is medium, and that of workflow application with 150 task nodes is the largest. This is because the larger the scale of the workflow application, the larger the completion time.

5.2.2. The Impact of Different Risk Probabilities. To examine the impact of different risk probabilities on the completion times of different workflows, the risk probability is varied from 0.2 to 1.0 with the increment of 0.2 for workflows with 50, 100, and 150 task nodes, respectively. Figure 4 shows the completion time of the SAWS, MSAWS, AWM, Greedy, and HEFT algorithms under different risk probabilities for workflows with 50, 100, and 150 task nodes. As shown in Figure 4(a), the completion time of the SAWS algorithm is less than that of the MSAWS, AWM, Greedy, and HEFT algorithms. The main reason is that the SAWS algorithm can learn a security-aware workflow scheduling scheme in a dynamic MEC with security threats. This scheme can make an optimal scheduling decision according to different system states, thereby minimizing the completion time of the workflow while satisfying the risk probability constraint. The AWM algorithm selects the edge server with the least average workload to execute task node; hence, it is difficult to obtain an optimal solution. Although the Greedy and HEFT algorithms select the edge server that enables the task node to execute the task node at the earliest completion, it does not consider the after effect of task scheduling and is difficult to get an optimal solution. The MSAWS algorithm always selects the security service with the security level 1 to encrypt these scheduled task nodes. The MSAWS algorithm can effectively ensure the risk probability but significantly increases the completion time of workflow application. Moreover, we can observe that the completion time of five algorithms gradually decreases with the increase of the risk probability. It is because the greater the risk probability, the lower the security service level employed by task node to ensure its risk probability, and thereby the shorter the completion time of the workflow.

In addition, we can observe from Figure 4 that the completion time of workflow gradually decreases with the increase of the number of task nodes in workflow. The reason for this is the same as discussed in Section 5.2.1.

5.2.3. The Impact of Different Security Services. To evaluate the impact of different security services on the completion times of different workflows, only encryption service or only integrity service is employed by task nodes in different workflows. For simplicity's sake, only encryption service and only integrity service are denoted by *Confi_Only* and *Integ_Only*, respectively. Figure 5 shows that the completion time of *Confi_Only* and *Integ_Only* gradually decreases with the increase of the risk probability. It can be explained that the higher the risk probability, the lower the security level employed, the higher the processing rate of the security service, and thereby the shorter the completion time of the workflow. Moreover, it can be further observed that the completion time of *Integ_Only* is shorter than that of *Confi_Only*. This is because when the security level of the encryption service is approximately equal to that of the hash service, the processing rate of the hash service is higher than that of the encryption service. At last, it can be observed from Figure 5 that, with the increase of workflow nodes, the completion times of *Confi_Only* and *Integ_Only* gradually increase. The reason for this is the same as that discussed above.

5.2.4. The Impact of Different Risk Coefficients. Figure 6 shows the impact of different risk coefficients on the completion times of different workflows. We vary the risk coefficients of stealing and tampering security threats from 0.3 to 3, with the increment of 0.3. We can observe from Figure 6 that the completion time of *Confi_Only* and *Integ_Only* gradually increases with the increase of the risk coefficient. It is due to the fact that the task nodes are attacked more frequently with the increase of risk coefficient. In order to satisfy the risk probability constraint, the security service with a higher level is employed, which leads to longer task processing delay and the completion time of workflow. Moreover, we can observe from Figure 6 that the completion time of *Confi_Only* is higher than that of *Integ_Only*. The main reason is that when the security level of the encryption service is approximately equal to that of the hash service, the processing rate of the encryption service is lower than that of the hash service, which leads to a longer task processing delay and the completion time of workflow. Finally, we can see from Figure 6 that the completion time of *Confi_Only* and *Integ_Only* gradually increases with the increase of the number of the task nodes in workflow. The reason for this is the same as that discussed in Section 5.2.1.

