Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Incentive-based task offloading for digital twin in 6G native AI networks: a learning approach*

Tianjiao CHEN[†‡1,2], Xiaoyun WANG[3], Meihui HUA[1], Qinqin TANG[4]

[1]*China Mobile Research Institute, Beijing 100053, China*

[2]*ZGC Institute of Ubiquitous-X Innovation and Applications, Beijing 100080, China*

[3]*China Mobile Communications Group Corporation, Beijing 100032, China*

[4]*Beijing University of Posts and Telecommunications, Beijing 100876, China*

[†]E-mail: chentianjiao@chinamobile.com

Received Mar. 30, 2024; Revision accepted July 25, 2024; Crosschecked

Abstract: A communication network can natively provide artificial intelligence (AI) training services for resource-limited network entities to quickly build accurate digital twins and achieve high-level network autonomy. Considering that network entities that require digital twins and those that provide AI services may belong to different operators, incentive mechanisms are needed to maximize the utility of both. In this paper, we establish a Stackelberg game to model the AI training task offloading for digital twins in native AI networks with the operator of base stations as the leader and resource-limited network entities as the followers. We analyze the Stackelberg equilibrium to obtain equilibrium solutions. Considering the time-varying wireless network environment, we further design a deep reinforcement learning algorithm to achieve dynamic pricing and task offloading. Finally, extensive simulation experiments are conducted to verify the effectiveness of our proposal.

unedited

Key words: Digital twin network; Native AI; Stackelberg game; Task offloading; Deep reinforcement learning
https://doi.org/10.1631/FITEE.2400240 **CLC number:**

1 Introduction

To meet the digital and intelligent requirements of various industries in the future, the sixth-generation (6G) wireless network extends typical scenarios of networks to immersive communication, integrated artificial intelligence (AI) and communication, hyper reliable and low-latency communication, ubiquitous connectivity, massive communication, and integrated sensing and communication (Wang et al., 2023; Zhang et al., 2024). In order to adapt to differentiated new scenarios and meet higher performance requirements, the network needs to have multidimensional capabilities that integrate communication, sensing, computing, AI, big data, and security (Nie et al., 2022). Consequently, the 6G network architecture will undergo significant changes to a platformized and integrated service system, providing users with diversified services that are dynamically matched on demand (Nguyen et al., 2022; Wu et al., 2023). However, the superposition of capabilities may lead to a complex network architecture. The current mode of network operation and maintenance, which mainly relies on manual experience and operation, will face great challenges in the future. Hence, high-level network autonomy has become a necessary condition for achieving the 6G vision.

Fortunately, the emergence of digital twin networks has provided a way to achieve network autonomy. Digital twin networks have precise perception

‡ Corresponding author
* Project supported by the National Key R&D Program of China (Project Number: 2022YFB2902100)
ORCID: Tian-jiao CHEN, https://orcid.org/0000-0002-2931-3487
© Zhejiang University Press 2024

and collection capabilities for physical network data, constructing high-fidelity digital twins and providing a virtual network environment that approximates reality (Mihai et al., 2022; Zhang et al., 2023). In a virtual network environment, future network states and potential failures can be predicted, and new decisions can be validated without considering trial-and-error risks. The preemptive intervention and verification of digital twin networks can improve network operation and maintenance efficiency, improve decision-making accuracy, and reduce the risk of network failures (Lu et al., 021b).

However, the accuracy of digital twin network prediction and verification relies on finding patterns from massive network data, requiring the training of AI models to support the construction of digital twins (Alexopoulos et al., 2020; Groshev et al., 2021). Native AI networks can utilize ubiquitous computing, data, AI models, and other resources within the network to achieve on-demand offloading and full life-cycle management of AI tasks, providing higher-quality AI services for digital twin networks (6GANA, 2022; Institute, 2022). Considering some resource-limited network entities (RL-NEs) in 6G, such as access points, the AI training tasks required for digital twin networks need to be offloaded to the servers of base stations (BSs) for execution. Some works have adopted a centralized offloading strategy, whereby BSs collect all necessary information from each RL-NE to seek globally optimal performance. However, in certain business-to-business (B2B) scenarios, RL-NEs are owned by different enterprises. In order to protect the privacy of decision-making, distributed offloading frameworks are more inclined to be adopted to allow each RL-NE to independently make offloading decisions based on its own needs. In distributed frameworks, economic incentives

In native AI networks for digital twins, the BS is owned by the operator (Liu et al., 2022). The operators can contribute their idle resources to assist network nodes in processing tasks. Incentives are used to encourage RL-NEs to offload AI training tasks, thereby alleviating resource shortages. The operator is selfish and hopes to maximize revenue by pricing its AI training services. RL-NEs, on the other hand, pay for the AI training services they receive from the operator and hope to maximize their utility. Therefore, there is a game relationship between the operator and RL-NEs. An incentive-based AI training service scheme is necessary to simultaneously increase operator revenue and maximize the utility of RL-NEs.

To this end, this paper proposes a Stackelberg game model to analyze AI training service interactions between BSs and RL-NEs in native AI networks. Especially, considering the dynamic wireless network environment of mobile RL-NEs, we investigate the use of deep reinforcement learning (DRL) to manage pricing and AI training service offloading. Specifically, the main contributions of this paper are summarized as follows:

- We establish a Stackelberg game model to solve the task-offloading problem for digital twins in 6G native AI networks. Specifically, the operator with BSs is denoted as the leader in determining AI training service pricing, while mobile RL-NEs are followers who make offloading decisions according to the announced pricing.
- We analyze the Stackelberg equilibrium (SE) of the proposed Stackelberg game to obtain equilibrium solutions for the operator and mobile RL-NEs.
- Considering the dynamic wireless network environment, we further design a DRL algorithm based on the soft actor-critic (SAC) method to enable dynamic pricing and task offloading for the operator and mobile RL-NEs.

The remainder of this paper is organized as follows. In Section 2, we introduce the related work. In Section 3, we present the system model of native AI networks for mobile RL-NEs digital twins. In Section 4, we formulate the Stackelberg game for task offloading. Section 5 develops the DRL-based algorithm for dynamic pricing and offloading. Simulations are conducted in Section 6, and Section 7 presents the conclusion.

2 Related work

igital twin networks can create virtual network environments before network deployment; validate the performance and effectiveness of new technologies, strategies, or configurations; and reduce trial-and-error costs. Currently, digital twin networks leverage AI algorithms to enhance their capabilities. Liu *et al.* (Liu et al., 2023) introduced a novel

unedited

paradigm, namely, digital twins, to generate virtual replicas of physical objects within the Internet of things (IoT) networks. Building upon this concept, they proposed a bidirectional gated recurrent unit algorithm based on federated learning to forecast resource requirements. Additionally, they utilized the DRL algorithm to make virtual network function (VNF) migration decisions, effectively reducing the number of VNF migrations needed and minimizing energy consumption in IoT networks. Yao *et al.* (Yao et al., 2023) established a virtual twin of the mobile edge computing network and proposed a graph attention-based multiagent reinforcement learning algorithm. This algorithm was designed to learn the optimal strategy for task offloading and service caching within the virtual twin environment. Jiang *et al.* (Jiang et al., 2022) proposed a framework for constructing digital twins of smart devices at the network edge, belonging to different mobile network operators. They devised a joint cooperative learning and local model update verification scheme to encourage mobile access points (mAPs) to assist with local model training. Addressing the security concerns associated with cloud data storage, Lv and Lou (Lv and Lou, 2022) developed a digital twin of intelligent manufacturing. They used deep learning techniques to detect and intercept malicious intrusions, while leveraging edge fog cloud computing and encryption technology to ensure secure data storage.

The AI application of digital twins necessitates the utilization of communication, computing, data, and AI model resources. The convergence of communication, computing, and caching (3Cs) can support network intelligence to adapt to the ever-changing services in 6G (Zhou et al., 2020). Moreover, native AI networks have been proposed to schedule AI tasks and provide AI training, inference, and other functionalities. Yang *et al.* (Yang et al., 2024) introduced a task-oriented native AI network architecture (TONA) to inherently support network AI and ensure personalized quality of experience for each user. Limited terminal computing resources can hardly meet the computing needs for uses such as AI services; therefore, Qi *et al.* (Qi et al., 2021) innovatively proposed a traffic-aware task offloading (TATO) mechanism based on the fusion of communication and perception, which can effectively reduce the total response time. Similarly, Wu *et al.* (Wu et al., 2022) proposed a native-AI network slicing architecture to facilitate intelligent network management and support AI services in 6G networks, achieving synergy between AI and network slicing. Furthermore, Hossain *et al.* (Hossain et al., 2023) presented the concept of AI-native 6G to enhance understanding of the control plane and user plane network function configurations of 6G core networks, aiming to achieve effective core network management.

