Human-centric Service Offloading with CNN Partitioning in Cloud-Edge Computing-Empowered Industrial Metaverse

Xiaolong Xu, Sizhe Tang, Lianyong Qi, Wanchun Dou, Qiang Ni and Muhammad Bilal

***Abstract*—Industrial Metaverse, an emerging human-centric paradigm of the modern industry, commits to constructing a fully immersive, hyper-spatial, extremely inter-operable virtual space for smart manufacturing. Benefiting from edge computing and 5G networks, operators can interact with industrial elements without manipulating them physically via virtualized tools, e.g., virtual reality and augmented reality. Recently, to construct a more human-centric Industrial Metaverse, massive intelligent services based on deep learning models deployed in edge are booming, covering fields of remote virtual controlling, real-time detection, etc. Convolutional Neural Networks (CNNs) are commonly the basic architecture of the above deployed deep learning models. Seeing that computation and communication resources at the edge are limited, service offloading is applied to ensure the quality of service by exploiting cloud resources. Since current 5G networks can not afford to offload an entire large-scale CNN model to the cloud due to resource constraints, by partitioning a CNN into several sub-models, model partitioning enables the inference to be partially offloaded and executed with parallelism. However, the dynamics of the entire system force the partitioning policy to adapt to the real-time status, otherwise, the partitioning will be counterproductive. Furthermore, the complex action space of the offloading decision makes it hard to determine an efficient and effective offloading policy. To tackle the above challenge, MP-DOB, an offloading optimization strategy is proposed, which consists of three modules. First, the CNN model partitioning module enables the inference to be accelerated in parallel and promotes cloud-edge collaboration. Based on CNN model partitioning, the game theory-based strategy optimizes the offloading decision process with low computational complexity. The load balance module further improves the service assignment in a system-wide view. Eventually, comparative experiments are conducted to indicate the effectiveness of our proposed MP-DOB.**

***Index Terms*—Metaverse, model partitioning, service offloading, game theoretic optimization, intelligent industry.**

I. Introduction

Described as the successor to the mobile Internet[1], the Metaverse will revolutionize the service manner in tremendous areas benefiting from its high integration of immersion and interoperability. Similar to how people navigate the Internet with a mouse, we could access and explore the Metaverse by virtualization technologies, e.g., virtual reality (VR) and augmented reality (AR), mixed reality (MR), and digital twinning (DT)[2]. Leveraging the above technologies to combine physical reality and cyber virtuality, the Metaverse establishes a human-centric multi-user platform where avatars achieve real-time communications and dynamic interaction in 5G networks. To promote the flourishing of Industry 4.0, Metaverse is constructed in industry thus forming Industrial Metaverse, which extends traditional manufacturing into an intelligent and human-centric stage[3]. Encapsulating a plethora of core components and enabling them to be interoperable in the form of virtuality, the Industrial Metaverse serves as a shared human-centric space for avatars to collaboratively, efficiently, and dynamically interact with the industrial elements.

In Industrial Metaverse, large amounts of sensors generate and transmit data concurrently, resulting in unbearable process latency. Therefore, edge computing (EC), a novel paradigm that brings the storage and computation of raw data closer to the data sources, is equipped by Industrial Metaverse to enhance the ability of real-time information processing[4], thus improving the human-centric service quality. Empowered by EC, Industrial Metaverse has the potential to provide better artificial intelligent services, e.g., production prediction and energy management [5]. However, these intelligent services tend to base on large-scale deep learning (DL) models such as convolutional neural networks (CNNs) of massive computation, memory, and energy consumption[6]. Considering the limited resources on the edge, EC shall collaborate with cloud computing (CC) to form a cloud-edge collaborative paradigm where edge servers (ESs) can offload services to the cloud server (CS) via wireless 5G networks, which eases the burden of the edge and improves the quality of service (QoS)[7]. To achieve collaborative inference, DL models are pre-deployed in ESs and CS and the corresponding part of models will be processed according to the offloading policy [8].

Nevertheless, due to the huge volume of DL models, offloading an entire intelligent service to the CS will lead to network congestion, thus sharply decreasing the QoS. To achieve more accurate and efficient service offloading for DL-based intelligent services in a cloud-edge collaborative scenario, neural network model partitioning, a strategy that partitions a model into several parts in the granularity of layer, is leveraged in the inference stage[9]. Applying model partitioning to optimize service offloading strategies in the Industrial Metaverse is extremely appropriate in light of the contemporary plant facility layout and common information handling processes. For recognition and perception-based intelligent services, CNNs are the basic architecture of the applied DL models, and their hierarchical features are very mild with the basic operation of model partitioning. Taking the advantage of model partitioning, the inference of CNNs is

Xiaolong Xu and Sizhe Tang are with School of Software, Nanjing University of Information Science and Technology.

processed in a parallel and dynamic manner, which accelerates the completion of services and optimizes the edge resource utilization. Hence, the QoS is improved and a more human-centric Industrial Metaverse is constructed. However, seeing that multiple ESs requesting offloading lead to the high dynamics of the system, it is a challenge to determine a reasonable offloading policy, including how to partition the inference model, how to decide the offloading choice, and how to balance the ES load.

To tackle the above problems, we design MP-DOB, a distributed, dynamic, and parallel game theoretic service offloading strategy combining CNN partitioning and considering the load balance of ESs. Compared with the existing algorithms, the proposed MP-DOB could achieve a more efficient and energy-saving service offloading decision and better load balance of ESs when the load of some ESs surges and exceeds the affordable range. Besides, the adoption of CNN partitioning makes it feasible to segment an inference in parallelism and further make the most of edge resources. In the architecture of MP-DOB, data collected by sensors are transmitted to the corresponding ESs pre-deployed with service-required DL models. Communicating with CS via wireless channels, ESs could offload part of services to improve the system-wide QoS.

The prime contributions of this paper are as follows:

- Construct a distributed service offloading framework based on cloud-edge collaboration in the EC-enabled Industrial Metaverse.
- Adopt spatial CNN partitioning with smaller granularity in the proposed strategy to achieve the parallel processing of CNN models, which contributes to finding more optimized partial offloading strategies.
- Design a game theoretic service offloading algorithm to jointly optimize the processing delay and energy consumption of edge devices in the Industrial Metaverse.
- Propose an iterative-based load balance algorithm to ease the overload of ESs and improve the stability of the proposed strategy.

The rest of the paper is organized as follows. The related work is reviewed in Section II. Section III illustrates the system model and the optimization problem. The CNN partitioning scheme, game theoretic service offloading algorithm, and the load balance strategy are described in Section IV, V, and VI respectively. The performance analysis is provided in Section VII, followed by the conclusion of this paper in Section VIII.

II. Related Work

Benefiting from the booming development of EC, a series of intelligent applications such as smart monitoring and equipment life prediction are moving into the practice in the Industrial Metaverse [3]. Requiring vast computation resources, it is hard for individual ESs to complete inference. As one of the key technologies in cloud-edge collaborative computing, service offloading has been investigated deeply by many scholars in recent years. Wu *et al.* [10] leveraged a particle swarm optimization algorithm-based strategy to solve the service offloading problem in the cloud-edge collaborative environment, which comprehensively considered the cost of latency and energy. Guo *et al.* [11] proposed a game theory-based offloading method and an approximation offloading method to obtain the optimal solution of cloud-edge collaborative computation offloading problem.