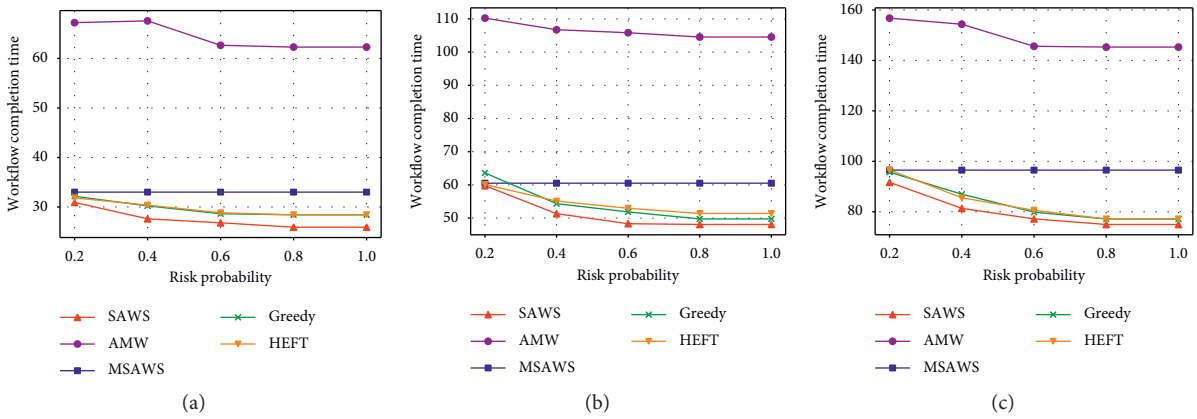

Figure 4: The impact of different risk probabilities. (a) The workflow with 50 nodes. (b) The workflow with 100 nodes. (c) The workflow with 150 nodes.

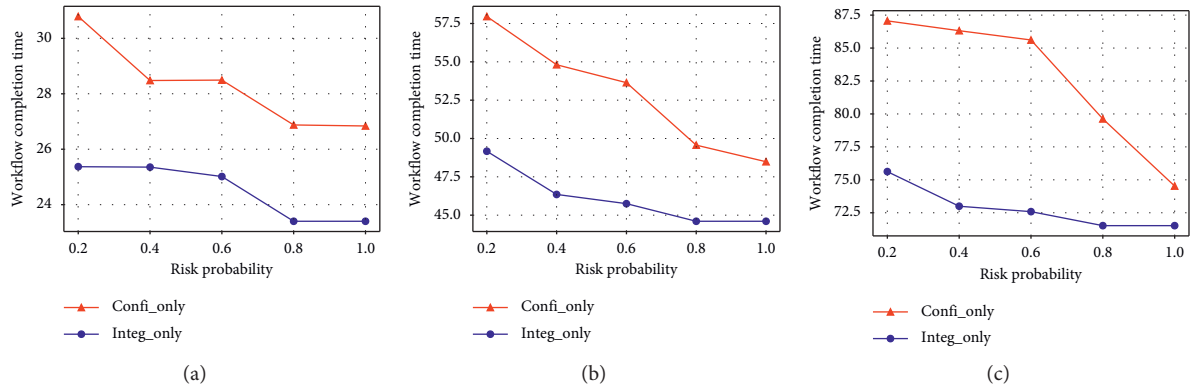

Figure 5: The impact of different security services. (a) The workflow with 50 nodes. (b) The workflow with 100 nodes. (c) The workflow with 150 nodes.

