A cloud-edge service offloading method for the metaverse in smart manufacturing

Haolong Xiang[1] Xuyun Zhang[1] Muhammad Bilal[2]

1 Scool of Computing, Macquarie University, Sydney, New South Wales, Australia

2 Department of Computer and Electronic Engineering, Hankuk University of Foreign Studies, Yongin, South Korea

Summary

With the development of artificial intelligence, cloud-edge computing and vir-tual reality, the industrial design that originally depends on human imagination and computing power can be transitioned to metaverse applications in smart manufacturing, which offloads the services of metaverse to cloud and edge platforms for enhancing quality of service (QoS), considering inadequate com-puting power of terminal devices like industrial sensors and access points (APs). However, large overhead and privacy exposure occur during data transmis-sion to cloud, while edge computing devices (ECDs) are at risk of overloading with redundant service requests and difficult central control. To address these challenges, this paper proposes a minority game (MG) based cloud-edge ser-vice offloading method named COM for metaverse manufacturing. Technically, MG possesses a distribution mechanism that can minimize reliance on cen-tralized control, and gains its effectiveness in resource allocation. Besides, a dynamic control of cut-off value is supplemented on the basis of MG for bet-ter adaptability to network variations. Then, agents in COM (i.e., APs) leverage reinforcement learning (RL) to work on MG history, offloading decision, QoS mapping to state, action and reward, for further optimizing distributed offload-ing decision-making. Finally, COM is evaluated using a variety of real-world datasets of manufacturing. The results indicate that COM has 5.38% higher QoS and 8.58% higher privacy level comparing to benchmark method.

KEYWORDS

cloud-edge computing, metaverse, minority game, reinforcement learning, service offloading

1 | INTRODUCTION

Accelerated by the technologies of metaverse, the traditional manufacturing is upgrading productions and services from digital to virtualization.[1] For data-driven manufacturing services in metaverse, massive raw data collected are required to be virtualized and processed for reliable manufactural terminals, such as sensors, programmable logic controllers (PLCs) and smart machines.[2] Among the processing approaches, the flourishing virtual reality (VR), machine learning (ML) and deep learning (DL) have achieved state-of-the-art results in fields such as image classification and automated driving.[3,4] Data-driven analytics with VR and DL are increasingly supplementing downstream applications such as autonomous control for systems or machines.[5] However, the terminal devices on a manufacturing system or an access point (AP) are limited in computing capability. When users request additional services that exceed the computing and storage capabilities of the terminal devices, the quality of services (QoS) will not meet their satisfaction only with local computing. More severely, the on-board computing unit has a high risk of overloading, increasing the probabilities of system errors and mechanical failures. With the improvement of cloud computing, the high transmission rate and large bandwidth enable the service offloading for high-demanding data execution.[6]

While advantages are brought by service offloading, challenges are posed at the same time. Communication between terminal devices and server platforms incurs additional overhead as the compute task is no longer performed locally.[7–9] In today's cloud computing, the server platform, that is, the cloud data centers, are typically distant from terminal devices.[10] With such an architecture, the scale efficiency of cloud computing is improved, but higher transmission latency is generated. In addition, due to the increasingly large number of terminal devices, the cloud undergoes redundant service requests proposed on the terminal, which is exacerbated the bandwidth tension and connection instability. Moreover, users of cloud computing are required to upload their data to the cloud data center, which will increase the risk of privacy exposure due to the traversal of the public Internet.[11] Given these challenges, edge computing, as an innovative computing paradigm, provides end-users with an alternative.[12]

Technically, edge computing is not a substitute but a complement of cloud computing.[13] Rather than centralizing computation at the cloud, edge computing enables offloading tasks to the distributed edge computing devices (ECDs) in vicinity of terminal users.[14] In this way, communication distance is shortened and the workload are allocated across ECDs, mitigating the overloading and unbearable feedback time. To address privacy issues, edge computing enables data to be analyzed and refined close to the source. After being processed on the ECD, users' raw captured data will not be uploaded to the cloud. Instead, only the extracted data will be uploaded after desensitization and anonymization. In this way, the risk of users' private data leakage can be reduced.[15,16]

For practical scheduling, each ECD's computing capacity is assumed to be lighter than the cloud in the implementation of edge computing. This limitation is in consideration of the maintenance cost and energy expenditure of the multiply deployed ECDs. However, it also breeds the risk of ECD overloading and deteriorates QoS potentially. Thus, it is necessary to have an approach to utilize each ECD efficiently.[17] Previously, the management of ECDs is commonly achieved with centralized control of a macro controller. However, such a centralized approach involves large-scale computation and detailed service information consolidated at one point, inducing additional time and risk of privacy exposure.[18] Considering the dynamic network state and service configuration, such a centralized control of ECDs is not always practical. Instead of managing the ECDs in a centralized manner, with a view from the terminals, the distributed control takes advantage in less computationally complex, implemented in parallel, and less sensitive to private information.[19]

Specifically, the services of metaverse offloading requires onboard sensors, industrial machines, APs, and virtualization platforms to be connected in real-time. With industrial sensors and machines, data are captured and shared through sensor to sensor. Afterward, high-demanding service requests, along with the captured data, are collected by APs through sensor to infrastructure.[20] Upon receipt of offloaded service requests and their corresponding data, which often consist of videos and audios, server platforms process these requests using digital virtualization technology and then transmitted these virtualized data to the experts. Finally, the infrastructure sends back consolidated information, including task scheduler and operating instructions, to the sensors through the APs.[21] Since these platforms in edge cloud are equipped with higher computation power than terminal devices, the terminal devices can acquire feedback without long-time execution or high energy cost.[22] Moreover, machine operation security and the quality of services of metaverse can be guaranteed during the offloading process.[23,24]

To achieve the distributed control of service offloading, terminal devices make the offloading decisions rather than the macro controller. Thus, APs in our metaverse scenarios are modeled as the agents, and game theory is adopted to analyze their behavior and find the equilibrium that optimizes the collective utilization. Particularly, the minority game (MG) is in line with distributed control characteristics and is receiving awareness in wireless network management problems, for example, congestion problems and resource allocation, with its effective performance.[25] In this paper, a minority game based cloud-edge service offloading method (COM) with reinforcement learning (RL) is proposed for high-demanding services of metaverse in smart manufactoring. Specifically, the contributions of this paper are as follows:

1. We analyze the service offloading of metaverse in cloud-edge computing, where high-demanding data-driven services are deployed for digital virtualization and expert feedback scenarios, with consideration of privacy preservation as well as QoS. Then, we propose the challenges under metaverse service offloading from edge to cloud.
2. We adopt MG with dynamic control of cut-off value to model the service offloading in cloud-edge computing, and solve the offloading problem with RL from the perspective of terminal APs.
3. Extensive experiments are conducted between proposed method COM and comparative methods over manufactoring datasets, and the experimental outcomes prove the effectiveness and improvement of COM.

2 | RELATED WORK

With the advancement of artificial intelligence (AI), cognitive services based on ML and DL are gaining popularity and greatly facilitating people's lives. For instance, computer vision (CV) enables the automatic recognition and identification of objects, which is widely adopted in machine fault detection and is promising in metaverse of smart manufacturing.[20,26–28] As high QoS is pursued in such services, offloading them to powerful platforms is indispensable for terminal devices with insufficient storage and computation. With cloud-edge computing, service offloading offers storage extending, latency controlling, energy saving, and so forth.[29] Service offloading has a tight relationship with cloud and edge computing.[30] However, edge computing is expected to be more promising than cloud computing in improving QoS, as is elaborated by Shi et al.[31]

Edge computing has a broad future in future computing and communication architecture. Therefore, scholars have studied various prospects. Starting from the basic implementation, edge servers ought to be properly located. In Reference 32, Wang et al. studied the placement of edge servers and proposed a mixed integer programming (MIP) approach while considering load balancing and access delay. From another starting point, Xu et al.[33] devised a collaborative method for edge server quantification and placement, aiming at improving QoS as well as minimizing the number of deployed edge servers. With ECDs located and deployed, service offloading has its basic to be operational. In Reference 34, a contract optimization algorithm is proposed by Zhou et al. to achieve high-performance selection of servers and service offloading considering the unpredictability of fog. He et al.[35] proposed a Lyapunov-optimization-based service offloading scheme in mobile edge computing by considering the privacy of users and QoS of services. Relevant studies on the use of game theory in service offloading also exist. In Reference 36, the offloading scenario is formulated as a non-cooperative exact potential game (EPG) by Dinh et al. Then, they proposed a model-free reinforcement learning offloading mechanism to maximize the long-term utility. Li et al.[37] explored the optimization of computation offloading strategy using a non-cooperative approach to meet the demands of multiple, heterogeneous, and competitive mobile users. However, these methods fail to solve the privacy leakage on the service offloading between mobile devices and edge servers.