With the increase of informationization, tremendous devices and the corresponding requested services make the cloud-edge network changes highly dynamic and extremely complex, thus increasing the difficulty of making efficient offloading decisions. To tackle such problems, game theory is widely applied to optimize offloading strategy making for the good dynamics and convergence. Chen *et al.* [12] proposed a distributed game theoretic offloading decision making strategy to improve the service experience of mobile users in a multi-channel wireless interference environment. He *et al.* [13] proposed EUAGame, a potential game-based scheme, to efficiently solve a user allocation problem which is used to be NP-hard. After modeling the problem as a potential game, they achieved the Nash equilibrium with the distributed algorithm.

Although an efficient and effective offloading decision strategy can improve the users' experience, the overload of servers is still a key problem. Therefore, leveraging load balancing technique to improve the utility of server resources has become an inevitable trend [14]. Dai *et al.* [15] proposed a joint load balancing and offloading strategy in edge vehicular networks to ease the limit on server performance due to overload. Benefiting from load balancing, the offloading ratio and service performance both gained optimization. Gao *et al.* [16] proposed a load balancing aware service offloading method to cope with computationally intensive services in mobile edge computing environment, aiming to minimizing the processing time of user requests by optimizing the server selection scheme. A SDN-based load balance approach is proposed in paper [17] to maximize resource allocation and processing time of services offloaded to CS. The proposed architecture effectively maintained the real-time load of servers with robustness.

Due to their large sizes, huge DL model-based services still struggle to achieve satisfactory results through existing cloud-edge network offloading strategies. Therefore, model partitioning [9] has been intensively investigated as a key technique to solve this challenge by segmenting a CNN into several parts linearly or spatially. Kang *et al.* [9] proposed a regression-based method to select the most rewarding layer of a CNN as the partition point, thus accelerating the inference and alleviate constrained resources of edge devices. In the proposed strategy, the origin CNN model is partitioned to two parts (head parts and tail parts), which are deployed on the terminal device and ES respectively. Teerapittayanon *et al.* [18] proposed a distributed CNN inference architecture and strategy to accelerate intelligent applications in end-edge-cloud scenario.

However, to our best knowledge, most of the existing research does not consider the scale of inference models and service quantity well. When the amount of requested service is large, the caused high dimension of status and offloading decision leads to the inability of algorithms to be well trained within an affordable time. Additionally, existing model par-

titioning strategies lack good adaptability to the dynamics of the network status and server states. To solve the mentioned problems, we design a dynamic spatial CNN partitioning strategy allowing the parallel collaboration of ESs and CS. Moreover, the optimization of the service offloading policy is modeled as a game, which effectively reduces the original computational complexity resulted by the large decision space.

III. System Model

In this section, we will describe the system model, CNN partitioning model, and service offloading model in the EC-empowered Industrial Metaverse.

In the proposed architecture, we denote the set of ESs deployed in the industrial environment as $\mathcal{E} = \{es_1, es_2, ..., es_E\}$, where each of them has a computing and energy-intensive CNN inference to be completed. The computing force of ESs is denoted as $\mathcal{F} = \{f_1^e, f_2^e, ..., f_E^e\}$. To offload the computation of services to the remote CS, there is a base station for wireless communication. Meanwhile, communication modules are equipped to with ESs and their corresponding wireless signal coverage (WSC) is denoted as $\mathcal{R} = \{r_1, r_2, ..., r_E\}$, through which ESs could access the CS. Note that all communication modules could reach the base station and exchange data. In accordance with existing papers related to edge computing, we adopt a quasi-static requesting manner in this paper to abstract the real-world environments hence get more insightful analysis [19] [20]. In such quasi-static scenario, ESs receive new service requests from industrial devices concurrently when each time slot t_i begins and the states of services as well as ESs remain the same within t_i until the next time slot t_{i+1} comes. Services requested by devices are denoted as $\mathcal{S} = \{s_1, s_2, ..., s_E\}$. Note that at each time slot, large amounts of industry devices generate service requests while only $|\mathcal{E}|$ services are received by ESs with other services waiting. To ensure that all ESs are aware of the status of the current network, the remote CS, and the other ESs, the base station collects the global information and delivers it to every ES at the beginning of each time slot. In this paper, we assume that one ES select only one service at each time slot, which efficiently narrows the solution space for offloading decisions. To demonstrate the offloading process clearly, we will introduce the requested service, communication, and computing models next. Generally, the overview of the architecture of MP-DOB is illustrated in Fig. 1.

A. CNN Partitioning-Based Service Model

We first introduce the model of requested services. Considering the structure characteristics of CNN models, the inference could be executed in a parallel manner and start at any layer [21]. In the first partitioning stage, we partition a CNN model into two parts sequentially, i.e., the head part and the tail part. We define the partitioning layer as parallel demarcation point (PDP). Head parts are processed by ESs locally and the tail parts will be further partitioned and considered to be offloaded. As to the mainstream CNN models, the data quantity of first several layers tends to be large, thus increasing the communication cost once they are chosen as

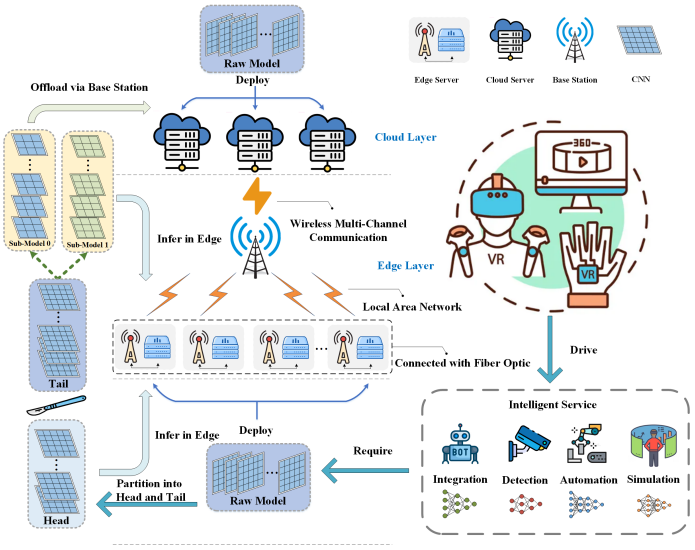

Fig. 1. The Architecture of MP-DOB

PDPs. However, the high inference complexity of latter layers will add the burden to ESs if the inference is not offloaded to the CS. Therefore, after considering the communication cost and computation benefit of each layer, selecting a layer with the highest benefit as PDP can effectively improve the QoS. We denote a CNN model by set $\mathcal{L} = \{l_1, l_2, ..., l_L\}$. To further model the CNN layer, the height, width, kernel size, and channel size of the input layer i are denoted as $H_i^{in}, W_i^{in}, K_i^{in} and C_i^{in}$ respectively. Additionally, the size of output channel of layer l_i is C_i^{out}. Hence, we can obtain the cost and revenue of choosing any layer as the PDP of CNN model in s_i, which contributes to the determination of the most proper PDP. Through these basic parameters, the communication and computing cost will be obtained and the most profitable PDP is selected.