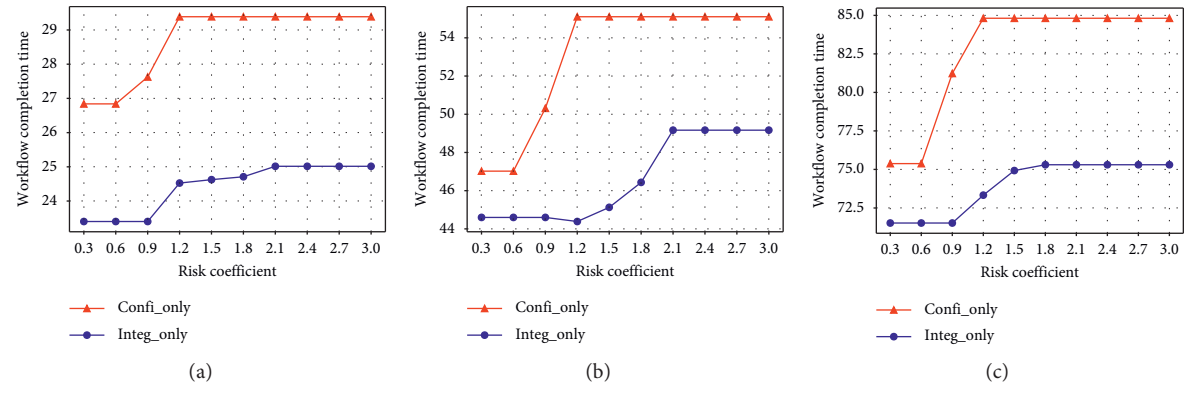

Figure 6: The performance of different risk coefficients. (a) The workflow with 50 nodes. (b) The workflow with 100 nodes. (c) The workflow with 150 nodes.

5.2.5. The Impact of Different Edge Server's Computing Capacities. Figure 7 shows the impact of different edge server's computing capabilities on the completion time of different workflows. As shown in Figure 7, we can see that the completion time of the SAWS, MSAWS, AWM, Greedy, and HEFT algorithms decreases with the increase of the number of the CPU cores. The main reason is that the more the CPU cores, the stronger the edge server's computing capacity, and thereby the shorter the task processing delay. Therefore, the completion time of workflow gradually decreases. In addition, we can further observe from Figure 7 that the SAWS algorithm performs better than the MSAWS, AWM, Greedy, and HEFT algorithms in terms of completion time of workflow. The reason for this is the same as that discussed in Section 5.2.2. Finally, we can observe that the completion time of the SAWS, MSAWS, AWM, Greedy, and

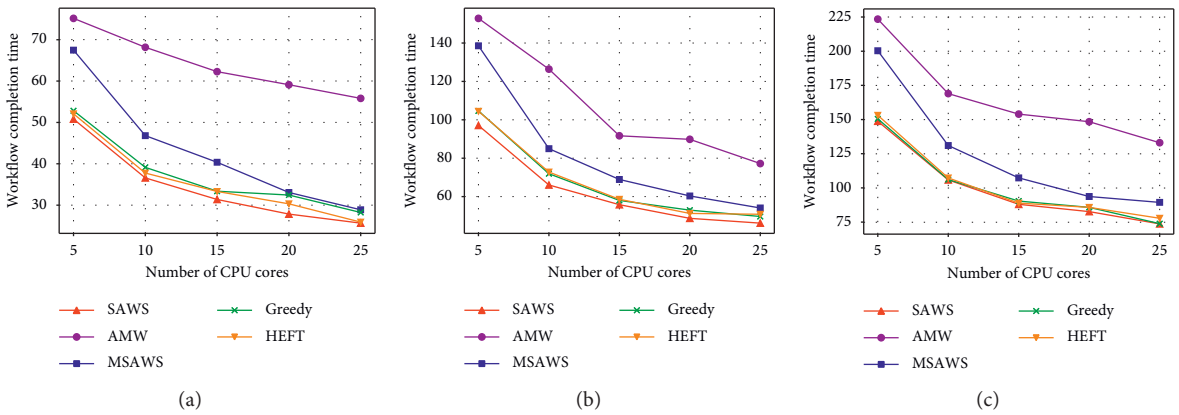

Figure 7: The impact of different edge server's computing capacities. (a) The workflow with 50 nodes. (b) The workflow with 100 nodes. (c) The workflow with 150 nodes.