Privacy protection has long been a significant concern since the data-driven services, for example, facial and speech recognition, usually involve the transmission of personal information. In Reference 38, Zhao et al. presented an architecture of collaborative deep learning system with privacy protection which enables the users to build a deep learning model with data of participants in collaboration, and does not involve direct data storage and access in the central unit. Each participant in the proposed system will leverage the data of their own to train a local model, and shares the parameters in the model with the others. Osia et al.[39] proposed a hybrid approach for disassembling deep neural networks (DNN) for cooperative and privacy-preserving analytics. In the proposed approach, the DNN is not completely performed on the cloud. Instead, the initial layers of DNN are run on a terminal Internet of Things (IoT) device, and the output is sent to the cloud for further computation and analysis. Generally, as indicated in Reference 31, since the collected physical data usually contain much private information, processing the data at the edge take advantages in protecting privacy over directly uploading them to the cloud.

Game theory has a broad spectrum of branches and extensions, and has long been effective in various fields. Among them, minority game, introduced in Reference 40 was initially leveraged for market mechanisms, including market volatility and market predictions. In recent years, its satisfying performance has been discovered in congestion problems and resource allocation problems. In Reference 41, MG is adopted for task offloading in cloud-edge computing. Shen et al. modeled the offloading problem as a variant of MG and solved using modified connectionist Q-learning. However, the MG framework in that previous work is with a static cut-off value, thereby decreasing the flexibility and adaptability. In other topics related to edge computing and distributed systems, MG is proved to be effective. Ranadheera et al.[42] modeled the activation problem of ECDs as a minority game, where ECDs choose to be active or inactive in each round, to realize an energy-efficient activation of ECDs in edge computing. In Reference 43, Huang et al. proposed the uncertainty-aware

TABLE 1 Key terms and descriptions.

Terms	Descriptions
N	Number of APs
K	Number of ECDs
RS	Set of APs, $RS = r_1, r_2, \dots, r_N$
ES	Set of ECDs, $ES = e_1, e_2, \dots, e_k$
$D(t)$	Size of data collected by APs at time t, $D(t) = d_1(t), d_2(t), \dots, d_N(t)$
TR	Transmission range of network nodes
hop	Network hop count in a transmission route
T	Service time consumption
δ	Sensitive factor of data partition
P	Privacy measurement of data
C	ECD concurrency
ϕ	Cut-off value

minority game based energy management system (MG-EMS), intending to achieve higher utilization in terms of fairness in the distribution of solar energy resources.

However, large overhead and privacy exposure occur during data transmission to cloud, while ECDs are at risk of overloading with redundant service requests and difficult central control. It is still a big challenge for current methods to achieve high QoS quality with privacy protection under the above scenarios.

3 | SYSTEM MODEL

This section introduces the metaverse service offloading model in cloud-edge computing. Specifically, three models, namely the time-aware model of QoS, the privacy measurement model, and the workload management model, are designed to ensure QoS and resource allocation stability. Table 1 demonstrated the key terms and their descriptions.

3.1 | Metaverse service offloading framework in cloud-edge computing

In the metaverse senarios of smart manufactoring, terminal infrastructures named APs are deployed along buildings to collect service requests and raw data captured by sensors. The APs are denoted by the set $RS = r_1, r_2, \dots, r_N$. Considering the spatially uneven distribution of machines, denote $D(t) = d_1(t), d_2(t), \dots, d_N(t)$ as the datasize at time t. Since the AP is usually regarded as a basic communication node, the storage of models and large-scale computing tasks are not competent for it. Therefore, ECDs, denoted by the set $ES = e_1, e_2, \dots, e_k$, are colocated with APs. In the network, APs can communicate with each other in their transmission range to find a route to an ECD and a cloud access point. At the cloud data center, models are trained and distributed to ECDs. During the operation of service offloading, the destination ECD executes the service requests and extracts crucial information from the raw data. Afterward, the extracted information is sent to the cloud to update the model. As a whole, the service offloading framework is shown in Figure 1.

Denote TR_r as the fixed transmission range of APs and TR_e of ECDs, the network nodes (i.e., APs and ECDs) are modeled as:

$$r_j(lat_j, lon_j, d_j(t), TR_r), \quad j \in [1, N], \tag{1}$$

$$e_i(lat_i, lon_i, TR_e), \quad i \in [1, K], \tag{2}$$

where lat_n and lon_n are coordinates of the physical location. Then the distance between two nodes in Euclidean distance is calculated by:

$$dist(node_i, node_j) = \sqrt{(lat_i - lat_j)^2 + (lon_i - lon_j)^2}. \tag{3}$$

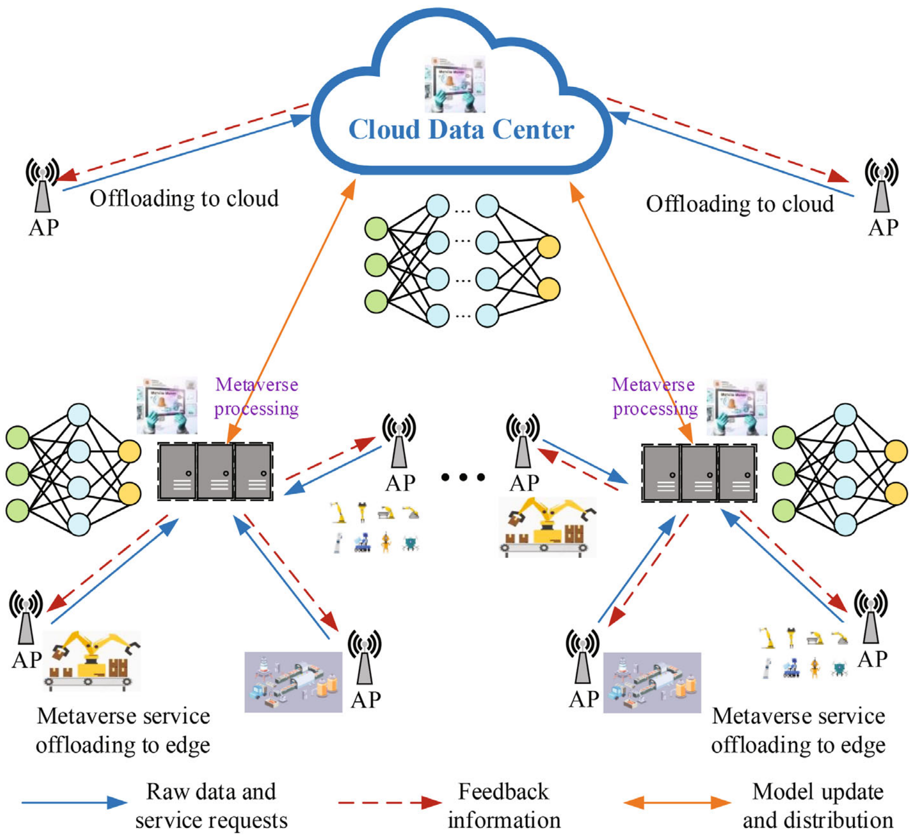

FIGURE 1 Metaverse service offloading framework in cloud-edge computing.