Assuming selecting l_i of a CNN model as the PDP, then we define the computation load, memory load and transmission load of the such case as

$$\chi_i^{com} = \sum_{i=j}^{L} H_i^{in} \times W_i^{in} \times (C_i^{in} \times K_i^2 + 1) \times C_i^{out}, \quad (1)$$

$$\chi_i^{mem} = \sum_{i=j}^{L} H_i^{in} \times W_i^{in} \times (C_i^{in} + C_i^{out}) + C_i^{in} \times C_i^{out}, \quad (2)$$

and

$$\chi_i^{trans} = H_i^{in} \times W_i^{in} \times C_i^{in} \quad (3)$$

respectively. Since the CS can offer more sufficient computing resources, the processing time of the tail part will decrease once it is offloaded to the CS despite the data transferring time, thus the total inference is accelerated. Hence, χ_i^{com}, the total computation from l_i to l_L is used to represent the revenue of inference time with model partitioning. Now that the inference of CNN models always requires a huge amount of memory, the saved memory resources denoted by χ_i^{mem} can also be viewed as the revenue of offloading.

Thus, the revenue and cost of selecting l_i as the PDP are calculated as

$$Re_i = \kappa_1 \chi_i^{com} + \kappa_2 \chi_i^{mem} \tag{4}$$

$$Co_i = \chi_i^{trans} \tag{5}$$

respectively. Therefore, let ϱ_i denote the profit rate of choosing l_i as the PDP. ϱ_i is calculated as

$$\varrho_i = \sqrt{\frac{Re_i}{Co_i}}. \tag{6}$$

It is intuitive that, with a higher ϱ_i, more benefits or earnings are obtained by determining l_i as the PDP.

Based on the generated head part and tail part, the origin service of es_i can be denoted as

$$s_i = \left\{\psi_i^1, \psi_i^2, d_i^{in}, d_i^{med}, d_i^{out}\right\} \tag{7}$$

where ψ_i^1 and ψ_i^2 denote the computation of the head and tail part, d_i^{in} and d_i^{out} denote the input and output of the entire origin model, and d_i^{med} is the transmitted data between the two parts. Considering the structure of CNN, the model could be further spatially partitioned into two sub-models, which leads to the potential of parallel inference. We use the segmentation rate δ to represent the percentage of the computation offloaded to the CS as the entire tail part. The determination of δ is introduced in Section IV.

B. Communication Model

In this paper, the 5G base station controls the up-link and down-link data transmission of ESs in the factory. There exists $\mathcal{M} = \{m_1, m_2, ..., m_M\}$ shared by ESs to communicate with the CS. At the stage of service offloading, the decision set of es_i is $\theta_i = \{0\} \cup \{\mathcal{M}\}$, where $\{0\}$ means that the ES processes the entire tail part locally. Specifically, if $\theta_i \neq 0$ then es_i will offload partial inference through m_{θ_i}. The systematic communication efficiency is influenced by the global offloading decision set $\Theta = \{\theta_1, \theta_2, ..., \theta_E\}$. By means of Θ, the data transmission rate of es_i could be calculated as

$$\lambda_i(\Theta) = Blog_2\left(1 + \frac{p_i g_i}{\sigma_0 + \sum\limits_{es_k \in \mathcal{E}\setminus\{es_i\}:\theta_k=\theta_i} p_k g_k}\right) \tag{8}$$

where B is the channel bandwidth and σ_0 is the background Gaussian noise power. Moreover, p_i and g_i denote the transmission signal power and the channel gain of es_i respectively. According to the above communication model, if there are excessive ESs that choose the same channel simultaneously, the incurred interference may lead to the deterioration in service performance. To achieve high QoS, the communication efficiency is optimized as well in the MP-DOB.

C. Computation Model

Upon collecting the service requests and ES statuses, the base station will deliver these information to ESs for choosing the computing mode and deciding the offloading channel distributedly. There are two feasible computing diagrams: edge-only computing and cloud-edge collaborative computing. We will introduce such two modes in detail.

1) Edge-Only Mode: In such computing paradigm, ESs receive the service requested from the industrial devices and complete the entire inference. Thus, the process latency is only composed of the computing delay. Let f_i^e denotes the computation capability of es_i. Since the inference of both the head and entire tail part is completed on the ES, the computing delay is calculated as

$$t_i^e = \frac{\psi_i^1 + \psi_i^2}{f_i^e}. \tag{9}$$

The computational energy is denoted as

$$e_i^e = \epsilon_i^e(\psi_i^1 + \psi_i^2) \tag{10}$$

where ϵ_i^e is the coefficient representing the required energy per FLOPs for es_i. We use Γ_i^e to denote the total overhead of edge-only computing mode, which is calculated as

$$\Gamma_i^e = \alpha_i t_i^e + \beta_i e_i^e \tag{11}$$

where α_i and β_i are the weight parameters of the execution time and energy consumption of s_i respectively. Note that the sum of α_i and β_i is 1 and they are both positive. The weight parameters are introduced to improve the response flexibility of the offloading decision model in the face of different service preferences. That is, if the user has a high requirement for low latency of the service, then the value of α_i shall increase accordingly; similarly, if the ES is more sensitive to energy consumption, then the value of β_i is higher to consider more about the impact of energy consumption on the offloading decision.

Connected via fiber optics, ESs have the ability to communicate with each other, which enables them to collaborate to chase for the higher system-wide QoS. If an ES transfers data of the input or intermediate result to the nearest ES, the communication delay generated by such one-hop cable transmission is negligible. Thus, once a certain ES is overload and the corresponding QoS can not be guaranteed, it can offload part of the service (specifically, the head part of service) to another ES for load balance.

2) Cloud-Edge Collaborative Computing Mode: In the cloud-edge collaborative paradigm, the head part is processed on ES and the tail part is partially offloaded to CS via wireless networks for acceleration. To achieve the consistency of co-inference, there are intermediate data needed to be transferred from the ES to the CS. Therefore, the total latency consists of three parts: the computing delay on edge, transmission delay, and computing delay on the cloud. Meanwhile, the transmission energy adds the inference cost and server load as well.

The transmission delay could be calculated as

$$t_{i,tran}^c = \frac{d_i^{med}}{\lambda_i(\Theta)} \tag{12}$$

where d_i^{med} is the intermediate data. The execution time in the ES and the CS is denoted as

$$t_{i,exe1}^c = \frac{\psi_i^1 + (1-\delta)\psi_i^2}{f_i^e} \tag{13}$$

$$t^c_{i,exe2} = \frac{\delta \psi_i^2}{f_i^c} \tag{14}$$

respectively, where f_i^c is the computation resources assigned to the tail part of s_i and δ is the segmentation rate of the tail part. Note that offloaded services processed simultaneously share the computation capability of the CS equally.

The consumed energy of cloud-edge offloading is composed of three parts as well: the computing energy of the local ES, communication energy and computing energy of the CS, which is denoted as

$$e_i^c = \frac{p_i d_i^{med}}{\lambda_i(\Theta)} + \epsilon_i^e[\psi_i^1 + (1-\delta)\psi_i^2] + \epsilon_i^c \delta \psi_i^2 + \zeta_i \tag{15}$$

where ζ_i is extra energy for holding the channel after delivering the data [22]. Analogous to ϵ_i^e, ϵ_i^c is the coefficient denoting the energy required per FLOPs for the CS.