For distributed scenarios, particularly when digital twin users and their required AI providers belong to different operators, game theory has emerged as a potent tool for solving task-offloading problems within networks. Tang *et al.* (Tang et al., 2022) investigated the distributed task-offloading problem in serverless edge computing networks. They modeled the task-offloading process as a partially observable stochastic game, enabling heterogeneous serverless edge computing nodes to optimize themselves in partially observable environments, thus enhancing offloading utility. To address the challenge of completing computing tasks within time-varying 5G wireless heterogeneous networks, Du *et al.* proposed a cloud computing resource sharing and offloading mechanism based on Stackelberg differential game in (Du et al., 2022). proposed a cloud computing resource-sharing and offloading mechanism based on Stackelberg differential game. Additionally, a game model for real-time task-offloading problems was developed (Liu et al., 2024), and a two-tier offloading approach was proposed to enhance the effectiveness and efficiency of real-time task offloading. This approach aims to reduce energy consumption and improve service-combining capability and service availability for decentralized cloud manufacturing resource providers.

In summary, although there have been some studies on enhancing the AI capabilities of digital twins and optimizing native AI task offloading, the game relationship between service provider and users of AI for digital twins has not been considered yet. Therefore, this paper uses the Stackelberg game model to explore an incentive-based AI training task-offloading scheme for AI-empowered digital twins. Its purpose is to strengthen incentives for native AI networks and maximize the utility of mobile RL-NEs. In addition, this paper also considers the dynamic changes in wireless environments to design dynamic pricing and task-offloading strategies.

unedited

3 System model

3.1 Network model

As shown in Fig. 1, we consider a native AI network to provide various AI training services for digital twins of mobile RL-NEs. The mAPs carried on vehicles and unmanned aerial vehicles are selected as a type of mobile RL-NEs. A group of BSs are distributed in a limited area, serving multiple mAPs belonging to different enterprises. Each BS has communication and computing capabilities, and it stores AI models of different accuracy levels for the construction of digital twins.

The mAP collects data from the physical world, trying to accurately reflect real user behavior and network activities. User behavior data include mobility patterns, preferences, speed, location changes, communication needs, and so on. Network behavior data mainly refer to wireless network characteristics, such as channel conditions, bandwidth utilization, and transmission delays (Lu et al., 021a). After collecting data, using methods such as trajectory analysis and historical data mining, the mAP can build a more realistic and accurate digital twin network model to achieve rapid decision-making. The mAPs in these areas have different AI model-training requirements for building their own digital twin, such as traffic planning, antenna tuning, and fault prediction. Due to limited local resources, mAPs expect the BSs to provide AI training services. The operator operates all BSs and sells its AI training services to mAPs. Each mAP selects a BS to offload tasks and pays for service.

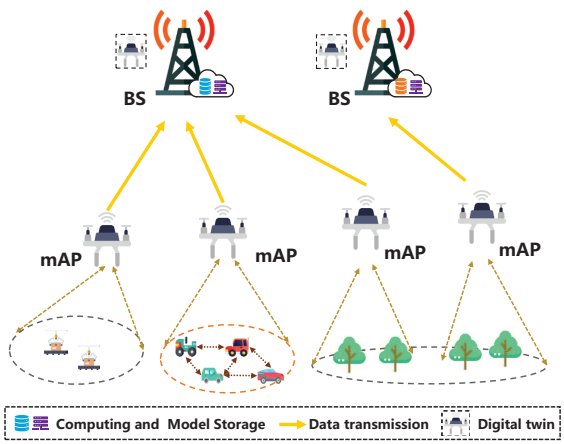


Fig. 1 The native AI network architecture for digital twin networks.

Table 1 List of notations

Symbol	Description
N	Total number of mAPs
M	Total number of BSs
$\chi_{n,m}$	Connection status between BS and mAP
$h_{n,m}$	Channel gain
W_m	Channel bandwidth
p_n	Transmission power
σ^2	Variance of the Gaussian noise
R_n	Uplink data rate
Z_n	Task size
c_n	CPU cycles required
ℓ_n	Selected sampling rate
o_n	Size of raw sampling rate
f_m	Computing capability of core
$T_{n,m}^{\text{trans}}$	Uplink transmission delay
$T_{n,m}^{\text{cmp}}$	Computing delay
$T_{n,m}^{\text{cpl}}$	Task completion delay
$\boldsymbol{\omega}_n^*$	Optimal model parameter
$\varphi_{n,m}$	Effectiveness of the AI model
U^l	Utility function for the leader
U^f	Utility function for the follower
$\mu_{n,m}$	Price to process a unit bit of task
ρ_m, x_m	Weight factor of AI training
γ_n^A, γ_n^D	Weight factor of utility function

Typically, AI model performance after training is usually related to the structure of the AI model and the quality of data used for training. Therefore, mAPs can collect high-quality data as much as possible from the physical network based on the price set by the operator for the BSâĂŹs computing resources and stored AI models to obtain better training results. The definitions of the main notations used in this article are summarized in Table 1.

3.2 Communication model

Each mAP needs to establish a connection with a BS to transmit its AI training tasks. The mAPs are denoted as $\mathcal{N} = \{1, 2, \ldots, N\}$, and the BSs are denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$. A set of binary variables $\chi_{n,m}$ indicates the connection status of an mAP $n \in \mathcal{N}$ and a BS $m \in \mathcal{M}$. Here, $\chi_{n,m} = 1$ indicates that the mAP n is connected to BS m, and $\chi_{n,m} = 0$ indicates otherwise. Note that each mAP can only be connected to one BS, which means $\sum_{m\in\mathcal{M}} \chi_{n,m} = 1$.

All mAPs connected to the same BS share the same frequency band, which creates interference. We

unedited

denote the channel gain between mAP n and BS m as $h_{n,m}$, which is related to the distance from mAP to the BS (Zhao et al., 2022). The mAP n's uplink data rate to the BS m is as follows:

$$R_n[m] = W_m \log_2 \left(1 + \frac{p_n h_{n,m}}{\sum\limits_{n' \in \mathcal{N}/\{n\}} \chi_{n',m} p_{n'} h_{n',m} + \sigma^2} \right). \tag{1}$$

Here, W_m is the BS m's channel bandwidth, p_n is mAP n's transmission power, and σ^2 is the variance of the Gaussian noise.

3.3 Computing model

After establishing the connection, the mAPs will transmit their AI training tasks to the BSs. Let Z_n represent the size, in bits, of the task generated by mAP n, and c_n denote the number of central processing unit (CPU) cycles required to process one unit bit of the AI training task. Let $\ell_n \in [0,1]$ denote the selected sampling rate of mAP n, where $\ell_n = 0$ indicates that mAP n does not collect any data, and $\ell_n = 1$ indicates that mAP n collects data at the raw sampling rate. The data from mAPs within a time slot are considered to comprise an AI training task, which can be scheduled to the BSs. The higher the sampling rate, the higher is the quality of the data, and consequently, the larger is the size of the AI training task generated. We have $Z_n = \ell_n \cdot o_n$, where o_n represents the size of the task generated when mAP n collects data at the raw sampling rate. The mAPs will utilize the computing resources and the AI training model deployed at the BS to complete their task. The task completion delay of mAP n consists of the uplink transmission delay, the computing delay at the BS, and the return transmission delay. Since the results of AI training tasks are usually small, we neglect the return transmission delay for sending task results from the BS to the mAP.

The uplink transmission delay from mAP n to BS m can be calculated as follows:

$$T_{n,m}^{\text{trans}} = \frac{Z_n[m]}{R_{n,m}}. \tag{2}$$

Here, $Z_n[m] = \chi_{n,m} Z_n$ represents the size of mAP n's data that are transmitted to BS m for processing.

Each BS m has δ_m number of CPU cores for processing computing tasks, and the processing capacity (in CPU cycles per second) of these CPU cores is exactly the same, i.e., f_m. Then, the computing delay of mAP n's task at the BS m is as follows:

$$T_{n,m}^{\text{cmp}} = \frac{Z_n[m] c_n}{f_m}. \tag{3}$$

Thus, the task completion delay of the task of mAP n on BS m is

$$T_{n,m}^{\text{cpl}} = T_{n,m}^{\text{trans}} + T_{n,m}^{\text{cmp}}. \tag{4}$$

3.4 AI training model

The process of training an AI model involves working with a substantial volume of data samples. In conventional AI training, given a data sample $\{\boldsymbol{z}_i, y_i\}$ comprising a multidimensional input feature $\boldsymbol{z}_i$, the objective is to determine a model parameter vector $\boldsymbol{\omega}$ that accurately represents the labeled output y_i based on a loss function $loss_i(\boldsymbol{\omega})$. The loss function for a local dataset, containing a total of D data samples, can be defined as follows:

$$Loss(\boldsymbol{\omega}) = \frac{1}{D} \sum_{i \in D} loss_i(\boldsymbol{\omega}) + \xi g(\boldsymbol{\omega}). \tag{5}$$

Here, $g(\cdot)$ represents a regularizer function, defined as $g(\cdot) = \frac{1}{2} \|\cdot\|^2$; $\forall \xi \in [0,1]$.