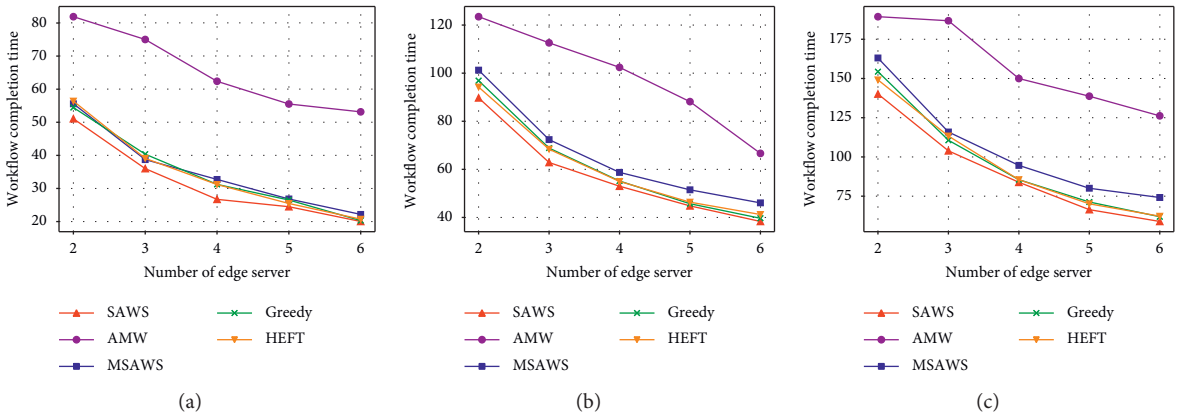

Figure 8: The impact of the number of edge servers. (a) The workflow with 50 nodes. (b) The workflow with 100 nodes. (c) The workflow with 150 nodes.

HEFT algorithms gradually increases with the increase of the number of task nodes in workflow. The reason for this is the same as that discussed in Section 5.2.1.

5.2.6. The Impact of the Number of Edge Servers. Figure 8 shows the impact of different number of edge servers on the completion time of different workflows with 50, 100, and 150 task nodes, respectively. To investigate the impact of different number of edge servers on performance, we vary the number of edge servers from 2 to 6 with the increment of 1. As shown in Figure 8, we can observe that the completion time of the SAWS, MSAWS, AWM, Greedy, and HEFT algorithms gradually decreases with the increase of the number of edge servers. It can be explained that the greater the number of edge servers, the stronger the computing capacity of the whole system, and thereby the shorter the completion time of workflow. Moreover, we can further observe that the completion time of the SAWS algorithm is lower than that of the MSAWS, AWM, Greedy, and HEFT algorithms. The reason for this is the same as that discussed in Section 5.2.5. At last, we can observe that, with the increase of task nodes in workflow, the completion times of the SAWS, MSAWS, AWM, Greedy, and HEFT algorithms gradually increase. The reason for this is the same as that discussed above.

6. Conclusions and Future Work

This paper proposes a reinforcement learning-based security-aware workflow scheduling (SAWS) scheme to solve the workflow scheduling problem in a dynamic MEC with security threats. This paper first constructs the mobile edge computing model, security cost model, communication model, and risk probability model, respectively. Then, this paper formulates the security-aware workflow scheduling problem to be a finite Markov Decision Process. To solve this problem, this paper adopts a deep Q network approach to learn an optimal security-aware workflow scheduling policy. The SAWS scheme enables minimization of the completion time of workflows while satisfying the risk probability. To verify the effectiveness of the SAWS scheme, this paper implements the MSAWS, AWM, Greedy, and HEFT baseline algorithms and compares the SAWS scheme with these four baseline algorithms under different experimental parameters such as the risk probability, the security service, the risk coefficient, the edge server's computing capacity, and the number of edge servers. The extensive experimental

results demonstrate the effectiveness of the proposed SAWS scheme.