Therefore, the total overhead of the cloud-edge collaborative mode considering both delay and energy can be calculated as

$$\Gamma_i^c = \alpha_i(t^c_{i,exe1} + t^c_{i,exe2} + t^c_{i,tran}) + \beta_i e_i^c. \tag{16}$$

According to studies [23] [24], the transmission delay and energy consumption generated by the feedback such as inference results are neglected since the scale of them is too small in the aspect of our scenario.

D. Problem Definition

In such multi-user service offloading system, the goal is to maximize the system-wide total QoS through an optimal offloading policy $\Theta = \{\theta_1, \theta_2, ..., \theta_E\}$ at each time slot. On the basis of models mentioned above, the service offloading problem in Industrial Metaverse is formulated as

$$\max_{\Theta} \quad \sum_{i=1}^{E} Q_i(\theta_i, \theta_{-i}) \tag{17}$$

$$s.t. \quad \forall es_i \in \mathcal{E}, \theta_i \in \{0\} \cup \mathcal{M}, \quad \forall es_i \in \mathcal{E}, \Gamma_i^c \leq \Gamma_i^e. \tag{18}$$

In target function (17), θ_{-i} refers to the offloading decision of other ESs, i.e., $\theta_{-i} = \{\theta_1, \theta_2, ..., \theta_{i-1}, \theta_{i+1}, ..., \theta_E\}$. $Q_i(\theta_i, \theta_{-i})$ is defined as the QoS value of s_i, which is be calculated as

$$Q_i(\theta_i, \theta_{-i}) = \begin{cases} 1 - log_2\left(1 + \frac{\Gamma_i^e}{\varpi}\right), if \theta_i = 0, \\ 1 - log_2\left(1 + \frac{\Gamma_i^c}{\varpi}\right), if \theta_i \in \mathcal{M}. \end{cases} \tag{19}$$

where φ is the threshold to achieve the normalization of QoS. Inequality (18) means that the offloading choices are finite and the offloading must be profitable. As the processing overhead comprehensively considering latency and energy increases, the corresponding QoS will decrease accordingly. Therefore, the goal of MP-DOB is to achieve the system-wide QoS maximization.

IV. Game-theoretic Service Offloading with CNN Model Partitioning and Load Balance

In this section, MP-DOB is designed for service offloading of intelligent services in Industrial Metaverse. First, the algorithm for optimizing the determination of PDP and segmentation rate is introduced, which is the basis for partial service offloading. Then, a game theory-based approach to optimizing the service offloading policy is described in detail. We further propose a load balancing strategy to ease the network congestion and service overload. The design of game theoretic offloading with CNN partition is shown in Fig. 2.

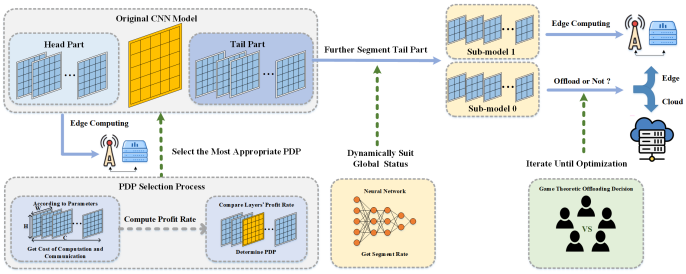

Fig. 2. The Design of Game Theoretic Offloading with CNN Partition

A. CNN Model Partitioning

Based on the service model defined in Section III, we can use the indicator ϱ_i to measure the appropriateness of choosing l_i as the PDP. Thus, the goal of the PDP selecting algorithm is to find the layer with maximum ϱ_i. The service information consists of input data, maximum tolerable latency and requesting mode type. Based on the received information, ESs can work out the best PDP since they could access the required parameters of the pre-deployed models.

According to (4) and (6), the value of κ_1 and κ_2 influences the calculation of ϱ_i. If these two parameters remain stable, ESs can pre-calculate the ϱ_i of each layer of every pre-deployed models, thus saving the computing time of PDP selecting algorithm. However, since the service requests of industrial devices and the computing as well as memory capacity of ESs are dynamically changing, κ_1 and κ_2 also fluctuate to suit different needs. For instance, bearing heavy workload, the ES can not afford too much energy, then it increases the value of κ_1 to reduce the tendency to handle high energy-consuming services. Similarly, if there are little memory in the ES, it will call for higher κ_2. Therefore, at each time slot, every ES shall recalculate the ϱ_i according to the current status of the service request and all servers. That is, ESs need to work out the ϱ_i of each layer of requested models and sort them, thus selecting the layer with maximum ϱ_i as the PDP under the current status.

Once the PDP is selected, the optimal value of segmentation rate is supposed to be obtained, thus determining the partitioning policy of tail parts. Dynamically changing with the real-time status of the global environment especially the available computing power of CS and the computing capability of the ES, we leverage a simple neural network to gain the best δ at each time slot. Specifically, the network status, current

global ES states, CNN model parameters are the input of the neural network, and then an optimal δ is predicted as the segmentation rate. Since the neural network for prediction the δ is shallow, the execution time could be ignored in the view of the whole offloading decision-making process. Hence, at each time slot, an optimal model partitioning policy is made dynamically, which is the basis of the latter game theoretic offloading decision. The pseudo code of model partitioning is shown in Algorithm 1.

Algorithm 1 CNN Model Partitioning Strategy

Input: Information of the CNN model $\mathcal{L}$, computing force $\mathcal{F}$, network status Sta,
Output: The optimal PDP l^* and segmentation rate δ^*

```
for each es_i in parallel do
  for each time slot t_i do
    for each l_j ∈ L do
      Compute Re_i and Co_i of l_i,
      Obtain ϱ_i of l_j.
    end for
    Sort ϱ_i of each layer and select l* with the maximum ϱ as current PDP.
    Partition the original model into the head and tail parts.
    Get the dynamic segmentation rate δ* via neural network according to the current global status.
    Update channel status and ES states.
  end for
end for
```

B. Game-theoretic Service Offloading Optimization

After determining the PDP and segmentation rate of each CNN model, the issue of optimizing the tail parts computation offloading in the Industrial Metaverse scenario needs to be considered. Since all head parts are assigned to be processed on ESs, we only have to consider whether offload and how to offload the sub-models of tail parts.

From the communication model and offloading modes mentioned in Section III, it is obvious that if multiple sub-models share a certain channel simultaneously, the generated wireless interference will result in the decrease of data transmission rate. Thus, the total processing delay will increase due to the low data rate. Meanwhile, the energy consumption produced by transmission will become higher, which worsens the problem of resource scarcity in the edge network. Therefore, the system-wide QoS will decrease accordingly and can not satisfy the service requests. Once the QoS of offloading mode decreases till even lower than the one of edge-only computing mode, the ES will not choose to offload the tail part to the CS. In such case, we claim the offloading decision is unprofitable. That is, in the current network scenario, if es_i chooses to offload the segmented sub-model of s_i from es_i to the CS under the policy $\theta_i \neq 0$ with higher QoS than the edge-only computing mode, the offloading decision is defined as profitable.