Let $\boldsymbol{\omega}_n^*$ denote the optimal model parameter for mAP n. The mAP n conducts iterative training of its local AI model, as described by (Yang et al., 2021):

$$\boldsymbol{\omega}_n^* = \arg\min\nolimits_{\boldsymbol{\omega}} Loss_n \left(\boldsymbol{\omega} | \boldsymbol{\omega}_n, \nabla Loss_n(\boldsymbol{\omega}) \right). \tag{6}$$

The effectiveness of an AI model can be assessed by its accuracy, represented by $\varphi \in [0,1]$. This accuracy is influenced by several factors including the data quality, data size, and the training algorithm used. Similar to the approach described by (Chen et al., 2023), the effectiveness of the AI model for mAP n processed by BS m conforms to the following expression:

$$\varphi_{n,m} = 1 - \exp\left(-\rho_m Z_n[m] x_m \right). \tag{7}$$

Here, ρ_m represents the weight factor, while x_m serves as a weight factor that reflects the AI training service capability deployed at BS m, contingent upon the structure of the deployed AI model.

In order to evaluate the quality of the solution, it is required that φ satisfies the following:

$$\|Loss(\boldsymbol{\omega}^*)\| \le (1 - \varphi) \|Loss(\mathbf{0})\|. \tag{8}$$

Here, achieving $\varphi = 1$ requires pinpointing the exact maximum, whereas $\varphi = 0$ indicates no improvement has been attained at the mAP.

unedited

4 Stackelberg game formulation

In this section, we use the Stackelberg game to devise an incentive-based pricing and offloading scheme. The operator aims to leverage the computing resources of BSs to offer AI training services, thus optimizing resource utilization. On the other hand, mAPs seek to enhance their task-processing efficiency by procuring these AI training services.

The operator initiates the game by setting the unit service price based on the current network conditions. Subsequently, mAPs select a specific BS for access and determine the data size to transmit, considering the price set by the operator. Each mAP independently devises wireless access and task-offloading strategies, leading to a distributed decision-making process that ultimately converges to an equilibrium. Hence, the game involves one leader (operator) and multiple followers (mAPs).

4.1 Utility functions

In our network scenarios, the operator assumes the role of the leader, aiming to maximize its utility through the pricing of AI training services. The operatorâĂŹs utility is delineated by a trade-off between the profit accrued from selling AI training services and the energy cost associated with processing the tasks offloaded by the mAPs. Mathematically, the utility function for the leader (operator) can be expressed as follows:

$$U^l(\mu_{n,m}, Z_n[m]) = \sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\left(\mu_{n,m}Z_n[m] - \epsilon\left(f_m\right)^2 Z_n[m]c_n\right). \quad (9)$$

Here, $\mu_{n,m}$ represents the price for BS m to process a unit bit of task from mAP n, and ϵ denotes the energy coefficient (Lin et al., 2022).

The first term encapsulates the profit derived from vending AI training services, while the second term denotes the energy consumption of BSs in executing AI training tasks received from the mAPs. Consequently, the operator's objective entails setting an appropriate unit service price $\mu_{n,m}$ to maximize its utility. This problem can be formulated as follows:

$$\textit{Problem (I):} \quad \max_{\mu_{n,m}} \quad U^l(\mu_{n,m}, Z_n[m])$$
$$s.t. \quad \mu_{n,m} > 0, \forall n \in \mathcal{N}, m \in \mathcal{M}. \quad (10)$$

In *Problem (I)*, $Z_n[m]$ can be determined by mAP n. In our Stackelberg game, the mAPs act as followers. Each mAP possesses knowledge of the entire network state and makes decisions simultaneously in a distributed manner. The computing resources of the BS are finite, with a fixed number of CPU cores, thereby restricting each BS to serve only a limited number of mAPs. Consequently, the offloading delay for an mAP is influenced not only by its own offloading strategy but also by the offloading strategies of other mAPs accessing the same BS. Upon the operator setting the unit service price, the utility of follower (mAP) n under the access of BS m is defined as follows:

$$U_n^f(\mu_{n,m}, Z_n[m]) = \gamma_n^A\varphi_{n,m} - \gamma_n^D T_{n,m}^{\text{cpl}} - \mu_{n,m}Z_n[m]. \quad (11)$$

Here, the first term represents the revenue gained from AI training, the second term denotes the cost incurred due to processing delay, and the third term signifies the payment to the operator. Each follower (mAP) aims to determine the appropriate BS for access and the optimal scheduled task size, $Z_n[m]$, to maximize its own utility function. Mathematically, the optimization problem for mAP n can be defined as follows:

$$\textit{Problem (II):} \quad \max_{Z_n[m]} \quad U_n^f(\mu_{n,m}, Z_n[m])$$
$$s.t. \quad 0 \le Z_n[m] \le g_n, \forall n \in \mathcal{N}, m \in \mathcal{M}. \quad (12)$$

The combination of Problems (I) and (II) constitutes the Stackelberg game. The objective is to identify an SE point, wherein neither the operator nor the mAPs are incentivized to deviate from their strategies.

4.2 The SE point

The SE point of our scenario is defined as follows.

Definition 1. Let $\mu_{n,m}^*$ be a solution for *Problem (I)* and $Z_n[m]$ be a solution for *Problem (II)* of mAP n. Then, the point $\left(\mu_{n,m}^*, Z_n^*[m]\right)$ is an SE for the proposed Stackelberg game if for any $(\mu_{n,m}, Z_n[m])$ with $\mu_{n,m}$ and $Z_n[m]$, the following conditions are satisfied:

$$U^l(\mu_{n,m}^*, Z_n^*[m]) \ge U^l(\mu_{n,m}^*, Z_n[m]). \quad (13)$$

$$U_n^f(Z_n^*[m], Z_{-n}^*[m], \mu_{n,m}^*) \ge U_n^f(Z_n[m], Z_{-n}^*[m], \mu_{n,m}^*). \quad (14)$$

unedited

In the proposed game, the mAPs engage in strict non-cooperative competition. Among the mAPs, a non-cooperative task offloading subgame is conducted, wherein the Nash equilibrium (NE) represents a state wherein no participant can unilaterally alter its strategy to enhance its utility while others maintain their strategies. Conversely, the operator can optimize its response by addressing *Problem (I)*. To attain the SE of the proposed game, *Problem (II)* is initially solved for a given $\mu_{n,m}$. Subsequently, utilizing the obtained best response $Z_n^*[m]$ from the mAPs, *Problem (I)* is tackled to derive the optimal price $\mu_{n,m}^*$.

4.3 Follower analysis

Once the operator provides the unit price $\mu_{n,m}$, each mAP selects an offloading strategy to maximize its utility. In order to achieve a higher utility U^f, the mAPs endeavor to offload more tasks. However, offloading an excessive number of tasks may lead to overpayment, outweighing the benefits of delay reduction. Consequently, the appropriate offloading strategy for each mAP is analyzed as follows.

Proposition 1. For a given unit price $\mu_{n,m}$, the optimal solution for *Problem (I)* is given by

$$Z_n^*[m] = -\frac{1}{\rho_m x_m} \ln\left(\frac{\gamma_n^D\left(f_m + c_n R_{n,m}\right)}{\gamma_n^A \rho_m x_m R_{n,m} f_m} + \frac{\mu_{n,m}}{\gamma_n^A \rho_m x_m}\right). \tag{15}$$

Proof. According to Eq. (11), the utility function of mAP n under the accessed BS m is given as

$$\begin{aligned} U_n^f(\mu_{n,m}, Z_n[m]) &= \gamma_n^A\left(1 - \exp\left(-\rho_m Z_n[m] x_m\right)\right) \\ &- \gamma_n^D\left(\frac{Z_n[m]}{R_{n,m}} + \frac{Z_n[m] c_n}{f_m}\right) - \mu_{n,m} Z_n[m]. \end{aligned} \tag{16}$$

For each mAP n, the first derivative of $U_n^f(\mu_{n,m}, Z_n[m])$ with respect to $Z_n[m]$ is

$$\begin{aligned} \frac{\partial U_n^f(\cdot)}{\partial Z_n[m]} &= \gamma_n^A \rho_m x_m \exp\left(-\rho_m Z_n[m] x_m\right) \\ &- \gamma_n^D\left(\frac{1}{R_{n,m}} + \frac{c_n}{f_m}\right) - \mu_{n,m}. \end{aligned} \tag{17}$$

Then, the second derivative is as follows:

$$\frac{\partial^2 U_n^f(\cdot)}{\partial Z_n^2[m]} = -\gamma_n^A\left(\rho_m x_m\right)^2 \exp\left(-\rho_m Z_n[m] x_m\right). \tag{18}$$

It is obvious that $\frac{\partial^2 U_n^f(\cdot)}{\partial Z_n^2[m]} < 0$. Therefore, $U_n^f(\mu_{n,m}, Z_n[m])$ is a concave function with respect to $Z_n[m]$, and $Z_n^*[m]$ is the game equilibrium solution. We set the first derivative equal to zero and obtain the value of $Z_n^*[m]$ as Eq. (15). □

According to Eq. (15), we can get the optimal task-offloading strategy since the offloading policy of a task is always > 0. In addition, the price set by the BS is always not negative. Therefore, we have the following relation:

$$0 \le \mu_{n,m} \le \gamma_n^A \rho_m x_m - \frac{\left(f_m + c_n R_{n,m}\right)}{R_{n,m} f_m} = \mu_{n,m}^{\max}. \tag{19}$$

4.4 Leader analysis

The leaderâĂŹs strategy is influenced by the followers, culminating in the formation of a Stackelberg game. A higher unit service price prompts followers to decrease the amount of data they offload or to opt for alternative BSs. While this may lead to a reduction in the service fee obtained, it also mitigates the energy cost incurred by the BSs for data processing. Conversely, setting a lower unit price may not enable the leader to attain optimal utility. Therefore, the appropriate price setting for each operator is analyzed as follows.