3.2 | Time-aware model of QoS

In the manufacturing system, APs can choose the offloading destination (i.e., the edge server or the cloud center) independently in each time slot. The service execution process can be divided into three subprocedures, that is, data transmission, data processing, and result feedback. Therefore, the service delay can be divided into the time of offloading, execution and feedback.

3.2.1 | Offloading Time

The offloading time is made up of four parts, that is, transmission latency, processing time, queuing delay, and propagation latency between network nodes. It is worth noting that processing time and queueing time can be omitted due to low processing and queuing delay caused by the moderate modification of vehicle data during transmission and the infrequent transmission burst resulted from the periodic traffic flow. Hence, only the transmission latency and propagation time are considered in service offloading. The offloading time of r_i to e_j, denoted by $T_{i,j}$, is calculated as:

$$T_{i,j}^{off} = \sum_{h=1}^{hop_{i,j}} \left[\frac{d_i(t)}{\lambda_{trans}} + \frac{dist(node_i, node_{next})}{\lambda_{prop}} \right], \tag{4}$$

where the transmission propagation rate is λ_{trans} and λ_{prop} respectively, and $hop_{i,j}$ is the network hop count from $node_i$ to $node_j$. In our model, the value of $hop_{i,j}$ is kept not larger than 3 when offloading to edge as ECDs are in close proximity to APs. However, offloading to the cloud usually has a hop count larger than 10.

3.2.2 | Execution time

The scale of computation and the computing capacity of the servers are analyzed below. The scale of computation depends on the amount of data as $c(d_i(t))$, and f is the function to evaluate the computational complexity of the service request. For simplicity, the server is treated as an aggregation of resources in the system model. Therefore, computing capacity and service requests are both measured by the resource unit number. Denote λ_{proc} as resource unit's processing rate, and μ_n as available resource units' quantity. Then, the execution time is:

$$T_i^{exe} = T_i^{que} + \frac{c(d_i(t))}{\mu_n \cdot \lambda_{proc}}, \tag{5}$$

where T_i^{que} is the queuing time of the service, denoted by the difference between the execution starting time and request receiving time as $T_i^{que} = T_i^{start} - T_i^{receiver}$.

3.2.3 | Feedback time

After the execution, the computing output are reported to the end-users. The output is with the size of $d_i'(t)$, much smaller than the input with the size of $d_i(t)$. Then, the feedback time is calculated as:

$$T(j,i)^{back} = \sum_{i=1}^{hop_{j,i}} \left[\frac{d_i'(t)}{\lambda_{trans}} + \frac{dist(node_j, node_{next})}{\lambda_{prop}} \right]. \tag{6}$$

As the transmission route of feedback information is opposite to the offloading route with the passing network nodes remain the same, the value of $hop_{j,i}'$ in feedback transmission follows $hop_{j,i}' = hop_{i,j}$.

3.2.4 | Time-aware QoS

With Equations (4)–(6) introduced above, the time-aware model of QoS is calculated by:

$$QoS = 1 - \frac{(T_{i,j}^{off} + T_i^{exe} + T_{j,i}^{back})}{T_{th}}, \tag{7}$$

where T_{th} is the tolerable time threshold of service, in our model, equivalent to the local execution time.

3.2.5 | Privacy measurement model

While offloading the services to cloud and edge platforms, the risk of privacy exposure exists in data transmission and storage. The following model is introduced to measure the privacy level.

During the data transmission, the risk of privacy leakage is influenced by the transmission time, network hop count, and data sensitivity. First, the transmitted data in offloading, that is, $d_i(t)$, is divided into $pt_i(t)$ partitions. For each partition pt, the proportion of it in the total data is denoted as ρ_{pt}, and the sensitive factor of it is denoted as δ_{pt}. Then, the data sensitivity of $d_i(t)$ is calculated by:

$$\begin{gathered} S(d_i(t)) = d_i(t) \sum_{pt=1}^{pt_i(t)} \rho_{pt}\delta_{pt}, \\ s.t. \sum_{pt=1}^{pt_i(t)} \rho_{pt} = 1, 0 < \delta_{pt} < 1. \end{gathered} \tag{8}$$

Within the offloading time $T_{i,j}^{off}$ and the feedback time $T_{j,i}^{back}$, each network node on the transmission route has a potential risk of privacy leakage. During execution time T_i^{exe}, the risk of privacy leakage is generated by illegal access. Then, the total privacy measurement is calculated by:

$$P_i = P_i^T + P_i^H = 1 - \frac{(T_{i,j}^{off} + T_i^{exe})s(d_i(t)) + T_{j,i}^{back}s(d_i'(t))}{T_{th}} + 1 - \frac{hop_{i,j}s(d_i(t)) + hop_{j,i}'s(d_i'(t))}{hop_{th}}, \tag{9}$$

where hop_{th} is the secure hop count threshold set as 5.

After the execution at a server, the feedback data $d_i'(t)$ has a relatively low sensitivity as the private information is mostly filtered out. Further, the model update and distribution involve data transmission between ECDs and the cloud. In this process, the sensitivity of data is considered negligible as the data are highly abstract.

3.3 | Workload management model

As the network condition is dynamic, the workload of ECDs needs to be appropriately managed. If an ECD is providing service for excessive customers, the QoS that provided drops significantly as it is overloaded.

Considering ECD $e_k \in E$ and the subset of APs within its coverage $RS_k \subset RS$, and the number of activated APs can be calculated as $N_k = |RS_k|$. Given the average service m_t requested to each AP, and average time consumption t_s, the concurrency expectation of an ECD is:

$$Ex[C] = \frac{N_k m_t t_s}{UT}, \tag{10}$$

where UT is the length of unit time. And the ideal number of accessed APs is calculated as:

$$N_k^{ideal} = Ex[C]f, \tag{11}$$

where f is a constant factor obeying $0 < f < 1$.

The workload upper limit is proportional to the concurrency expectation and the ideal number of accessed APs. Hence, the workload management of ECD is reflected through the ratio of APs requesting service (i.e., the cut-off value ϕ) at a specific time. Then ϕ is based on Equations (10) and (11) as:

$$\phi = \frac{N_k^{ideal}}{N_k} = \frac{m_t t_s}{UT} f. \tag{12}$$

4 | COM WITH MINORITY GAME FOR METAVERSE SERVICES

In this section, COM is designed for the metaverse service offloading. Specifically, this section begins with the MG scenario of COM. Then, the implementation of RL is introduced for agents' offloading decision-making in the MG.

4.1 | MG scenario of COM

Considering the scenario in which APs choose their offloading destinations autonomously, COM introduces the framework of multi-agent game, the MG. Besides the elemental mechanism, a variant of MG is adopted in the service offloading for better performance and adaptability.

4.1.1 | The original definition of MG

MG is a game model for complex competitive systems under limited resources. The original concept of MG is that, agents with an odd quantity N_k independently select a group from two to join. In this paper, the selection is either cloud computing or edge computing. In the process of MG, agents repeatedly compete for being a member of the minority (i.e., the group with a smaller size) as the winning group. Specifically, agents in the MG have knowledge of a certain length (i.e., the memory size m) of history winning moves and the cut-off value ϕ. Thus, with its own strategy determined and others strategies unknown, each agent independently makes decisions of the next move based on the history winning moves. After the decision-making of all agents, minority agents are considered winners and rewarded for the winning action, and then MG progresses to the next round.

For APs' offloading decision at the t-th round, $a_i(t) = 1$ indicates that r_i choose to offload to ECD e_k, while $a_i(t) = 0$ indicates that it leverages cloud computing. On such basis, the attendance is as $A_k(t) = \sum_{r_i \in RS_k} a_i(t)$. And when $A_k(t) = \phi N_k$, the game achieves the equilibrium in which ECD is ideally working. However, agents have limited information of the game, which requires them to dynamically switch actions. Hence, the attendance rather fluctuates slightly around ϕ than being fixed. Denote δ_A as the standard deviation to quantify the volatility of attendance. Then, δ_A is calculated by:

$$\delta_{A_k} = \left[\frac{1}{T} \sum_{t=1}^{T} (A_k(t) - \overline{A_k})^2 \right]^{\frac{1}{2}}. \tag{13}$$

where the mean value of attendance in e_k is denoted by $\overline{A_k}$.