Data Availability

The experiment data supporting this experiment analysis are from previously reported studies, which have been cited. The experiment data used to support the findings of this study are included within the article. The experiment data are described in Section 5 in detail.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Science Foundation of China (Nos. 62002316, 61802095, 61572162, and 61572251), the Zhejiang Provincial National Science Foundation of China (Nos. LQ19F020011 and LQ17F020003), the Zhejiang Provincial Key Science and Technology Project Foundation (No. 2018C01012), and the Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) (No. SKLNST-2019-2-15).

References

[1] C. Yi, J. Cai, and Z. Su, "A multi-user mobifle computation offloading and transmission scheduling mechanism for delay-sensitive applications," *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 29–43, 2020.

[2] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: task allocation and computational frequency scaling," *IEEE Transactions on Communication*, vol. 65, no. 8, pp. 3571–3584, 2017.

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[4] Z. Yuezhi and Z. Di, "Near-end cloud computing: opportunities and challenges in the post-cloud computing era," *Chinese Journal of Computers*, vol. 42, no. 4, pp. 677–700, 2019.

[5] C. Calero, "5Ws of green and sustainable software," *Tsinghua Science and Technology*, vol. 25, no. 3, pp. 401–414, 2020.

[6] B. Huang, "Security modeling and efficient computation offloading for service workflow in mobile edge computing," *Future Generation Computer System*, vol. 97, pp. 755–774, 2019.

[7] B. Huang, Y. Li, Z. Li et al., "Security and cost-aware computation offloading via deep reinforcement learning in mobile edge computing," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 3816237, 20 pages, 2019.

[8] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4642–4655, 2018.

[9] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," 2017, https://arxiv.org/abs/1711.09012.

[10] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: review and analysis of mobile edge computing and fog from a security and resilience perspective," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2586–2595, 2017.

[11] I. A. Elgendy, W. Zhang, Y. C. Tian, and K. Li, "Resource allocation and computation offloading with data security for mobile edge computing," *Future Generation Computing System*, vol. 100, pp. 531–541, 2019.

[12] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: a survey and analysis of security threats and challenges," *Future Generation Computing System*, vol. 78, pp. 680–698, 2018.

[13] T. Hongliang, Z. Yong, L. Chao, and X. Chunxiao, "Overview of research on database confidentiality protection technology in cloud environment," *Journal of Computers*, vol. 40, no. 10, pp. 2245–2270, 2017.

[14] O. Pedreira, F. Garcia, M. Piattini, A. Cortinas, and A. Cerdeira-Pena, "An architecture for software engineering gamification," *Tsinghua Science and Technology*, vol. 25, no. 6, pp. 776–797, 2020.

[15] M. Maimaiti, Y. Liu, H. Luan, and M. Sun, "Enriching the transfer learning with pre-trained lexicon embedding for low-resource neural machine translation," *Tsinghua Science and Technology*, vol. 40, p. 1, 2020.

[16] E. A. A. Alaoui, S. C. K. Tekouabou, S. Hartini, Z. Rustam, H. Silkan, and S. Agoujil, "Improvement in automated diagnosis of soft tissues tumors using machine learning," *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 33–46, 2021.

[17] Y. N. Malek, M. Najib, M. Bakhouya, and M. Essaaidi, "Multivariate deep learning approach for electric vehicle speed forecasting," *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 56–64, 2021.

[18] A. Guezzaz, Y. Asimi, M. Azrour, and A. Asimi, "Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection," *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 18–24, 2021.

[19] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[20] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "TOFFEE: task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Transactions on Cloud Computing*, vol. 10, p. 1, 2019.

[21] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 570–584, 2020.

[22] G. Chalapathi, V. Chamola, C. K. Tham, S. Gurunarayanan, and N. Ansari, "An optimal delay aware task assignment scheme for wireless SDN networked edge cloudlets," *Future Generation Computer Systems*, vol. 102, pp. 862–875, 2020.