It is of great significance to further discuss the definition of profitable offloading. From the perspective of an individual service, chasing profitable offloading policy means they will not bear the loss of QoS due to the offloading, which makes sense to individuals in a multi-user system. Therefore, the system-wide QoS can also obtain positive growth, which leads to the higher utilization of network resources and the computing capability of the CS. Compared with conventional manners where each ES can achieve offloading via all accessible channels, we only consider the profitable offloading decisions so that the decision space is compressed, thus reducing the computation complexity significantly. The discussion of the complexity of offloading decisions in MP-DOB will be expanded below.

1) Game Formulation: According to paper [12] [13], the centralized optimization of offloading scheduling is NP-hard since the solution is to be found in the multiple dimension space, i.e., $\{0, 1, 2, ...M\}^E$. Inspired by the definition of profitable computation offloading, we apply game theory to system-widely optimize the offloading strategy by abstracting each ES as a player (i.e., intelligent agent) in the game[25]. These players are self-organized in the process of strategy-making by their intelligence and self-interested traits. Thus, the primary NP-hard centralized optimization problem is solved in a distributed way. In the framework of game theory, each player interacts and competes with each other, chasing for the maximum individual QoS according to the competition rule. Hence, the optimal offloading policy will be determined when no player wants to change its offloading decision.

Then, we will formulate the distributed offloading decision game. Once receiving the global status θ_{-i} from the base station, all the ESs can determine a most profitable offloading policy θ_i, aiming at maximizing the QoS. Now, we can construct a game to formulate the distributed offloading decision problem as

$$\Xi =< \mathcal{E}, \{\mathcal{A}_i\}_{es_i \in \mathcal{E}}, \{Q_i\}_{es_i \in \mathcal{E}} > . \tag{20}$$

Specifically, $\mathcal{E}$ is the set of players composed of ESs, $\mathcal{A}_i$ is the set of strategies of player i, and Q_i denotes the corresponding QoS value, which is to be maximized by player i. In the rest of this paper, we use Ξ to represent the distributed offloading decision game.

In the framework of MP-DOB, if we want to achieve the maximization of system-wide QoS, we have to determine an offloading policy where no player can further increase the profit by updating its offloading decision unilaterally. Based on the game theory, the process of searching for the optimal offloading policy can be transformed into the determination of a Nash equilibrium.

Definition 1: Given a offloading policy set $\Theta^{'} = (\theta_1^{'}, \theta_2^{'}, ..., \theta_E^{'})$, if there is a set $\Theta^{'}$ making

$$Q_i(\theta_i^{'}, \theta_{-i}^{'}) \geq Q_i(\theta_i, \theta_{-i}^{'}), \forall \theta_i \in \mathcal{A}_i, es_i \in \mathcal{E}, \tag{21}$$

true, then the set $\Theta^{'}$ is the Nash equilibrium of the distributed offloading decision game.

According to Definition 1, no player could further improve the QoS when other players hold the offloading decision unchanged at the Nash equilibrium. After numbers of iteration, the system-wide QoS maximization is obtained if each player

would not change the offloading decision when they achieve the Nash equilibrium.

In the duration of offloading, the channel interference caused by the fact that multiple players share the same channel simultaneously is the main reason for the network congestion and high transmission latency. Intuitively, if a certain channel is shared by excessive players and the corresponding interference exceeds a threshold, the offloading via this channel would be not profitable. Next, we give the mathematical expression of the above case.

Lemma 1: Provided an offloading policy set Θ, the player i could achieve profitable computation offloading if the network interference $\gamma_i(\Theta) = \sum_{es_k \in \mathcal{E} \setminus \{es_i\}: \theta_i = \theta_k} p_k g_k$ of channel θ_i satisfies that $\gamma_i(\Theta) \leq \xi_i$, with the threshold

$$\xi_i = \frac{p_i g_i}{2^{\frac{d_i^{med}(\alpha_i + \beta_i p_i)}{\alpha_i \delta \psi_i^2 (\frac{1}{f_i^e} - \frac{1}{f_i^c}) + \beta_i (\epsilon_i^e \delta \psi_i^2 - \epsilon_i^c \psi_i^2 - \zeta_i)}} - 1} - \sigma_0. \tag{22}$$

Proof 1: According to (11), (16), (19), and Definition 1, the condition $Q_i(\theta_i \in \mathcal{M}, \theta_{-i}) \geq Q_i(\theta_i = 0, \theta_{-i})$ is equivalent to

$$\alpha_i t_i^e + \beta_i e_i^e \geq \alpha_i (t_{i,exe1}^c + t_{i,exe2}^c + t_{i,tran}^c) + \beta_i e_i^c. \tag{23}$$

Then, we substitute (9), (10), (12), (13), (14), and (15) into the inequality respectively. The inequality could be represented as

$$\begin{aligned} &\alpha_i \frac{\psi_i^1 + \psi_i^2}{f_i^e} + \beta_i \epsilon_i^e \left(\psi_i^1 + \psi_i^2\right) \geq \\ &\alpha_i \left(\frac{\psi_i^1 + (1-\delta)\psi_i^2}{f_i^e} + \frac{\delta \psi_i^2}{f_i^c} + \frac{d_i^{med}}{\lambda_i(\Theta)} \right) + \\ &\beta_i \left\{ \frac{p_i d_i^{med}}{\lambda_i(\Theta)} + \epsilon_i^e [\psi_i^1 + (1-\delta)\psi_i^2] + \epsilon_i^c \delta \psi_i^2 + \zeta_i \right\}. \end{aligned} \tag{24}$$

That is,

$$\alpha_i \delta \psi_i^2 \frac{f_i^c - f_i^e}{f_i^e f_i^c} + \beta_i \left(\epsilon_i^e \delta \psi_i^2 - \epsilon_i^c \delta \psi_i^2 - \zeta_i\right) \geq \frac{d_i^{med}(\alpha_i + \beta_i p_i)}{\lambda_i(\Theta)}. \tag{25}$$

To further simplify the inequality, we move $\lambda_i(\Theta)$ to the left side alone,

$$\lambda_i(\Theta) \geq \frac{d_i^{med}(\alpha_i + \beta_i p_i)}{\alpha_i \delta \psi_i^2 (\frac{1}{f_i^e} - \frac{1}{f_i^c}) + \beta_i (\epsilon_i^e \delta \psi_i^2 - \epsilon_i^c \delta \psi_i^2 - \zeta_i)}. \tag{26}$$

By expanding out $\lambda_i(\Theta)$ by (16), the above inequality turns out to be

$$\sum_{es_k \in \mathcal{E} \setminus \{es_i\}: \theta_i = \theta_k} p_k g_k \leq \frac{p_i g_i}{2^{\frac{d_i^{med}(\alpha_i + \beta_i p_i)}{\alpha_i \delta \psi_i^2 (\frac{1}{f_i^e} - \frac{1}{f_i^c}) + \beta_i (\epsilon_i^e \delta \psi_i^2 - \epsilon_i^c \delta \psi_i^2 - \zeta_i)}} - 1} - \sigma_0. \tag{27}$$

Let

$$\gamma_i(\Theta) = \sum_{es_k \in \mathcal{E} \setminus \{es_i\}: \theta_i = \theta_k} p_k g_k \tag{28}$$

and

$$\xi_i = \frac{p_i g_i}{2^{\frac{d_i^{med}(\alpha_i + \beta_i p_i)}{\alpha_i \delta \psi_i^2 (\frac{1}{f_i^e} - \frac{1}{f_i^c}) + \beta_i (\epsilon_i^e \delta \psi_i^2 - \epsilon_i^c \delta \psi_i^2 - \zeta_i)}} - 1} - \sigma_0. \tag{29}$$

Then, inequality (27) is equivalent to $\gamma_i(\Theta) \leq \xi_i$. Hence, the proof of Lemma 1 is done.