Analytically, the mean value and volatility are indicators of the attendance and fluctuation. Meanwhile, the two different indicators cannot be analyzed distinctly. For instance, consider two groups of 101 agents who participated in five rounds of MG. The attendance of group 1 is $[50, 47, 54, 51, 48]$, and the attendance of group 2 is $[41, 49, 62, 55, 43]$. With calculations, we can observe that the average attendance of the two groups are the same as 50. Nevertheless, the volatility of the two groups are 2.45 and 7.74, respectively. For group 1, the mean value of winners is 48.4, while for group 2, 43.6. The instance depicts a stable and a volatile system, and the results exhibit the advantage of stability in MG.

4.1.2 | MG with arbitrary cut-off value

For elemental MG, the winning minority is with the size equivalent to or below $(N-1)/2$. However, the constraint of winning is not particularly as no more than half. In the archetype of MG, that is, the El-Farol bar, the bar's comfortable attendance is 60 with a total amount of 100 potential visitors. In other words, the winning minority consists has its size equivalent to or below 60% of total. In the same consideration, in COM, as well as other problems, the winning group can have its size specified according to the practical situation. Thus, a variant of MG with arbitrary cut-off value is leveraged. This variant of MG is more practical than elemental MG, where the cut-off values in most cases are generally other than 50%, but a value more suitable for the problem.

4.2 | RL-based decision-making in MG

In COM, APs are modeled as the agents of MG. Each agent has a state set (S) and a virtual score table (Q). Each state of S represents a possible history with the length of m. And table Q records the quality (cumulated payoff) of each state. After each round of MG, there is a winning move:

$$w(t) = \begin{cases} 1, & A(t), \phi N', \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

and the winning moves are stored in history μ. With the definition of winning move and history, the size of S is determined as 2^m, and the size of Q is $2^m \times 2$. The distributed learning scheme of agents is shown in Algorithm 1.

In the original mechanism of MG, the update of cumulated reward is simplified as:

$$Q^{r_i}_{s,a}(t+1) \leftarrow Q^{r_i}_{s,a}(t) + 1 - a_i(t) \oplus w_i(t). \tag{15}$$

where $\oplus$ is the exclusive or operator. However, various learning algorithms can be leveraged by agents.

Algorithm 1. Distributed learning of APs within the coverage of ECD e_k in MG

```
Input: Round of MG $T$, subset of APs $RS_K \subset RS$, set of states $S$.
Output: Came history $\mu$.
1:  Initialize $\mu$, current state $S$, and each $Q^{ri}$ with 0.
2:  for $t = 1$ to $T$ do
3:      for each $r_i \in RS_k$ do
4:          Offloading decision-making of $r_i$.
5:          $A(t) \leftarrow A(t) + a_i(t)$.
6:      end for
7:      Obtain winning move $w(t)$ based on Equation (14).
8:      Append $w(t)$ to $\mu$.
9:      for $r_i \in RS_k$ do
10:         if $a_i(t) = w(t)$ then
11:             Update $Q^{r_i}_{s,a}(t+1)$ as cumulated reward.
12:         else
13:             $Q^{r_i}_{s,a}(t+1) \leftarrow Q^{r_i}_{s,a}(t)$
14:         end if
15:     end for
16:     Update $s$ as the binary value of last $m$ bits of $\mu$.
17: end forreturn $\mu$.
```

Commonly, reinforcement learning is a practical choice.[44] In RL, agents consider the balance between exploring new strategies and exploiting most beneficial strategies to realize decision optimization.[45]

A well-known RL algorithm named Q-learning enables an agent to search for the maximum long-term reward, but the off-policy feature of Q-learning makes it unsuitable for COM. However, an on-policy variant of Q-learning named Sarsa (acronym of state, action, and reward) is leveraged in COM.[5] Thus, the fundamental method of updating cumulated reward can be replaced with:

$$Q^{r_i}_{s,a}(t+1) \leftarrow Q^{r_i}_{s,a}(t) + \alpha\left[R + \gamma Q^{r_i}_{s',a'}(t) - Q^{r_i}_{s,a}(t)\right]. \tag{16}$$

where α and γ are two parameters of the learning rate and discount factor. R is calculated after a round of game as the QoS in Equation (7) to represent the reward of action α.

To avoid falling into the local optimum, agents are not necessarily to choose the best action all the time. Instead, exploration is encouraged to limit the over-exploitation. On such basis, the ó-greedy strategy is introduced during offloading decision-making, which allows a certain degree of randomness. With the probability represented as $Pr[a_i(t) = pi_{best}] = 1-$ ó, the agent chooses to exploit its knowledge by selecting the highest Q-value. With a probability of ó, it randomly selects another action to explore for more available choices. In COM, as there are two offloading decisions, the agent choose the action with lower score with a probability of ó. Normally, ó is decreasing with rounds of MG, facilitating early exploration and limiting blindness and variability after obtaining enough knowledge. The RL-based decision-making scheme is shown in Algorithm 2.

4.3 | Dynamic control of cut-off value

As the cut-off value of MG can be customized, it is essential to dynamically adjust the value to meet the diversity of the network environment. After each round of the game, the cut-off value ϕ and average QoS of each AP are feedback to the ECD. During a period of RN rounds, the cut-off value ϕ remains unchanged. Then, the ECD will adjust ϕ based on the performance and utilization of ECD computational resources during the last period.

After adjusting ϕ, the ECD e_k will broadcast a signal to all the APs within its range to inform them of the change. Simultaneously, the APs in RS_k will increase the value of ó for a quicker exploration. The dynamic control of ϕ is shown in Algorithm 3.

Algorithm 2. RL-based offloading decision-making for AP r_i

```
Input: Virtual score $Q^{r_i}_{s,a}(t)$ of $r_i$, $\epsilon$.
Output: Action $a_i(t)$ of $r_i$ at $t$.
1: $a_{best} \leftarrow \underset{a}{\mathrm{argmax}} Q^{r_i}_{s,a}(t)$.
2: $Pr[a_i(t) = a_{best}] \leftarrow 1 - \epsilon$.
3: $P_{rand} \leftarrow random(0, 1)$.
4: if $P_{rand} \leq Pr[a_i(t) = a_{best}]$ then
5:     $r_i$ picks the best offloading decision $a_i(t) \leftarrow a_{best}$.
6: else
7:     $r_i$ picks another offloading decision $a_i(t) \leftarrow \neg a_{best}$.
8: end if
9: $r_i$ feedbacks $a_i(t)$ to $e_k$.return $a_i(t)$.
```

Algorithm 3. Dynamic control of cut-off value ϕ of ECD e_k

```
Input: Current cut-off value $\phi$ of ECD $e_k$, AP subset $RS_K \subset RS$.
Output: Next round cut-off value $\phi'$ of ECD $e_k$.
1: for each $r_i \in RS_k$ do
2:     Calculate $QoS_i$ accoding to Equation (7) and feedback to ECD $e_k$.
3: end for
4: $\bar{QoS} \leftarrow \frac{1}{|RS_k|} \sum_{r_i \in RS_k} QoS_i$.
5: if $\bar{QoS} < QoS_{th}$ then
6:     if $u_{idle} < u_{th}$ then
7:         $\phi' \leftarrow \min(\phi - \frac{u_{th} - u_{idle}}{u_k - u_{idle}} \phi, 1)$.
8:     else
9:         $\phi' \leftarrow \max(\phi + \frac{u_{idle} - u_{th}}{u_k - u_{idle}} \phi, 0)$.
10:    end if
11:    Broadcast the change of $\phi'$ to all APs in $RS_K$.
12: end ifreturn $\phi'$.
```

4.4 | COM overview

In general, COM is devised in the following logic flow shown in Figure 2. In COM, service offloading is formalized as MG with arbitrary cut-off value and dynamic control of cut-off value. APs observe the history winning moves and leverages an RL-based method to make offloading decisions individually. ECDs in the offloading system focus on the execution of service requests and the cut-off value adjustment. Specifically, the pseudo code of COM is shown in Algorithm 4. Comparing with centralized offloading control, COM avoids the drawbacks such as extensive administrative overhead and detailed information requirements. Therefore, COM can achieve a higher level of QoS as well as privacy in IoV service offloading.