[23] X. Xu, X. Zhang, X. Liu, J. Jiang, L. Qi, and M. Z. A. Bhuiyan, "Adaptive computation offloading with edge for 5G-envisioned internet of connected vehicles," *IEEE Transactions on Intelligent Transportation System*, vol. 99, pp. 1–10, 2020.

[24] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "EEDTO: an energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2163–2176, 2020.

[25] J. Xu, X. Li, R. Ding, and X. Liu, "Multi-resource computing offloading strategy for energy optimization in mobile edge computing," *Computer Integrated Manufacturing System*, vol. 25, no. 04, pp. 954–961, 2019.

[26] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel Distributed System*, vol. 30, no. 7, pp. 1464–1480, 2019.

[27] A. Zhu, S. Guo, M. Ma et al., "Computation offloading for workflow in mobile edge computing based on deep Q-learning," in *Proceedings of 2019 28th Wireless and Optical Communications Conference WOCC 2019*, July 2019.

[28] H. Liu, "Scheduling multi-workflows over edge computing resources with time-varying performance, a novel probability-mass function and ∂ DQN-based approach," in *Proceedings of International Conference on Web Services*, pp. 197–209, Beijing, China, October 2020.

[29] M. C. Sanchez, J. M. C. de Gea, J. L. Fernández-Alemán, J. Garcerán, and A. Toval, "Software vulnerabilities overview: a descriptive study allow," *Tsinghua Science and Technology*, vol. 25, no. 2, pp. 270–280, 2020.

[30] M. S. Mahmud, J. Z. Huang, S. Salloum, T. Z. Emara, and K. Sadatdiynov, "A survey of data partitioning and sampling methods to support big data analysis," *Big Data Mining and Analytics*, vol. 3, no. 2, pp. 85–101, 2020.

[31] X. Jia, D. He, N. Kumar, and K. K. R. Choo, "A provably secure and efficient identity-based anonymous authentication scheme for mobile edge computing," *IEEE Systems Journal*, vol. 14, no. 1, pp. 560–571, 2020.

[32] D. He, S. Chan, and M. Guizani, "Security in the internet of things supported by mobile edge computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 56–61, 2018.

[33] Y. Chen, Y. Zhang, S. Maharjan, M. Alam, and T. Wu, "Deep learning for secure mobile edge computing in cyber-physical transportation systems," *IEEE Networks*, vol. 33, no. 4, pp. 36–41, 2019.

[34] X. Xu, Q. Huang, H. Zhu et al., "Secure service offloading for internet of vehicles in SDN-enabled mobile edge computing," *IEEE Tranactions on Intelligent Transportation System*, vol. 10, pp. 1–10, 2020.

[35] X. Xu, Q. Huang, Y. Zhang, S. Li, L. Qi, and W. Dou, "An LSH-based offloading method for IoMT services in integrated cloud-edge environment," *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 16, no. 3, 2021.

[36] H. Chen, X. Zhu, D. Qiu, L. Liu, and Z. Du, "Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds," *IEEE Transactions on Parallel Distributed Systems*, vol. 28, no. 9, pp. 2674–2688, 2017.

[37] Y. Wu, J. Shi, K. Ni et al., "Secrecy-based delay-aware computation offloading via mobile edge computing for internet of things," *IEEE Internet Things Journal*, vol. 6, no. 3, pp. 4201–4213, 2019.

[38] Z. Li, J. Ge, C. Li et al., "Energy cost minimization with job security guarantee in Internet data center," *Future Generation Computing Systems*, vol. 73, pp. 63–78, 2017.

[39] Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai, "An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning," *The Journal of Supercomputing*, vol. 76, no. 1, pp. 455–480, 2020.

[40] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel Distributed System*, vol. 13, no. 3, pp. 260–274, 2002.