According to Lemma 1, when the network interference of player i is low to a certain threshold, it is feasible for the player to adopt the computation offloading mode and the profit can be achieved. Otherwise, the service shall be processed on the ES.

2) Structural Properties: Next, we will leverage a powerful tool, potential game, to study the Nash equilibrium of the above distributed service offloading game. First, we give the definition of the potential game.

Definition 2: If a game satisfies the function $\Psi(\Theta)$ such that for every $es_i \in \mathcal{E}$, $\theta_{-i} \in \Pi_{k \neq i} \mathcal{A}_k$ and $\theta_i^{'}, \theta_i \in \mathcal{A}_i$, if

$$Q_i(\theta_i^{'}, \theta_{-i}) \geq Q_i(\theta_i, \theta_{-i}), \tag{30}$$

we have

$$\Psi_i(\theta_i^{'}, \theta_{-i}) \geq \Psi_i(\theta_i, \theta_{-i}), \tag{31}$$

it is a potential game.

Based on Definition 2, it is feasible for players to achieve the maximum revenues within finite improvement steps, which fits well with our distributed offload decision for asynchronous update operations. Meanwhile, the system-wide average QoS is also optimized considering the interests of each game player. Furthermore, a Nash equilibrium always exists in the potential game [26] such that the finite decision changes will result in a Nash equilibrium. Before proving that the distributed computation offloading game is a potential game and there exists the Nash equilibrium, we need to construct the potential function as

$$\begin{aligned} \Psi(\Theta) = &- \sum_{i \in \mathcal{N}} p_i g_i \gamma_i(\theta_i, \theta_{-i}) \Lambda[\theta_i > 0] \\ &- \sum_{i \in \mathcal{N}} p_i g_i \xi_i \Lambda[\theta_i = 0] \end{aligned} \tag{32}$$

where $\Lambda[\varsigma]$ is a judgement function. If the condition variable ς is true, the value of $\Lambda[\varsigma]$ is 1, otherwise the function value is 0. The value of the wireless interference threshold ξ_i could be obtained by (29).

Lemma 2: The distributed offloading computing game in this paper shares the same property as the potential game and could be transformed to the latter.

Proof 2: To prove Lemma 2, we partition the decision update operation into three cases. These three cases correspond to three decision changing conditions respectively.

Based on the constructed potential function above, the value of individual function to the player i can be denoted as

$$\begin{aligned} \Psi(\theta_i, \theta_{-i}) = &- p_i g_i \gamma_i(\theta_i, \theta_{-i}) \Lambda[\theta_i > 0] \\ &- p_i g_i \xi_i \Lambda[\theta_i = 0] \end{aligned} \tag{33}$$

Case 1: $\theta_i > 0$, $\theta_i^{'} > 0$. According to Lemma 1 and the definition of profitable distributed offloading game, $\gamma_i \leq \xi_i$ under such condition. Therefore,

$$\begin{aligned} &\Psi(\theta_i^{'}, \theta_{-i}) - \Psi(\theta_i, \theta_{-i}) \\ &= - p_i g_i \gamma_i(\theta_i^{'}, \theta_{-i}) + p_i g_i \gamma_i(\theta_i, \theta_{-i}) > 0. \end{aligned} \tag{34}$$

Case 2: $\theta_i = 0$, $\theta_i^{'} > 0$.

$$\begin{aligned} &\Psi(\theta_i^{'}, \theta_{-i}) - \Psi(\theta_i, \theta_{-i}) \\ &= - p_i g_i \gamma_i(\theta_i^{'}, \theta_{-i}) + p_i g_i \xi_i > 0 \end{aligned} \tag{35}$$

Case 3: $\theta_i > 0$, $\theta_i^{'} = 0$.

$$\begin{aligned} &\Psi(\theta_i^{'}, \theta_{-i}) - \Psi(\theta_i, \theta_{-i}) \\ &= - p_i g_i \xi_i + p_i g_i \gamma_i(\theta_i, \theta_{-i}) > 0 \end{aligned} \tag{36}$$

Algorithm 2 Game-based distributed service offloading

Input: ES set $\mathcal{E}$, computing force set $\mathcal{F}$, channel set $\mathcal{M}$, service requested set $\mathcal{S}$, and the CS

Output: Optimal offloading decision set Θ^*, Highest system-wide QoS Q^*

1: **for** each ES $\in \mathcal{E}$ in time slot $t = 0$ **do**
2: es_i chooses the edge-only computation mode and sets the offloading decision as $\theta_i(0) = 0$.
3: **end for**
4: **repeat**
5: **for** each ES $\in \mathcal{E}$ in time slot t with parallelism **do**
6: The base station **collects** the relevant information from all ESs in the system.
7: The base station **delivers** the states of all monitored channels to all the ESs.
8: es_i **receives** the signal power of each channel from the base station.
9: es_i **computes** the most profitable service offloading decision update $\Upsilon_i(t)$.
10: es_i sends the expected decision change $\Upsilon_i(t)$ to the base station and **competes** for the chance of updating its decision.
11: **end for**
12: The base station **chooses** the best update from $\Upsilon(t) = \{\Upsilon_1(t), \Upsilon_2(t), ..., \Upsilon_N(t)\}$.
13: The base station **notifies** all the ESs of the current offloading decision set.
14: **for** each ES $i \in \mathcal{V}$ **do**
15: **if** $\Upsilon_i(t)$ is the best update in time slot t **then**
16: es_i **updates** the corresponding offloading decision, i.e., $\theta_i(t+1) = \Upsilon_i(t)$.
17: **else**
18: es_i **remains** the same offloading decision as the last time slot, i.e., $\theta_i(t+1) = \theta_i(t)$.
19: **end if**
20: **end for**
21: **until** No ES changes its offloading decision and the base station sends END information.
22: The base station obtains the final optimal offloading decision set Θ^* and the calculated corresponding system-wide QoS Q^*.

By formulating the distributed service offloading game and leveraging its asynchronous update of optimization, we design a game theory-based distributed service offloading algorithm, which is given in detail in Algorithm 2. The novelty of our proposed offload strategy lies in its excellent use of the asynchronous update of all players in the game, that is, it improves the traditional centralized decision-making method with high complexity into a distributed manner. From line 5 to line 11, each player calculates the best offloading decision under the current network state and contends for the update opportunity. The base station shall choose the best update decision from the requests as the final update, which leads to the maximum increase of the system-wide QoS. From line 14 to line 19, players change or hold on their offloading decisions in accordance with the order of the base station. The algorithm repeats the above process until the Nash equilibrium of the distributed computation offloading game is reached, that is, there is no better offloading policy leading to a higher system-wide QoS.