In Figure 3, an example of COM is depicted. With an ECD and 12 APs assigned to it, the system is initialized with a cut-off value $\phi = 0.6$, then, ϕN is set as its round down value as $\lfloor 0.6 \times 12 \rfloor = 7$. At the first stage, 6 APs choose to offloading to ECD while the others choose to offloading to the cloud. Thus, the actual attendance is $A = 6$, which is smaller than ϕN. In the next round, more APs tend to offload to ECD, and the actual attendance is 7, equals to ϕN. In such a circumstance, the MG is in the optimal state as shown in stage ii. However, as more APs are exploring for better offloading decisions, the MG might transit to a unoptimized state. In state iii, there are nine APs offloading to ECD while ϕN is 9. In view of this, COM enables two methods for the system to regain stableness. One is that APs adjust their offloading decision-making in the following rounds subject to the rule of MG. However, the adjustments of APs are with limits. When ϕ is low, the resources of ECD cannot be fully utilized, while the ECD is likely to be overloaded when ϕ is high and the service requests are excessive. Concerning the limitation, the other methods provided by COM is that ECD adjusts the cut-off

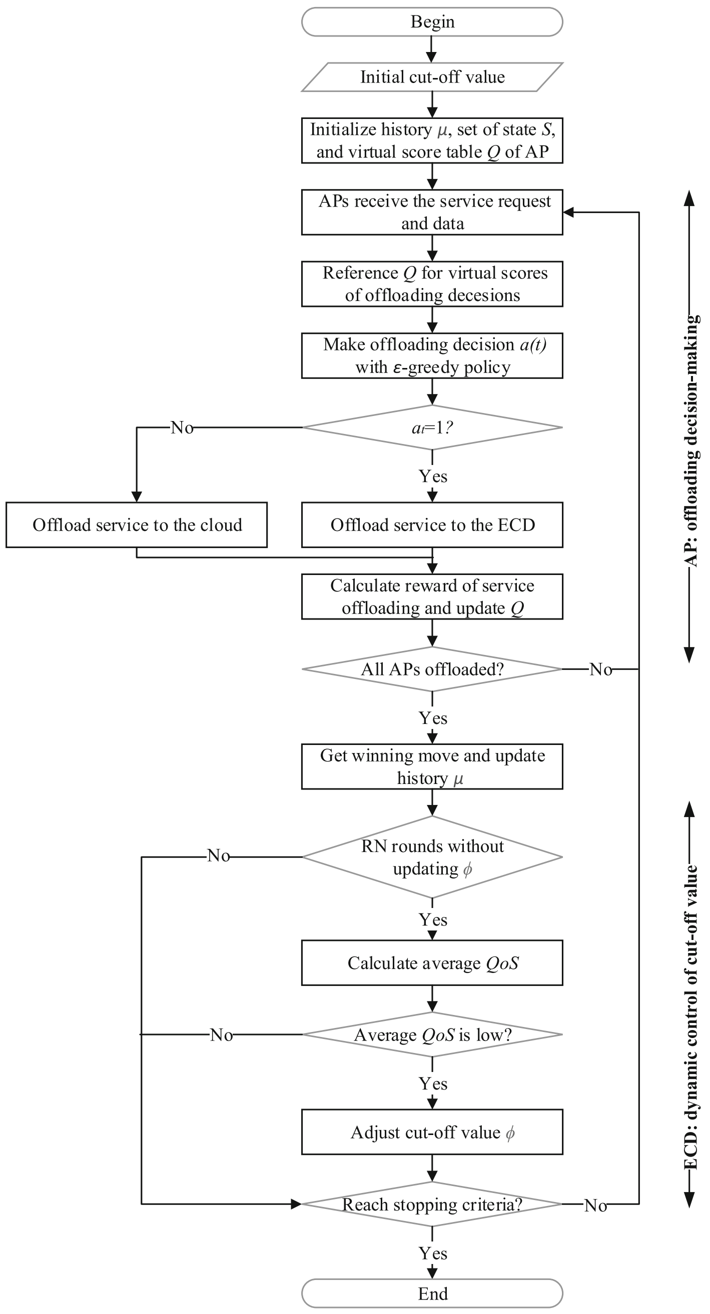
Begin
Initial cut-off value
Initialize history μ, set of state S, and virtual score table Q of AP
APs receive the service request and data
Reference Q for virtual scores of offloading decesions
Make offloading decision $a(t)$ with ε-greedy policy
a_t=1?
No
Yes
Offload service to the cloud
Offload service to the ECD
Calculate reward of service offloading and update Q
All APs offloaded?
No
Yes
Get winning move and update history μ
RN rounds without updating ϕ
No
Yes
Calculate average QoS
Average QoS is low?
No
Yes
Adjust cut-off value ϕ
Reach stopping criteria?
No
Yes
End
AP: offloading decision-making
ECD: dynamic control of cut-off value

FIGURE 2 The programming flowchart of COM.

value ϕ according to the performance for a higher overall QoS. As depicted in state iv, the ECD update ϕ to 0.8 for a higher utilization rate. Commonly, ϕ is supposed to rise during off hours where service requests by each AP are lightweight, and decline during peak hours where service requests are excessive.

5 | EXPERIMENTAL EVALUATION

In this section, experiments on COM are conducted with real-world manufacturing datasets. Then, results of COM with other service offloading strategies are exhibited and compared. Finally, the availability and extensibility of COM are validated with experiment outcome.

Algorithm 4. Overall process of COM

```
Input: Initial cut-off value $\phi$.
Output: Result of APs' service offloading.
1:  Initialize history, state, and virtual score.
2:  for $t$ in total rounds do
3:      for $e_k$ in ES do
4:          Offloading decision-making of APs in $ES_K$.
5:          MG results and QoS calculation.
6:          History update of MG.
7:          Virtual score update of APs.
8:          if $Q\bar{o}S$ is low then
9:              Cut-off value $\phi$ update of $e_k$.
10:             Broadcast of new $\phi$ to APs.
11:         end if
12:     end for
13:     Store $Q\bar{o}S$ and $\phi$ in result $res$.
14: end forreturn $res$.
```

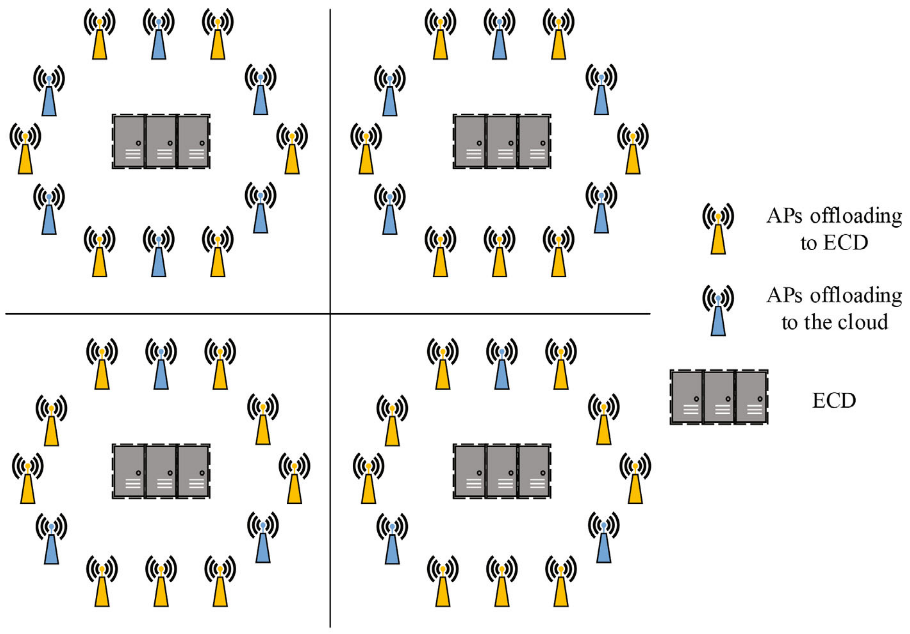

FIGURE 3 The programming flowchart of COM.