By substituting the followersâĂŹ optimal task-offloading strategy in Eq.(15) into the operator's utility function, we get

$$\begin{aligned} &U^l(\mu_{n,m}, Z_n[m]) = \\ &\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\left(\mu_{n,m} - \epsilon\left(f_m\right)^2 c_n\right)\cdot\left(-\frac{1}{\rho_m x_m}\right)\cdot \\ &\ln\left(\frac{\gamma_n^D\left(f_m + c_n R_{n,m}\right)}{\gamma_n^A \rho_m x_m R_{n,m} f_m} + \frac{\mu_{n,m}}{\gamma_n^A \rho_m x_m}\right). \end{aligned} \tag{20}$$

Proposition 2. Given the followers' best response offloading strategy, the optimal unit service price $\mu_{n,m}^*$, $\forall n \in \mathcal{N}$, $\forall m \in \mathcal{M}$ can be uniquely obtained.

unedited

Proof. The first derivative of $U_n^l(\mu_{n,m}, Z_n[m])$ is

$$\frac{\partial U_m^l(\cdot)}{\partial \mu_{n,m}} = \left(-\frac{1}{\rho_m x_m}\right) \cdot \left(\ln\left(\frac{\gamma_n^D\left(f_m + c_n R_{n,m}\right)}{\gamma_n^A \rho_m x_m R_{n,m} f_m} + \frac{\mu_{n,m}}{\gamma_n^A \rho_m x_m}\right) + \frac{R_{n,m} f_m \left(\mu_{n,m} - \epsilon\left(f_m\right)^2 c_n\right)}{\gamma_n^D\left(f_m + c_n R_{n,m}\right) + \mu_{n,m} R_{n,m} f_m}\right). \quad (21)$$

The second derivative of $U_n^l(\mu_{n,m}, Z_n[m])$ is

$$\frac{\partial^2 U_m^l(\cdot)}{\partial \mu_{n,m}^2} = \left(-\frac{1}{\rho_m x_m}\right) \cdot \left(\frac{R_{n,m} f_m}{\gamma_n^D\left(f_m + c_n R_{n,m}\right) + \mu_{n,m} R_{n,m} f_m} + \frac{c_n\left(R_{n,m}\right)^2\left(f_m\right)^4}{\left(\gamma_n^D\left(f_m + c_n R_{n,m}\right) + \mu_{n,m} R_{n,m} f_m\right)^2} + \frac{R_{n,m} f_m \gamma_n^D\left(f_m + c_n R_{n,m}\right)}{\left(\gamma_n^D\left(f_m + c_n R_{n,m}\right) + \mu_{n,m} R_{n,m} f_m\right)^2}\right) < 0. \quad (22)$$

The utility function of the operator is a strictly concave function; thus, there is a game equilibrium solution. We make $\frac{\partial U_m^l(\cdot)}{\partial \mu_{n,m}} = 0$ and then obtain the optimal unit price $\mu_{n,m}^*$, as shown in Eq. (19). □

4.5 Joint BS access and task offloading

Building upon the analysis of the leader (operator) and followers (mAPs), an appropriate unit price for the leader and a suitable offloading strategy for each mAP can be determined. Furthermore, each mAP must select an appropriate BS for access. After the operator sets its service price, according to the optimal offloading strategy calculated in Eq.(9), the BS access strategy of mAP n is given by

$$m^* = \arg\max_m \left(U_m^f(\mu_{n,m}^*, Z_n^*[m])\right). \quad (23)$$

Hence, mAP n chooses BS m^* to maximize its utility function, i.e., $\chi_{n,m} = 1$.

Based on Propositions 1 and 2, there exists a game equilibrium point in both the first and the second stages, as defined in Definition 1 (Shi et al., 2022). To compute the game equilibrium point, we use a gradient-based iteration algorithm outlined in Algorithm 1. In the considered scenario, it is assumed that each mAP shares information regarding its task and is aware of the offloading strategies of other mAPs. In each iteration of the algorithm, the pricing strategy is first computed using the derivative direction. Subsequently, the offloading strategy is determined based on the pricing strategy. As the number of iterations increases, the operatorâĂŹs price and the mAPsâĂŹ offloading strategies converge toward their optimal values. If multiple mAPs choose the same BS, their offloading experience deteriorates. Consequently, in the subsequent iteration, the operator may announce a new price, prompting mAPs to choose different BSs or reduce their scheduled tasks, as per the aforementioned analysis. Through multiple interactions and iterations, despite each mAP acting in its self-interest, an equilibrium is reached wherein both the leader and the followers attain increased utility. According to Algorithm 1, the complexity is $\mathcal{O}\left(\mathcal{W}^2 N\right)$, where $\mathcal{W}$ is the iteration number and N is the number of mAPs.

Algorithm 1 The gradient iteration algorithm

Require: The task information of mAPs; the communication information between mAPs and BSs

Ensure: The optimal pricing strategy $\mu_{n,m}^*$, the optimal offloading strategy $Z_n^*[m]$

```
Initialize the pricing strategy μ_{n,m}(0) with a random number, ∀n ∈ N, ∀m ∈ M;
Initialize the iteration number W;
for ω = 1, 2, ..., W, do
    Update the price for each mAP μ_{n,m}(ω+1) = μ_{n,m}(ω) + κ ∂U^l(·)/μ_{n,m}(ω);
    for ω = 1, 2, ..., W, do
        Update the offloading strategy for each mAP Z_n[m](ω+1) = Z_n[m](ω) + ν ∂U_n^f(·)/Z_n[m](ω);
        Select the BS to access for each mAP n(w + 1) = arg max_n (U_n^f(μ_{n,m}(ω+1), Z_n[m](ω+1)))
    end for
end for
```

5 DRL-based pricing and task scheduling algorithm

The proposed Stackelberg game approach provides a viable strategy for determining pricing and offloading in a communication stage. However, in the dynamic environment of 6G native AI networks for digital twins, strategies will change over time. Con-

unedited

sequently, executing the game anew with each environment change impedes efficient decision-making. Moreover, in practical scenarios, mAPs may struggle to acquire information about offloading decisions from other mAPs (Zhu et al., 2022). To mitigate this challenge, inspired by the DRL approach (Peng and Shen, 2020; Xiong et al., 2019), we further propose a DRL-based mechanism for pricing and task offloading. This mechanism adapts to changing network environments and makes decisions without requiring complete information sharing.

In the DRL environment, the operator assumes the role of the leader. In each training round, the operator observes the current communication environment, including the channel gain between mAPs and BSs, to determine the pricing strategy for the subsequent round. As mAPs lack information about other mAPs, they independently learn the optimal strategy. We use the DRL algorithm to train mAPs in learning the offloading model to derive the optimal offloading strategy. Following the completion of each training round, the current state transitions to the next state, and rewards are assigned to both the operator and the mAPs. Through iterative training, the final DRL-based pricing and offloading algorithm learn the optimal operator pricing strategy and mAP task-offloading strategy.

5.1 SAC-based pricing strategy for the operator

We propose a pricing strategy based on the SAC algorithm (Haarnoja et al., 2018; Tang et al., 2024). The operator determines the prices for unit AI training service based on the communication information between the BSs and the mAPs. To comprehensively capture the system dynamics, such as time-varying channel gain, we adopt a discrete time-slotted model. Here, t denotes the time index, with $t \in \{0, 1, \ldots, T\}$.

5.1.1 State

First, the controller collects the channel gain between the mAPs and the BSs periodically for learning. The state $s_t \in S$ of 6G native AI networks at time slot t is given by

$$\begin{aligned} s_t = \big[&h^t_{1,1}, \ldots, h^t_{1,m}, \cdots, h^t_{1,M}, \ldots, \\ &h^t_{n,1}, \ldots, h^t_{n,m}, \ldots, h^t_{n,M}, \ldots, \\ &h^t_{N,1}, \ldots, h^t_{N,m}, \ldots, h^t_{N,M}\big]. \end{aligned} \tag{24}$$

Here, $h^t_{n,m}$ ($\forall n \in \mathcal{N}$, $\forall m \in \mathcal{M}$) is the channel gain between the mAP n and the BS m at time slot t.