We then conduct the computational complexity analysis of our proposed distributed offloading strategy. Due to the parallelism of our strategy, each ES can make the update decision simultaneously. Thus, the main computational complexity lies in the sorting operation, which is produced by the process of choosing the final update decision that leads to the maximum system-wide QoS. Thus, the computational complexity of distributed offloading decision is $\mathcal{O}(T|\mathcal{E}|log|\mathcal{E}|)$, where T is the time slots required for convergence. Due to the adaptability of our service offloading algorithm, the size of time slots tends to be considered a constant.

C. Load Balance

Benefiting from the above game theory-based service offloading strategy, the utilization of network and cloud resources as well as the scheduling of tail parts of models are optimized. However, the potential of edge collaboration has not been exploited since such game strategy focuses on the competition relationship among ESs while ignoring the cooperation value. Therefore, although the cloud-edge collaboration computing mode has alleviated much of the overload in edge, it is still a problem because the resources of ESs are not allocated well. Hence, a load balance module is added in our proposed MP-DOB, which is designed to further optimize the service assignment in the view of edge system.

The load balance module is applied to re-schedule the inference of head parts when CNN model partitioning has finished. Considering the inference of head parts is independent of the segmentation and offloading of tail parts, the load balance can be conducted at the same time as the offloading decision-making process of tail parts. Such parallel optimization commits to the efficiency of MP-DOB and improves the effectiveness in realistic scenarios.

To reduce the impact of ES communication on total service latency, each ES is considered to offload services only to the ES closest to it and directly connected to it. According to existing papers such as [27], the communication in edge via fiber optics can be neglected, so the influencing factors of edge collaboration policy are available resources of ESs, states of head parts. Moreover, through limiting offloading targets, the decision set is degraded from $\mathcal{E}^{|E|}$ to $\{0,1\}^{|E|}$, which narrows the solution space to a great extent. The offloading decision is 1 means that the ES chooses to offload the head-part inference to the nearest ES while 0 means that no offloading. Considering the scale of solution space, we leverage the deep neural network (DNN) with the states of $\mathcal{E}$, $\mathcal{F}$, and $\mathcal{S}$ as the input to obtain the optimal offloading policy. Seeing that the base station is responsible to collecting the global status

and the applied DNN is not complex, it is suitable to deploy the DNN for offloading policy-making in the base station. In particular, the base station inputs the collected status into the DNN and delivers the generated optimal policy to each ES. Algorithm 3 demonstrates the process of load balance module in detail.

Algorithm 3 Load Balance in Edge

Input: ES set $\mathcal{E}$, computing force set $\mathcal{F}$, service set $\mathcal{S}$
Output: Load balance-aimed ES scheduling policy Ω and system-wide QoS improvement

for each ES $\in \mathcal{E}$ in a time slot parallelly **do**
  es_i **determines** the best PDP of s_i according to Algorithm 1 and **sends** it to the base station.
end for
The base station **collects** the PDP selection of all services and the global status.
The base station **obtains** the optimal offloading policy Ω with pre-deployed DNN.
for each ES $\in \mathcal{E}$ in a time slot parallelly **do**
  es_i **receives** the offloading policy set Ω from the base station.
  if $\omega_i = 0$ **then**
    es_i **completes** the head part of s_i.
  else if $\omega_i = 1$ **then**
    es_i **transfers** the input data of s_i via fiber optics to the nearest ES and **offloads** the head part of s_i there.
    es_i **receives** the intermediate inference result of s_i.
  end if
  es_i **calculates** the increase of QoS compared with that gained without load balance module.
end for

V. Experimental Evaluation

In this section, we will conduct comparative experiments and present the numerical analysis to demonstrate the efficiency of the proposed MP-DOB. The performance metrics to quantify the benefits of our offloading and load balance strategy is QoS defined in the Section III. Furthermore, the system-wide QoS or global QoS, equal to the sum of QoS of all users in such game system, is leveraged to measure the overall level of service satisfaction in a system-wide perspective.

A. Experiment Setup

The experiments are run on Apple M1 Pro CPU, at 3.2 GHz, with 16 GB of RAM. To simulate the realistic scenario of Industrial Metaverse, we set the scale of ESs as 9 with 6-TFLOPs computing power, which are scattered uniformly in the industry. The CS with 14-TFLOPs computing power is assumed to be placed 3 km away from the industry, connected with the base station located in the industry center. The transmission power q_i is set as 150 mWatts and the background noise σ_0 equals to -100 dBm. Moreover, we assume that the there are 15 channels and the bandwidth is 15 MHz. Since image process-based intelligent applications are common in Industrial Metaverse, we select YOLO [28] as the representative of inference models, with the input data set in the interval [2000 KB, 3000 KB]. The value of ϵ_i^e and ϵ_i^c vary from 0.8 to 1.2.

B. Comparative Offloading Strategies

We mainly conduct comparative experiments in two aspects, i.e., 1) comparing the decision optimization effect of MP-DOB with other offloading strategies and 2) comparing MP-DOB with it without load balance component. The later comparison is aimed at validating the effect of load balance component when there are massive concurrent service requests. The five comparative strategies are as follows.

1) Entirely Local Computing: Entirely local computing is a traditional manner where the ESs process all the service request on their own without offloading it to the CS for acceleration. Obviously, such an offloading paradigm does not make good use of the network and server resources. Such strategy is referred as ELC.

2) Entirely Cloud Computing: Contrary to the entirely local computing, entirely cloud computing enables all ES to offload their service requests to the CS via wireless channels. This approach makes greater use of the computing resources of the CS and maintains high channel resource utilization when there are not many services and the channel resources are sufficient. However, since the utilization of ESs is extremely low, it is a waste of local computation. Such strategy is referred as ECC.

3) Randomly Binary Distributed Computation Offloading: Service request of an ES is offloaded to the CS in a pre-configured probability. That is, the service may be processed locally or offloaded entirely. Seeing that this offload method takes into account both local resources and cloud resources, it makes up for the shortcomings of the above two methods to some degree. Such strategy is referred as BDO.

4) Approximation Collaborative Computation Offloading Algorithm: In paper [29], approximation collaborative computation offloading (ACCO) strategy is proposed to approximate a near-optimal solution of offloading decision, by adopting a greedy scheme to schedule the offloading decision of each user iteratively.

5) Game-theory Based Offloading Computing with Resource Allocation Optimization: To optimize the multi-user computing offloading decision in edge vehicular networks, a game theoretic approach combined with reinforcement learning (RL) -based resource allocation is designed in the paper[30]. After determining the offloading policy in the respect of the whole system, the algorithm then optimize the resource allocation to further improve the service satisfaction. Such strategy is referred as MTTO.

C. Numerical Analysis

In the scenario of Industrial Metaverse, the generation of service requests, ES status and network environment are highly dynamic. Meanwhile, the number of ESs may vary according to the working hours. Thus, in addition to the effectiveness, the offloading strategy shall have great adaptability as well. To evaluate the performance of our proposed MP-DOB, we

conducted a series comparative experiments and obtained numerical results.