5.1 | Experiment setup

All the simulations are conducted on a PC Server of Manjaro Linux 21.1.0 with Intel i5-10600KF (4.10GHz), 64GB RAM, and 2 NVIDIA RTX3090 GPUs. Python 3.9.5 with Pytorch 1.8.1 is employed as the experimental environment.

The experiments leveraged two manufacturing datasets. One consists of 436 APs' detail in an intelligent industrial park, including their location and activation information. Based on the AP locations, a collaborative method of edge server quantification and placement in Reference 33 is adopted for a simulated ECD deployment. Finally, 385 activated APs and 42 ECDs are used in the experiments.

Over 160 million service requests, which each AP collects from 00:00:00 September 1 to 23:59:59 September 29 in 2014, are included in the other dataset. For analysis purposes, the service requests are segmented into 1800 different offloading rounds during each workday.

5.2 | Comparative decision-making strategies

5.2.1 | Random

As an unoptimized decision-making strategy, random selection is often used for benchmarking. With all the APs' offloading decisions selected in random, the attendance obeys binomial distribution. Then, the expectation and the volatility are as:

$$Ex[A] = Np, \quad \delta_A = \sqrt{Np(1-p)}, \tag{17}$$

where p is a parameter representing the probability of AP applying edge computing. Let $N = 385$ and $p = \phi = 0.8$, random selection obeys the statistics as:

$$Ex[A] = N\phi = 308, \quad \delta_A = \sqrt{N\phi(1-\phi)} = 7.85. \tag{18}$$

5.2.2 | Win-stay-lose-shift

As a stochastic strategy-based learning method, WSLS is with a straightforward mechanism. As its name indicated, the agent is more likely to maintain the previous decision after winning a round of the game. In contrast, the offloading decision of the agent is switched with a probability if it loses in the previous round.[46]

5.2.3 | Static cut-off value approaches

In previous work,[41] the cut-off value was assumed to be static within an experiment, and it achieved a relatively stable performance with different cut-off values. To analyze the overall performance, static cut-off value methods are used as a comparison. Specifically, we use the suffix SC (static cut-off) and DC (dynamic cut-off) to distinguish between the previous work and the current work.

5.3 | Evaluation on the stability of COM

In this section, COM is implemented in two ways, namely COM with static cut-off value (COM-SC) and COM with dynamic cut-off value (COM-DC), to measure its stability. The major indicators are mean value and volatility of attendance, and the mean square error (MSE) with MG optimal (i.e., $\sum \phi_k N_k$).

5.3.1 | COM with static cut-off value

Figure 4 elaborates the attendance of edge computing within 1800 rounds of service offloading. To evaluate the performance, the 1800 rounds are partitioned into 36 periods, each with a length of 50 rounds. After the first period, the attendance tends to converge at ϕ. For a more rigorous evaluation, statistics of the overall performance and the converged performance are listed in Table 2. In the later phase, agents mainly exploit the knowledge they have learned, and the attendance of edge computing reveals a stable feature.

Experiment outcome in Table 2 infers that the mean value of attendance obeys WSLS < COM-SC < Random. Nevertheless, such a phenomenon cannot imply random selection's better performance in resource allocation. Instead, jointly investigating the volatility which obeys COM-SC < WSLS < Random and the MSE which obeys COM-SC < Random < WSLS, COM provides a more stable performance. It is noting that WSLS reveals a worse-than-random performance in MSE but also has a volatility smaller than random. The rationale is that, APs with WSLS tend to avoid ECD overloading, and the attendance is thus fluctuating below the MG optimal, which brings a larger MSE than random choice.

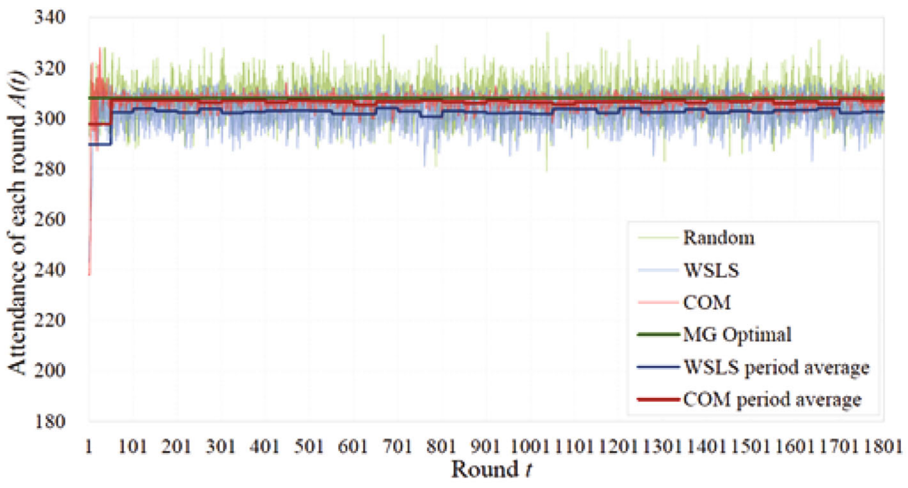

FIGURE 4 Attendance of edge computing with static cut-off value.

TABLE 2 Numerical results of edge computing attendance with static cut-off value.

		Random	WSLS-SC	COM-SC
Average	Overall	308.16	302.48	306.49
	Converged	308.16	302.71	306.57
Volatility	Overall	7.74	7.08	6.37
	Converged	7.73	4.12	2.44
MSE	Overall	59.91	80.72	19.22
	Converged	59.78	66.73	7.77

5.3.2 | COM with dynamic cut-off value

When the cut-off value ϕ is not fixed, the service offloading can be more effective. In this experiment, the initial cut-off value is set as $\phi = 0.6$. Similar to the previous experiment, the attendance of edge computing is shown in Figure 5, and the statistics are listed in Table 3.

With dynamic cut-off value, the average attendance and volatility are not major but supplemental indicators. Instead, we pay more attention to the periodic performance and the MSE. As exhibited in Figure 5, COM-DC has its periodic average value of attendance close to MG optimal. Meanwhile, as can be observed in Table 3, the MSE obeys COM-DC < WSLS < Random. Specifically, COM-DC has 73.3% lower MSE than WSLS, which indicates the effectiveness and adaptability of COM-DC. However, comparing with COM-SC, the MSE of COM-DC is larger. The rationale is that, when the cut-off value is adjusted, APs in the system need additional rounds to explore and adapt to the new environment.

Generally, ECD overloading often take place in random method as the attendance frequently exceeds ϕ. As for WSLS-based method, it exhibits a more stable result than random and a simpler mechanism than COM, but its optimization capability is relatively limited compared with RL-based decision-making of COM. In general, random decisions often lead to ECD overload, because the probability of attendance exceeding the cut-off value is high. The decision-making mechanism based on WSLS is simpler than COM, and the result is more stable than random selection, but the optimization performance is limited. COM decision-making based on RL has high optimization ability and its mechanism is more complex than random selection and WSLS.

5.4 | Evaluation on the effectiveness of COM

In this section, the effectiveness of COM is evaluated with regard to QoS and privacy level. During the experiments, the 1800 offloading rounds are partitioned into six periods, each with a length of 300 rounds, for evaluation.