5.1.2 Action

In time slot t, after the operator obtains the system state of the BSs, it determines the pricing strategy. The operator's action a_t is defined as

$$\begin{aligned} a_t = \big[&\mu^t_{1,1}, \ldots, \mu^t_{1,m}, \cdots, \mu^t_{1,M}, \ldots, \\ &\mu^t_{n,1}, \ldots, \mu^t_{n,m}, \ldots, \mu^t_{n,M}, \ldots, \\ &\mu^t_{N,1}, \ldots, \mu^t_{N,m}, \ldots, \mu^t_{N,M}\big]. \end{aligned} \tag{25}$$

Here, $\mu^t_{n,m}$ is the announced unit service price of the BS m for the mAP n at time slot t.

5.1.3 Reward

The reward function of the operator is designed to optimize the reward, which an be calculated as

$$r^{\mathrm{so}}_t = \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} \left(\mu^t_{n,m} Z^t_n[m] c_n - \epsilon \left(f_m\right)^2 Z^t_n[m] c_n\right). \tag{26}$$

Here, $Z^t_n[m]$ is the offloading strategy of mAP n under the accessed BS m at time slot t.

5.1.4 Algorithm design

In the SAC framework, the learning agent (operator) is divided into two distinct entities: the actor (policy) and the critic (value function). The objective of SAC is to determine a price policy $\pi(a|s)$ that maximizes the expected reward. An entropy term $\mathcal{H}(\pi(a|s))$ is incorporated into the reward function to ensure continual exploration. To evaluate policy π, the soft Q-value function is estimated as follows:

$$\begin{aligned} &Q^\pi(s,a) = \\ &\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t + \alpha \sum_{t=1}^{\infty} \gamma^t \mathcal{H}\left(\pi\left(\cdot \mid s_t\right)\right) \mid s_0 = s, a_0 = a\right]. \end{aligned} \tag{27}$$

Here, $\mathcal{H}\left(\pi\left(\cdot|s\right)\right) = \mathbb{E}_a\left[-\log \pi\left(a|s\right)\right]$, where α represents the temperature parameter and γ is the discount factor. The soft Q-value function and the policy can be parameterized as $Q_\theta(s,a)$ and $\pi_\psi(a|s)$ with parameters θ and ψ, respectively, through fully connected deep neural networks (DNNs) (Shen et al., 2020).

Based on the framework outlined above, we devise a SAC-based pricing algorithm, as depicted in

unedited

Algorithm 2 The SAC-based pricing algorithm for the operator

Require: Network environment, evaluation and target critic network parameters, actor parameters, reply memory

Ensure:

The optimal pricing strategy $\mu^*_{n,m}$

for $episode = 1, 2, \ldots,$ **do**
  Initialize environment and state s_0;
  for $t = 1, 2, \ldots, T$, **do**
    Observe the current state: $s_t, \forall n \in \mathcal{N}, m \in \mathcal{M}$;
    Select pricing decision a_t with policy $\pi_\psi(a_t|s_t)$;
    Obtain the immediate reward r_t;
    Observe the next state s_{t+1};
    Store the tuple $\{s_t, a_t, r_t, s_{t+1}\}$ into $\mathcal{M}_r$;
    Randomly sample a mini-batch of tuples from $\mathcal{M}_r$;
    Compute the target Q-value $y^k_{i,t}$ by Eq. (29);
    Update soft Q parameters θ_d by minimizing the loss function in Eq. (28);
    Update the pricing strategy parameters ψ via the gradient in Eq. (32);
    Update temperature parameter α by computing the gradient defined in Eq. (33);
    Update target networks with Eq. (30);
  end for
end for

Algorithm 2. To begin with, it initializes the parameters of the soft Q-value functions and the policy. Subsequently, an experience replay memory $\mathcal{M}_r$ is instantiated. At the onset of each time slot, the operator observes the network state s_t. Based on state s_t and policy $\pi_\psi(a_t|s_t)$, the operator generates action a_t. Following this, feedback reward r_t and the subsequent state s_{t+1} are obtained. The experience $\{s_t, a_t, r_t, s_{t+1}\}$ is then stored in the experience replay memory $\mathcal{M}_r$. By randomly sampling a mini-batch $\mathcal{M}_b$ of tuples $\left\{s^b_t, a^b_t, r^b_t, s^b_{t+1}\right\}$ from $\mathcal{M}_r$, the actor and the critic are updated. For critic updating, two separate critic networks are designed to mitigate overestimation. The parameters θ_d, $\forall d \in \{1, 2\}$, of the evaluation critic networks is updated via minimizing the loss function, which is given by the following:

$$\mathcal{L}(\theta_d) = \mathbb{E}_{\mathcal{M}_b}\left[\frac{1}{2}\left(Q_{\theta_d}(s_t, a_t) - y_t\right)^2\right]. \tag{28}$$

Here, y_t is the target soft value with target parameter $\bar{\theta}_d$, $\forall d \in \{1, 2\}$. It can be calculated as follows:

$$\begin{aligned} y_t =& r_t + \gamma\Big(\min_{d=1,2} Q_{\bar{\theta}_d}(s_{t+1}, a_{t+1}) \\ &- \alpha \log\left(\pi_\psi\left(a_{t+1}|s_{t+1}\right)\right)\Big). \end{aligned} \tag{29}$$

Then, the parameters θ_d, $\forall d \in \{1, 2\}$, is updated by computing the gradients of $\mathcal{L}(\theta_d)$. Moreover, we denote $\tau \in (0, 1)$ as the update factor; the target parameter $\bar{\theta}_d$ is updated as follows:

$$\bar{\theta}_d = \tau\theta_d + (1 - \tau)\bar{\theta}_d, \forall d = 1, 2. \tag{30}$$

For actor updating, the policy parameters of the actor can be learned by minimizing the expected KullbackâĂŞLeibler (KL) divergence:

$$J(\psi) = \mathbb{E}_{\mathcal{M}_b}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi_\psi(\cdot|s_t)\,\middle|\middle|\,\frac{\exp\left(\min_{d=1,2} Q_{\theta_d}(s_t, \cdot)\right)}{Z_{\theta_k}(s_t)}\right)\right]. \tag{31}$$

Here, $Z_{\theta_d}(s_t)$ is an intractable partition function that does not contribute to the new policy's gradient. Therefore, we further transform the above KL-divergence as follows:

$$\begin{aligned} J(\psi) = -\mathbb{E}_{\mathcal{M}_b}\Bigg[\mathop{\mathbb{E}}_{a_{t+1}\sim\pi_\psi}\Big[&\min_{d=1,2} Q_{\bar{\theta}_d}(s_{t+1}, a_{t+1}) \\ &- \alpha \log\left(\pi_\psi\left(a^m_{t+1}|s_{t+1}\right)\right)\Big]\Bigg]. \end{aligned} \tag{32}$$

Then, the parameters ψ of the policy is updated by computing the gradient of $J(\psi)$.

To improve the stability of the training process, the temperature parameter is updated automatically by calculating the gradient of the following objective:

$$J(\alpha) = \mathbb{E}_{\pi_\theta}\left[-\alpha \log\left(\pi_\theta(a_t|s_t)\right) - \alpha\bar{H}\right]. \tag{33}$$

Here, $\bar{H}$ is the value of target entropy.

5.2 Multiagent SAC-based offloading strategy for mAPs

We propose a task-offloading algorithm based on a multiagent SAC algorithm for mAPs. Each mAP determines its offloading strategy based on the communication state between itself and the BSs, as well as the pricing information provided by the operator. Notably, mAPs do not require offloading information about other mAPs.

unedited

5.2.1 State

First, each mAP gets the channel gain between itself and the BSs, along with the price announced by the operator for learning. The state $s_{n,t} \in S$ of mAPs at time slot t is given by

$$s_{n,t} = \big[h^t_{n,1}, \ldots, h^t_{n,m}, \ldots, h^t_{n,M}, \mu^t_{n,1}, \ldots, \mu^t_{n,m}, \ldots, \mu^t_{n,M}\big], \forall n \in \mathcal{N}. \quad (34)$$

5.2.2 Action

Each mAP n obtains the state $s_{n,t}$. Then, in time slot t, based on the actor network, the mAP n outputs its offloading policy as follows:

$$a_{n,t} = \big[Z^t_n[1], \ldots, Z^t_n[m], \ldots, Z^t_n[M]\big]. \quad (35)$$

Here, the offloading policy consists of two parts: how much data to offload and which BS m to access.

5.2.3 Reward

The reward function of mAP n is designed to optimize the reward, which is calculated as follows:

$$r^{\mathrm{mAP}}_{n,t} = \sum_{m \in \mathcal{M}} \gamma^A_n \left(1 - \exp\left(-\alpha_m Z_n[m] x_m\right)\right) - \gamma^D_n \left(\frac{Z^t_n[m]}{R_{n,m}} + \frac{Z^t_n[m] c_n}{f_m}\right) - \mu^t_{n,m} Z^t_n[m] c_n. \quad (36)$$

5.2.4 Algorithm design

In the multi-agent SAC architecture, each learning agent n corresponds to an mAP, comprising actor and critic networks. The objective of each mAP n is to determine an offloading policy $\pi(a_n|s_n)$ that maximizes the expected reward. Similar to Eq. (27), the parameterized soft Q-value function and policy can be represented as $Q_{\theta_n}(s_n, a_n)$ and $\pi_{\psi_n}(a_n|s_n)$ with parameters θ_n and ψ_n, respectively.

The parameters of the networks are updated through network training. The multiagent SAC-based offloading algorithm for mAPs is outlined in *Algorithm 3*. At the outset, it initializes the parameters of the soft Q-value functions and the policy. Additionally, an experience replay memory $\mathcal{M}_{r_n}$ is instantiated. Subsequently, in each time slot t, the mAP observes the network state $s_{n,t}$. Based on state $s_{n,t}$ and policy $\pi_{\psi_n}(a_{n,t}|s_{n,t})$, mAP n generates offloading action $a_{n,t}$. Following this, the feedback reward $r_{n,t}$ and the subsequent state $s_{n,t+1}$ are obtained. The experience $\{s_{n,t}, a_{n,t}, r_{n,t}, s_{n,t+1}\}$ is then stored in the experience replay memory $\mathcal{M}_{r_n}$.

Algorithm 3 The multiagent SAC-based task-offloading algorithm for mAPs

Require: Network environment, evaluation and target critic parameters, actor parameters, reply memory

Ensure:
The optimal offloading strategy $Z^*_n[m]$

```
for episode = 1, 2, ..., do
  Initialize environment and state s_{n,0};
  for t = 1, 2, ..., T, do
    for each mAP n = 1, 2, ..., N do
      Observe the current state: s_{n,t}, ∀n ∈ N;
      Select offloading decision a_{n,t} with policy
        π_{ψ_n}(a_{n,t}|s_{n,t});
      Obtain the immediate reward r_{n,t};
      Observe the next state s_{n,t+1};
      Store the tuple {s_{n,t}, a_{n,t}, r_{n,t}, s_{n,t+1}}
        into M_{r_n};
      Randomly sample a minibatch of tuples
        from M_{r_n};
      Compute target Q-value y_{n,t} by Eq. (38);
      Update soft Q parameter θ_{d_n} by minimiz-
        ing the loss function in Eq. (37);
      Update the offloading policy parameter
        ψ_n via the gradient in Eq. (39);
      Update temperature parameter α_n by
        computing the gradient defined in Eq.
        (40);
      Update target networks;
    end for
  end for
end for
```

The actor and the critic are updated by randomly sampling a minibatch $\mathcal{M}_{b_n}$ of tuples $\{s^b_{n,t}, a^b_{n,t}, r^b_{n,t}, s^b_{n,t+1}\}$ from $\mathcal{M}_{r_n}$. The parameters $\theta_{d_n}, \forall d_n \in \{1, 2\}$, of the evaluation critic networks is updated via minimizing the following loss function.

$$\mathcal{L}(\theta_{d_n}) = \mathbb{E}_{\mathcal{M}_b}\left[\frac{1}{2}\left(Q_{\theta_{d_n}}(s_{n,t}, a_{n,t}) - y_{n,t}\right)^2\right]. \quad (37)$$

Here, $y_{n,t}$ is defined as

$$y_{n,t} = r_{n,t} + \gamma\Big(\min_{d=1,2} Q_{\bar{\theta}_{d_n}}(s_{n,t+1}, a_{n,t+1}) - \alpha_n \log\left(\pi_{\psi_n}(a_{n,t+1}|s_{n,t+1})\right)\Big). \quad (38)$$

unedited

The target parameters $\bar{\theta}_{d_n}$, $\forall d_n = 1, 2$ is updated as $\bar{\theta}_{d_n} = \tau\theta_{d_n} + (1-\tau)\bar{\theta}_{d_n}$, where $\tau_n \in (0, 1)$ is the update factor.

The policy parameters ψ_n of the actor is updated by computing the gradient of $J(\psi_n)$, where $J(\psi_n)$ is given by

$$J(\psi_n) = -\mathbb{E}_{\mathcal{M}_b}\left[\mathop{\mathbb{E}}_{a_{n,t+1}\sim\pi_{\psi_n}}\left[\min_{d_n=1,2} Q_{\bar{\theta}_{d_n}}\left(s_{n,t+1}, a_{n,t+1}\right) - \alpha_n \log\left(\pi_{\psi_n}\left(a_{n,t+1}|s_{n,t+1}\right)\right)\right]\right]. \quad (39)$$

The temperature parameter is updated automatically by calculating the gradient of the following objective:

$$J\left(\alpha_n\right) = \mathbb{E}_{\pi_{\theta_n}}\left[-\alpha_n \log\left(\pi_{\theta_n}\left(a_{n,t}|s_{n,t}\right)\right) - \alpha_n \bar{H}_n\right]. \quad (40)$$

Here, $\bar{H}_n$ is the value of target entropy.

The SAC algorithm's total computational complexity is $\mathcal{O}\left(ET(W_c + W_f)\right)$, where E is the number of episodes and T is the number of steps in each episode; W_c is a convolution layer's computational complexity and W_f is a fully connected layer's computational complexity (Tang et al., 2024). For the proposed multi-agent algorithm containing $N+1$ agents, the total complexity is $\mathcal{O}\left(NET(W_c + W_f)\right)$.

6 Simulation results

In this section, the performances of the proposed DRL-based algorithm for task offloading in 6G native AI networks are evaluated through simulations.

6.1 Simulation settings

The parameters for the experiment are set with reference to previous works (Chen et al., 2023). We consider a 6G native AI network for digital twin consisting of two BSs and four mAPs. In each time slot, the mAPs collect data of different qualities and generate an AI training task. The size of the task is distributed in the range of [0, 5] MB determined by the data quality. The values of the quality parameter xm of the AI model stored in the two BSs are 0.64 and 0.84, and the number of CPU cycles required to process the task is 1.6 and 2.1 Gcycles/MB. The bandwidth of each BS is set as 5 MHz. The computing capability of each core in the BS is 20 Gcycles/s.

For the proposed DRL-based pricing and offloading algorithm, both the leader and the follower models are trained in an iterative manner. The running environment of the DRL is built on the PyTorch framework. To ensure stable performance, the learning rate and the discount factor are respectively set as $l^l = 5e-3$ and $d^l = 0.99$ for the leader; and as $l^f = 3.3e-5$ and $d^f = 0.99$ for the followers. We utilize ReLU as the activation function in the hidden layer and the sigmoid function in the output layer. To calculate the loss function, a minibatch of experience tuples is randomly sampled from 2000 previously experienced memory, and the size of the minibatch is set to 64. All numerical results presented in our work are obtained through multiple experiments to ensure the accuracy of the estimation.

Table 2 List of simulation parameters

Symbol	Value
N	[4, 8]
M	[2, 4]
Z_n	[0, 5] MB
x_m	(0.64, 0.84)
c_n	(1.6, 2.1) Gcycles/MB
W_m	[3, 7] MHz
f_m	[16, 24] Gcycles/s
d^l, d^f	0.99, 0.99
l^l, l^f	5e-3, 3.3e-5
$\mathcal{M}_r$	2000
$\mathcal{M}_b$	64

In order to demonstrate the performance of our scheme, we use the following benchmark algorithms:

- **Twin delayed deep deterministic policy gradient (TD3)-based scheme:** At each time slot, the agents of the leader and the followers learn the policy based on the multiagent TD3 (Zeng et al., 2023) method to maximize its long-term reward;
- **Deep deterministic policy gradient (DDPG)-based scheme:** At each time slot, the agents of the leader and the followers learn the policy based on the multiagent DDPG method to maximize its long-term reward.