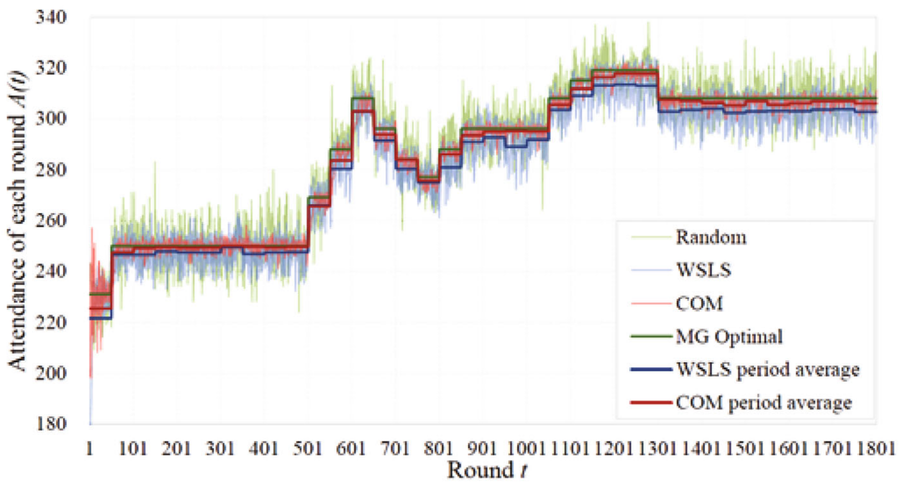

FIGURE 5 Attendance of edge computing with dynamic cut-off value.

TABLE 3 Numerical results of edge computing attendance with dynamic cut-off value.

		Random	WSLS-DC	COM-DC
Average	Overall	287.32	282.63	285.42
	Converged	288.98	284.28	287.03
Volatility	Overall	28.19	26.58	26.44
	Converged	26.76	24.94	24.95
MSE	Overall	71.22	69.68	18.63
	Converged	71.29	62.09	14.29

5.4.1 | Analysis on QoS level

From Figure 6, COM outperforms WSLS with both static and dynamic cut-off value. Overall, COM-DC has 5.38% higher QoS than WSLS-DC, while COM-SC has 5.53% higher QoS than WSLS-SC. During the first two periods, the cut-off value of COM-DC and WSLS-DC is kept low, while the static cut-off value is fixed at $\phi = 0.8$. Benefiting from this, the SC methods exhibit a higher QoS with more edge computing attendances. However, During the third period (i.e., round 601 to 900), the rush hours involve excessive service requests, and the ECDs are more likely to be overloaded with a high cut-off value. When the overall QoS is low, DC methods reduce the cut-off value and offload more services to the cloud to prevent ECDs from continuously overloading.

5.4.2 | Analysis on privacy level

The average privacy level is shown in Figure 7. The relatively low starting point of cut-off value in DC methods makes the performance temporarily worse than SC methods. Similar to the QoS analysis, COM-DC outperforms the other methods during rush hours in privacy level. COM-DC and COM-SC has 8.58% and 7.84% higher privacy level than WSLS respectively. A noticeable difference is that, in the privacy measurement, SC methods exhibit a greater advantage over DC methods. The rationale is that offloading to the cloud involves a significant amount of packet forwarding in a large number of network nodes. Thus, the risk of privacy leakage is seriously impacted by the large network hop count in cloud computing. On behalf of this, offloading to edge is usually a better choice for privacy protection.

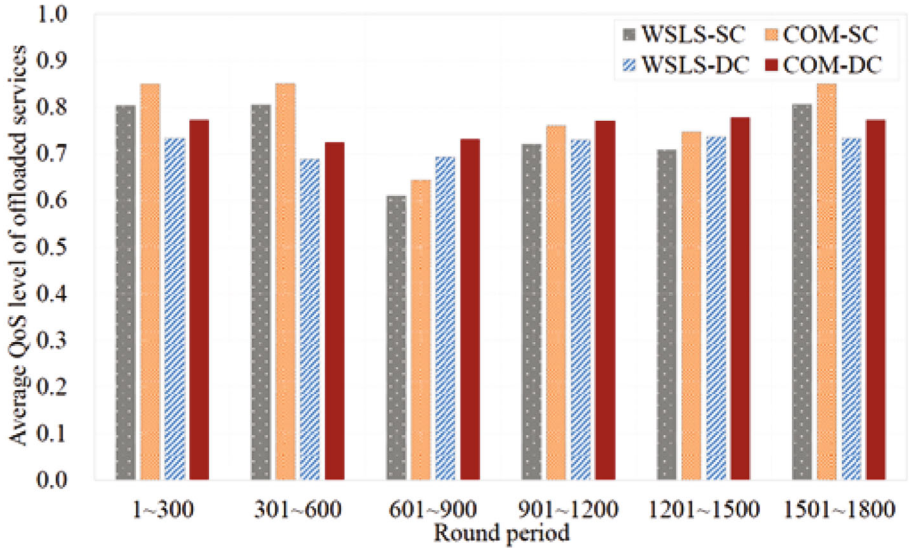

FIGURE 6 Comparison of average QoS level in six periods.

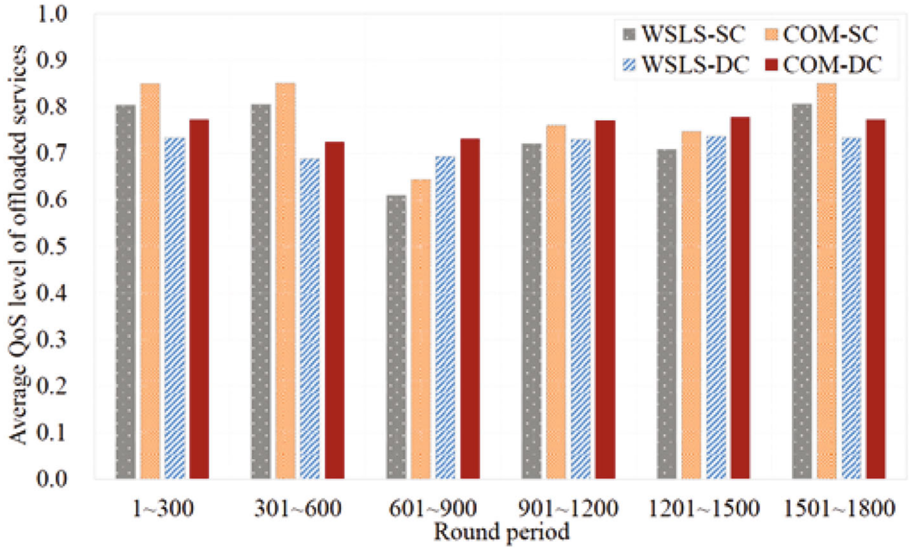

FIGURE 7 Comparison of average privacy level in six periods.

6 | CONCLUSION

This paper considered the scenario of cloud-edge computing for the services of metaverse in smart manufactoring, and proposed a minority game based cloud-edge service offloading method named COM, which enables terminal devices to make offloading decisions independently. In the previous work, the service offloading was modeled as a minority game with a fixed cut-off value. While in this extension, a dynamic controlling method is supplemented to the model for better adaptability to the network environment. Then, an RL-based decision-making scheme is proposed for APs. Based on real-world manufactoring datasets, experiments were conducted with COM and comparative strategies, and the results verified the adaptability and effectiveness of COM.

7 | FUTURE WORK

In the proposed metaverse service offloading framework, APs were fixed infrastructures deployed alongside the buildings, and this feature ensures a fixed number of total agents in MG. To expand the scope of application, moveable agents, for example, smartphones, should be taken into consideration. It remains challenging to make MG effective with a variable number of agents. Besides, the offloading destinations of APs were limited to the cloud or the nearest ECD, while more offloading choices of APs is worth investigation. To achieve multiple offloading destinations, the multiple-choice MG is suggested which may require a more precise decision-making method and take a longer time for convergence.

REFERENCES

1. Yao X, Ma N, Zhang J, Wang K, Yang E, Faccio M. Enhancing wisdom manufacturing as industrial metaverse for industry and society 5.0. *J Intell Manuf* . 2022;1:1-21.
2. Wang Y, Zhao C, Yang S, et al. MPCSM: microservice placement for edge-cloud collaborative smart manufacturing. *IEEE Trans Industr Inform*. 2020;17(9):5898-5908.
3. Zhang S, Yao L, Sun A, Tay Y. Deep learning based recommender system: a survey and new perspectives. *ACM Comput Surv*. 2019;52(1):1-38.