6.2 Simulation results

We first verify the validity and feasibility of the proposed scheme. Figs 2 and 3 depict the convergence process of the proposed DRL-based algorithm. The figures show that our proposed scheme

unedited

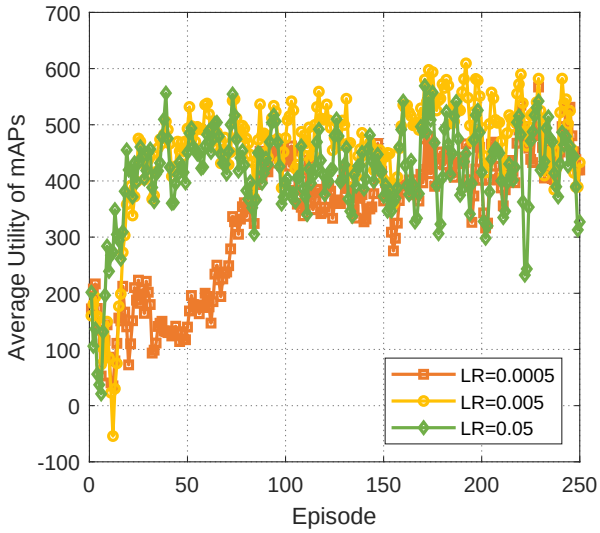


Fig. 2 The convergence of the mAPs

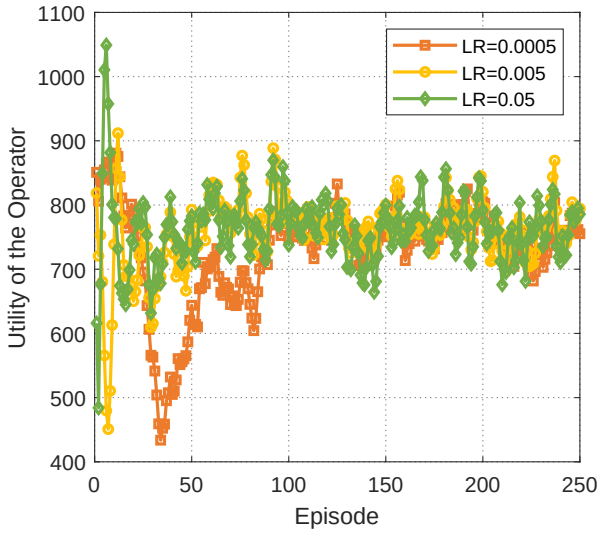


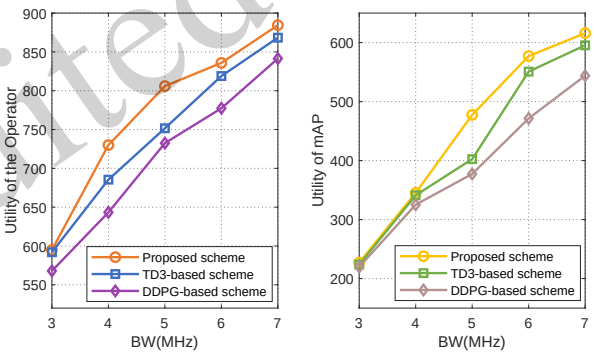
Fig. 3 The convergence of the operator

can obtain convergence after observing and training. Specifically, as shown in Fig. 2, the DRL-based multiagent task-offloading algorithm converges to the stable strategy (i) at about 50 episodes when the learning rates are 0.05 and 0.005 and (ii) at about 90 episodes when the learning rate is 0.0005. In Fig. 3, the DRL-based pricing algorithm converges to stable pricing at the corresponding episode. Thus, parameters such as the learning rate are closely related to the convergence performance of the proposed DRL-based scheme. As can be seen in Figs 2 and 3, a small learning rate leads to a slow convergence rate and a relatively smooth convergence curve, while a fast convergence rate may facilitate its fall into a local optimum. Therefore, it is crucial to set the learning rate to a reasonable value.

Next, we compare the proposed scheme with DDPG/TD3-based scheme to accurately verify our experiment results.

In Fig. 4, we simulate the average utility of different schemes under different bandwidths of BSs. The average utilities of the operator and mAPs increase with bandwidth. A larger bandwidth results in lower latency, increasing the utilities of the mAPs. The mAP is willing to pay more to offload more data and achieve better training results. Correspondingly, leaders have more utility. Compared with the benchmark schemes, the proposed scheme can achieve better utility.

In Fig. 5, we simulate the average utility of different schemes under different computing capacities



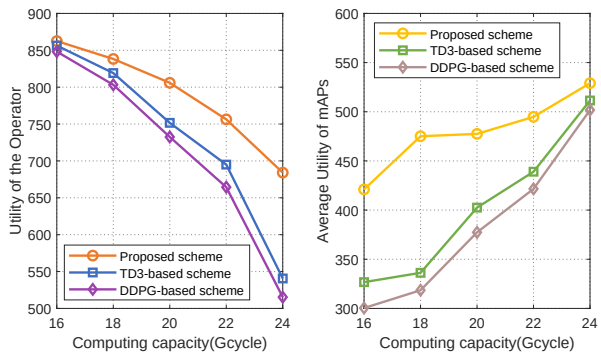
(a) Utility of operator (b) Utility of mAPs

Fig. 4 Impact of the bandwidth on the pricing and offloading performance.

of BSs. The average utility of mAPs increases with computing capacity. A larger computing capacity results in lower latency, increasing the utilities of mAPs. However, the increase in expenditure caused by energy consumption may be higher than the increase in revenue calculated based on data volume. The utility of the operator may thus decrease. The proposed solution is better than the DDPG- and TD3-based schemes.

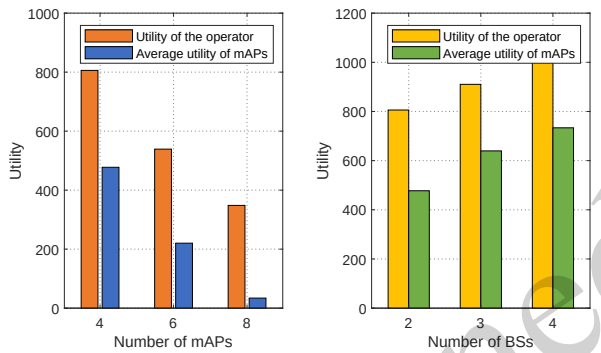
In Fig. 6, we simulate the utility of different numbers of BSs and mAPs. As shown in Fig. 6a, the utilities of the operator and the mAPs decrease with the number of mAPs. The reason is that the resources of the BSs are limited, and more mAPs lead to increased resource competition and interference, resulting in increased task completion delay.

unedited



(a) Utility of operator (b) Utility of mAPs

Fig. 5 Impact of the computing capacity on the pricing and offloading performance.

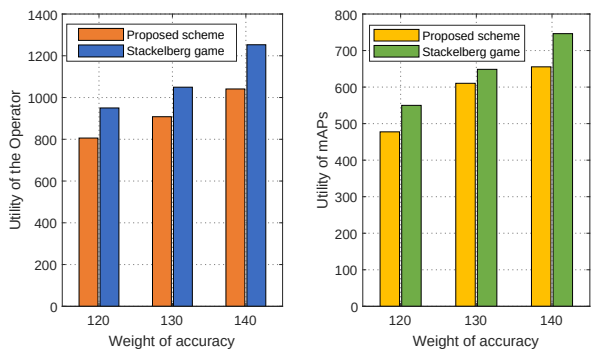


(a) Changes in mAP number (b) Changes in BS number

Fig. 6 Impact of the number of BSs and mAPs on the pricing and offloading performance.

The willingness of mAPs to offload tasks decreases, leading to a decrease in the utility of the operator. In Fig. 6b, the utilities of the operator and the mAPs increase with the number of BSs. As the number of BSs increases, the corresponding resources increase, which can effectively reduce task completion time, and the mAP can choose AI models with higher accuracy to increase the willingness to offload. Therefore, the utility of both increases simultaneously.

Finally, we compare the performance of the proposed scheme and the Stackelberg game scheme in Fig. 7. Since the Stackelberg game scheme is a static scheme, we take the average value of the utility. In Fig. 7, we simulate the average utility for different weights of accuracy. The utility of the operator and the mAPs increases as the weight of accuracy increases, because a higher weight of accuracy will drive the mAP to offoad more data. Compared with the Stackelberg game scheme, the utility of the



(a) Utility of operator (b) Utility of mAPs

Fig. 7 Impact of the weight of accuracy on the pricing and offloading performance.

proposed scheme will decrease. However, after training, the DRL-based algorithm can be deployed offline for inference. The AI model performs inference via forward propagation, which has a linear complexity. The inference complexity of our proposed scheme is smaller than that of the Stackelberg game scheme, and it can quickly adapt to the dynamic changes of the network.

unedited

7 Conclusions

In this paper, we investigated the incentive-based AI training task-offloading process for digital twin in 6G native AI networks. A Stackelberg game was established to model the interaction between the BSs and the mAPs, wherein the operator with BSs was denoted as the leader in determining the price of the AI training service, and the mAPs were the followers who make offloading decisions according to the announced price. We analyzed the SE of the proposed game to obtain equilibrium solutions for the operator and the mAPs. Furthermore, considering the time-varying wireless network environment, we used a DRL algorithm to achieve dynamic pricing and task offloading. Finally, numerical results were presented to verify the feasibility and effectiveness of the proposed scheme.

Contributors

Tianjiao CHEN, and Xiaoyun WANG designed the research. Tianjiao CHEN processed the data. Meihui HUA and Tianjiao CHEN drafted the paper. Tianjiao CHEN, Xiaoyun WANG and Qinqin TANG revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

References

6GANA, 2022. Ten questions of 6G native AI network architecture. *6GANA White Paper*, 1(1).

Alexopoulos K, Nikolakis N, Chryssolouris G, 2020. Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing. *Int J Comput Integr Manuf*, 33(5):429-439. https://doi.org/10.1080/0951192X.2020.1747642

Chen T, Deng J, Tang Q, et al., 2023. Optimization of quality of AI service in 6G native ai wireless networks. *Electronics*, 12(15):3306. https://doi.org/10.3390/electronics12153306

Du J, Jiang C, Benslimane A, et al., 2022. SDN-based resource allocation in edge and cloud computing systems: An evolutionary stackelberg differential game approach. *IEEE/ACM Trans Netw*, 30(4):1613-1628. https://doi.org/10.1109/TNET.2022.3152150

Groshev M, GuimarÃčes C, MartÃŋn-PÃľrez J, et al., 2021. Toward intelligent cyber-physical systems: Digital twin meets artificial intelligence. *IEEE Commun Mag*, 59(8):14-20. https://doi.org/10.1109/MCOM.001.2001237

Haarnoja T, Zhou A, Abbeel P, et al., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. Proc 35th Int Conf on Machine Learning, p.1861-1870.

Hossain AR, Liu W, Ansari N, et al., 2023. AI-native for 6G core network configuration. *IEEE Netw Lett*, 5(4):255-259. https://doi.org/10.1109/LNET.2023.3302833

Institute CMC, 2022. 6G native AI architecture and technologies white paper. *China Mobile White Paper*, .

Jiang L, Zheng H, Tian H, et al., 2022. Cooperative federated learning and model update verification in blockchain-empowered digital twin edge networks. *IEEE Internet Things J*, 9(13):11154-11167. https://doi.org/10.1109/JIOT.2021.3126207

Lin R, Xie T, Luo S, et al., 2022. Energy-efficient computation offloading in collaborative edge computing. *IEEE Internet Things J*, 9(21):21305-21322. https://doi.org/10.1109/JIOT.2022.3179000

Liu G, Deng J, Zheng Q, et al., 2022. Native intelligence for 6G mobile network: technical challenges, architecture and key features. *J China Univ Posts Telecommun*, 29(1):27-40. https://doi.org/10.19682/j.cnki.1005-8885.2022.2004

Liu Q, Tang L, Wu T, et al., 2023. Deep reinforcement learning for resource demand prediction and virtual function network migration in digital twin network. *IEEE Internet Things J*, 10(21):19102-19116. https://doi.org/10.1109/JIOT.2023.3281678

Liu S, Li L, Zhang L, et al., 2024. Game theory based dynamic event-driven service scheduling in cloud manufacturing. *IEEE Trans Autom Sci Eng*, 21(1):618-629. https://doi.org/10.1109/TASE.2022.3226444

Lu Y, Maharjan S, Zhang Y, 2021a. Adaptive edge association for wireless digital twin networks in 6g. *IEEE Internet Things J*, 8(22):16219-16230. https://doi.org/10.1109/JIOT.2021.3098508

Lu Y, Huang X, Zhang K, et al., 2021b. Communication-efficient federated learning for digital twin edge networks in industrial IoT. *IEEE Trans Ind Inf*, 17(8):5709-5718. https://doi.org/10.1109/TII.2020.3010798

Lv Z, Lou R, 2022. Edge-fog-cloud secure storage with deep-learning-assisted digital twins. *IEEE Internet Things Mag*, 5(2):36-40. https://doi.org/10.1109/IOTM.002.2100145

Mihai S, Yaqoob M, Hung DV, et al., 2022. Digital twins: A survey on enabling technologies, challenges, trends and future prospects. *IEEE Commun Surv Tutorials*, 24(4):2255-2291. https://doi.org/10.1109/COMST.2022.3208773

Nguyen DC, Ding M, Pathirana PN, et al., 2022. 6G internet of things: A comprehensive survey. *IEEE Internet Things J*, 9(1):359-383. https://doi.org/10.1109/JIOT.2021.3103320

Nie G, Zhang J, Zhang Y, et al., 2022. A predictive 6G network with environment sensing enhancement: From radio wave propagation perspective. *China Commun*, 19(6):105-122. https://doi.org/10.23919/JCC.2022.06.009

Peng H, Shen X, 2020. Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks. *IEEE Trans Netw Sci Eng*, 7(4):2416-2428. https://doi.org/10.1109/TNSE.2020.2978856

Qi Y, Zhou Y, Liu YF, et al., 2021. Traffic-aware task offloading based on convergence of communication and sensing in vehicular edge computing. *IEEE Internet Things J*, 8(24):17762-17777. https://doi.org/10.1109/JIOT.2021.3083065

Shen X, Gao J, Wu W, et al., 2020. AI-assisted network-slicing based next-generation wireless networks. *IEEE Open J Veh Technol*, 1:45-66. https://doi.org/10.1109/OJVT.2020.2965100

Shi D, Li L, Ohtsuki T, et al., 2022. Make smart decisions faster: Deciding D2D resource allocation via stackelberg game guided multi-agent deep reinforcement learning. *IEEE Trans Mobile Comput*, 21(12):4426-4438. https://doi.org/10.1109/TMC.2021.3085206

Tang Q, Xie R, Yu FR, et al., 2022. Distributed task scheduling in serverless edge computing networks for the internet of things: A learning approach. *IEEE Internet Things J*, 9(20):19634-19648. https://doi.org/10.1109/JIOT.2022.3167417

Tang Q, Xie R, Fang Z, et al., 2024. Joint service deployment and task scheduling for satellite edge computing: A two-timescale hierarchical approach. *IEEE J Sel Areas Commun*, :1-1. https://doi.org/10.1109/JSAC.2024.3365889

Wang CX, You X, Gao X, et al., 2023. On the road to 6G: Visions, requirements, key technologies, and testbeds. *IEEE Commun Surv Tutorials*, 25(2):905-974. https://doi.org/10.1109/COMST.2023.3249835

Wu W, Zhou C, Li M, et al., 2022. AI-native network slicing for 6G networks. *IEEE Wireless Commun*, 29(1):96-103. https://doi.org/10.1109/MWC.001.2100338

unedited

Wu W, Li M, Qu K, et al., 2023. Split learning over wireless networks: Parallel design and resource management. *IEEE J Sel Areas Commun*, 41(4):1051-1066. https://doi.org/10.1109/JSAC.2023.3242704

Xiong Z, Zhang Y, Niyato D, et al., 2019. Deep reinforcement learning for mobile 5G and beyond: Fundamentals, applications, and challenges. *IEEE Veh Technol Mag*, 14(2):44-52. https://doi.org/10.1109/MVT.2019.2903655

Yang Y, Wu J, Chen T, et al., 2024. Task-oriented 6G native-ai network architecture. *IEEE Netw*, 38(1):219-227. https://doi.org/10.1109/MNET.2023.3321464

Yang Z, Chen M, Saad W, et al., 2021. Energy efficient federated learning over wireless communication networks. *IEEE Trans Wireless Commun*, 20(3):1935-1949. https://doi.org/10.1109/TWC.2020.3037554

Yao Z, Xia S, Li Y, et al., 2023. Cooperative task offloading and service caching for digital twin edge networks: A graph attention multi-agent reinforcement learning approach. *IEEE J Sel Areas Commun*, 41(11):3401-3413. https://doi.org/10.1109/JSAC.2023.3310080

Zeng Y, Pou J, Sun C, et al., 2023. Autonomous input voltage sharing control and triple phase shift modulation method for isop-dab converter in dc microgrid: A multiagent deep reinforcement learning-based method. *IEEE Trans Power Electron*, 38(3):2985-3000. https://doi.org/10.1109/TPEL.2022.3218900

Zhang H, Ma X, Liu X, et al., 2023. GNN-based power allocation and user association in digital twin network for the terahertz band. *IEEE J Sel Areas Commun*, 41(10):3111-3121. https://doi.org/10.1109/JSAC.2023.3313192

Zhang J, Lin J, Tang P, et al., 2024. Deterministic ray tracing: A promising approach to thz channel modeling in 6G deployment scenarios. *IEEE Commun Mag*, 62(2):48-54. https://doi.org/10.1109/MCOM.001.2200486

Zhao N, Ye Z, Pei Y, et al., 2022. Multi-agent deep reinforcement learning for task offloading in uav-assisted mobile edge computing. *IEEE Trans Wireless Commun*, 21(9):6949-6960. https://doi.org/10.1109/TWC.2022.3153316

Zhou Y, Liu L, Wang L, et al., 2020. Service-aware 6G:an intelligent and open network based on the convergence of communication,computing and caching. *Digital Commun Netw*, 6(3):253-260. https://doi.org/10.1016/j.dcan.2020.05.003

Zhu X, Luo Y, Liu A, et al., 2022. A deep reinforcement learning-based resource management game in vehicular edge computing. *IEEE Trans Intell Transp Syst*, 23(3):2422-2433. https://doi.org/10.1109/TITS.2021.3114295

unedited