. Wu X, Qi L, Gao J, Ji G, Xu X. An ensemble of random decision trees with local differential privacy in edge computing. *Neurocomputing*. 2022;485:181-195.

5. Lv W, Wang Q, Yang P, et al. Microservice deployment in edge computing based on deep q learning. *IEEE Trans Parallel Distrib Syst*. 2022;33(11):2968-2978.
6. He Q, Cui G, Zhang X, et al. A game-theoretical approach for user allocation in edge computing environment. *IEEE Trans Parallel Distrib Syst*. 2019;31(3):515-529.
7. Ren J, Yu G, He Y, Li GY. Collaborative cloud and edge computing for latency minimization. *IEEE Trans Veh Technol*. 2019;68(5):5031-5044.

. Ren Q, Liu K, Zhang L. Multi-objective optimization for task offloading based on network calculus in fog environments. *Digital Commun Netw*. 2022;8(5):825-833.

9. Qi L, Lin W, Zhang X, Dou W, Xu X, Chen J. A correlation graph based approach for personalized and compatible web apis recommendation in mobile app development. *IEEE Trans Knowl Data Eng*. 2022;35(6):5444-5457.
10. Wang M, Wu T, Ma T, Fan X, Ke M. Users' experience matter: delay sensitivity-aware computation offloading in mobile edge computing. *Digital Commun Netw*. 2022;8(6):955-963.
11. He Q, Tan S, Chen F, et al. EDIndex: enabling fast data queries in edge storage systems. Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2023 675–685.
12. Roman R, Lopez J, Mambo M. Mobile edge computing, fog et al: a survey and analysis of security threats and challenges. *Future Gener Comput Syst*. 2018;78:680-698.
13. Mahenge MPJ, Li C, Sanga CA. Energy-efficient task offloading strategy in mobile edge computing for resource-intensive mobile applications. *Digital Commun Netw*. 2022;8(6):1048-1058.

1 . Zhang M, Zhang , ane ND, et al. Deep learning in the era of e ge computing: challenges an opportunities. og Comput: heory Pract. 2020;1:67-7 .

15. Chen J, Ran X. Deep learning ith e ge computing: a revie . Proc . 2019;107( ):1655-167 .
16. ang , Ding S, iu , et al. ast ireless sensor for anomaly etection base on ata stream in an e ge-computing-enable smart greenhouse. Digital Commun Net . 2022; ( ): 9 -507.
17. Sun , Guo X, Song J, et al. A aptive learning-base tas offloa ing for vehicular e ge computing systems. rans eh echnol. 2019;6 ( ):3061-307 .

1 . Wang , Cao Z, Wang S, et al. Privacy-enhance ata collection base on eep learning for internet of vehicles. rans n ustr nform. 2019;16(10):6663-6672.

19. Zhan , Guo S, i P, Zhang J. A eep reinforcement learning base offloa ing game in e ge computing. rans Comput. 2020;69(6): 3- 93.
20. Gupta M, Benson J, Pat a , San hu R. Secure 2 an 2 communication in intelligent transportation using clou lets. rans Serv Comput. 2020;15( ):1912-1925.
21. Mourtzis D, Panopoulos N, Angelopoulos J, Wang B, Wang . Human centric platforms for personalize value creation in metaverse. J Manuf Syst. 2022;65:653-659.
22. Qian , Chen M, Chen J, Hossain MS, Alamri A. Secure enforcement in cognitive internet of vehicles. nternet hings J. 201 ;5(2):12 2-1250.
23. Nain G, Pattanai K, Sharma G. o ar s e ge computing in intelligent manufacturing: past, present an future. J Manuf Syst. 2022;62:5 -611.

2 . Rahimi H, Picau , Singh KD, Ma husu an G, Costanzo S, Boissier . Design an simulation of a hybri architecture for e ge computing in 5G an beyon . rans Comput. 2021;70( ):1213-122 .

25. Wang , uan J, u G, Chen Q, in R. Minority game for istribute user association in unlicense heterogenous net or s. rans Wirel Commun. 2020;19(6): 220- 233.
26. Rehm S , Goel , Crespi M. he metaverse as me iator bet een technology, tren s, an the igital transformation of society an business. J irtual Worl s Res. 2015; (2):1- .

27. W, Xu X, Li D, et al. Privacy preservation for federated learning with robust aggregation in edge computing. *IEEE Internet Things J*. 2022;10(8):7343-7355.
2 . Xu X, Jiang Q, Zhang P, et al. Game theory for distributed IoV task offloading with fuzzy neural network in edge computing. *IEEE Trans Fuzzy Syst*. 2022;30(11):4593-4604.
29. Wang S, Chen X, Jannach D, Yao L. Causal Decision Transformer for Recommender Systems Via Offline Reinforcement Learning. arXiv preprint arXiv:230407920. 2023.
30. Zhang X, Qi L, Yuan Y. Convergency of AI and cloud/edge computing for big data applications. *Mob Netw Appl*. 2022;27(6):2292-2294.
31. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. *IEEE Internet Things J*. 2016;3(5):637-646.
32. Wang S, Zhao Y, Xu J, Yuan J, Hsu CH. Edge server placement in mobile edge computing. *J Parallel Distrib Comput*. 2019;127:160-168.
33. Xu X, Shen B, Yin X, et al. Edge server quantification and placement for offloading social media services in industrial cognitive IoV. *IEEE Trans Industr Inform*. 2020;17(4):2910-2918.
3 . Zhou Z, Liao H, Zhao X, Ai B, Guizani M. Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty. *IEEE Trans Veh Technol*. 2019;68(9):8322-8335.
35. He X, Jin R, Dai H. Peace: privacy-preserving and cost-efficient task offloading for mobile-edge computing. *IEEE Trans Wirel Commun*. 2019;19(3):1814-1824.
36. Dinh TQ, La QD, Quek TQ, Shin H. Learning for computation offloading in mobile edge computing. *IEEE Trans Commun*. 2018;66(12):6353-6367.
37. Li K. A game theoretic approach to computation offloading strategy optimization for non-cooperative users in mobile edge computing. *IEEE Trans Sustain Comput*. 2018;1:1-12.
3 . Zhao L, Wang Q, Zou Q, Zhang Y, Chen Y. Privacy-preserving collaborative deep learning with unreliable participants. *IEEE Trans Inf Forensics Secur*. 2019;15:1486-1500.
39. Osia SA, Shamsabadi AS, Sajadmanesh S, et al. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet Things J*. 2020;7(5):4505-4518.
0. Challet D, Zhang YC. On the minority game: analytical and numerical studies. *Phys A: Stat Mech Appl*. 1998;256(3-4):514-532.
1. Shen B, Xu X, Dai F, Qi L, Zhang X, Dou W. Dynamic task offloading with minority game for internet of vehicles in cloud-edge computing. Paper presented at: IEEE International Conference on Web Services (ICWS) IEEE. 2020 372–379.
2. Ranadheera S, Maghsudi S, Hossain E. Computation offloading and activation of mobile edge computing servers: a minority game. *IEEE Wirel Commun Lett*. 2018;7(5):688-691.
3. Huang H, Cai Y, Xu H, Yu H. A multiagent minority-game-based demand-response management of smart buildings toward peak load reduction. *IEEE Trans Comput-Aided Design Integr Circuits Syst*. 2016;36(4):573-585.
. Cao Y, Chen X, Yao L, Wang X, Zhang WE. Adversarial attacks and detection on reinforcement learning-based interactive recommender systems. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). 2020 1669–1672.
5. Zhang SP, Zhang JQ, Huang ZG, Guo BH, Wu ZX, Wang J. Collective behavior of artificial intelligence population: transition from optimization to game. Nonlinear Dyn. 2019;95:1627-1637.
6. Reents G, Metzler R, Kinzel W. A stochastic strategy for the minority game. Phys A: Stat Mech Appl. 2001;299(1-2):253-261.