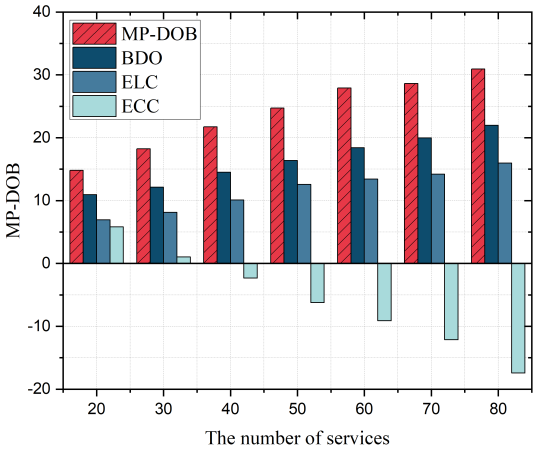

Fig. 3. Comparison analysis of system-wide QoS of MP-DOB, BDO, ELC, and ECC

Fig. 3 shows the system-wide QoS generated by MO-DOB, BDO, ELC, and ECC. With the growth of service request scale, the system-wide QoS of MP-DOB is always nearly doubled compared with that of ELC. When there are few services, the system-wide QoS of ECC is nearly half of that of MP-DOB. However, as the service number increases, the QoS of ECC falls sharply, which reflects the congestion of the channel to the CS due to the massive concurrent service request. Although the system-wide QoS of BDO is larger than that of ELC and ECC, the gap between it and MP-DOB becomes more and more significant as the number of services rises, since BDO only implements cloud-edge collaborative offloading crudely. It is obvious that our proposed MP-DOB outperforms BDO, ELC and ECC in the aspects of service improvement and robustness. Moreover, benefiting from MP-DOB, the stability of the service is also guaranteed in the face of tremendous services.

According to Fig. 4, the system-wide QoS of MP-DOB is only slightly more than that of ACCO when there are few services. Nevertheless, the QoS of MP-DOB is far superior to that of ACCO with the increase of service scale though the QoS of both are growing. The reason for this phenomenon is that as the size of the service rises, the space for offloading decisions becomes very large, which reduces the probability that the approximation algorithm finds the near-optimal offloading policy. While the game theoretic strategy of MP-DOB is deeply optimized for offloading decisions in the layer granularity and provides a more fine-grained offloading solution, which explores the internal constraint relationships of all players. Meanwhile, the load balance module of MP-DOB that improves the edge resource utilization also contributes to the higher QoS.

As is demonstrated in Fig. 5, both MPDOB and MTTO have achieved steady growth in system QoS as the service scale increases, but MP-DOB always has better results than MTTO.

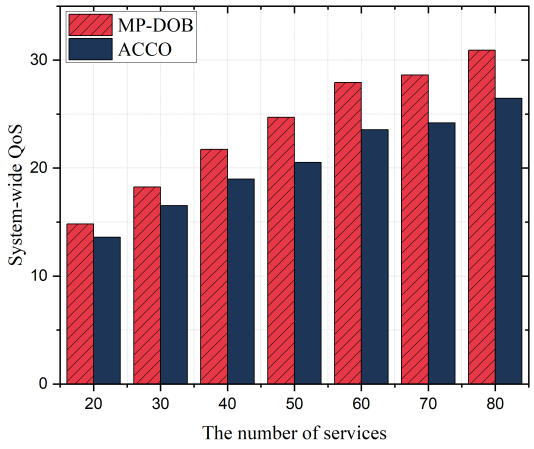

Fig. 4. Comparison analysis of system-wide QoS of MP-DOB and ACCO

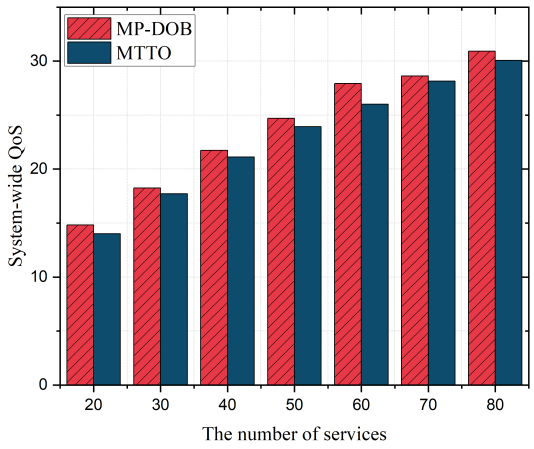

Fig. 5. Comparison analysis of system-wide QoS of MP-DOB and MTTO

Benefiting from the CNN partitioning module of MP-DOB, it facilitates the internal features of the target inference model so that the balance between the computation and communication cost could be better achieved. As a precursor to the offloading decision phase, such feature extraction enables players to express their demands more clearly, which provides a stronger orientation for decision making, thus improving the efficiency and effectiveness of offloading decision optimization. Furthermore, although MTTO leverages an RL-based resource allocation scheme to balance the burden of ESs, the final optimal QoS is still lower than that of MP-DOB equipped with load balance module. Compared with MTTO, our MP-DOB further considers the network status as a form of system load. Hence, the system-level load balance could be achieved and then the QoS improvement is more stable.

Fig. 6 illustrates the trend of the offloading rate of MP-DOB, MTTO, and ACCO. Since only the tail parts are considered to be offloaded to the CS or not, the offloading rate of MP-

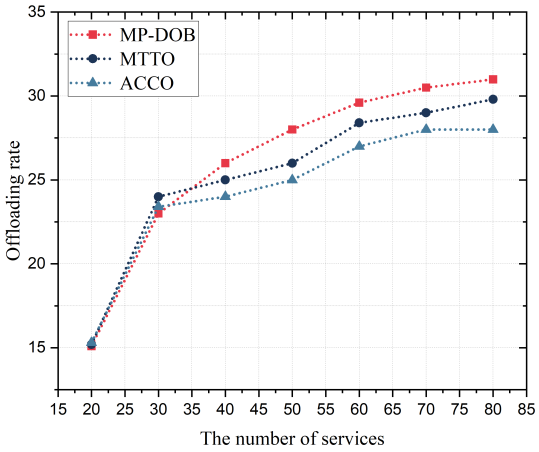

Fig. 6. Comparison analysis of offloading rate of MP-DOB, MTTO, and ACCO

DOB equals to the offloading computation divided by the total computation of tail parts. As to MTTO and ACCO, the offloading rate is equal to the offloaded service number divided by the total number of services in the system. In the above three cases, the distribution of tail parts of original services is roughly the same, which are finally saturated. With the increase of service number, they still keep the similar trend while MP-DOB always occupies the top if the number of service is large. For MTTO and ACCO, when the service number reaches a certain scale, the resources of the cloud-edge environment are in shortage and the high concurrency leads to channel congestion, which indicates an uneven resource utilization. That is, compared with these two strategies, MP-DOB achieves a more reasonable allocation of computing resources, communication resources and energy consumption in the aspect of the entire system. However, the resources of the whole system are limited, so the offloading methods involved all tend to saturate at the end.

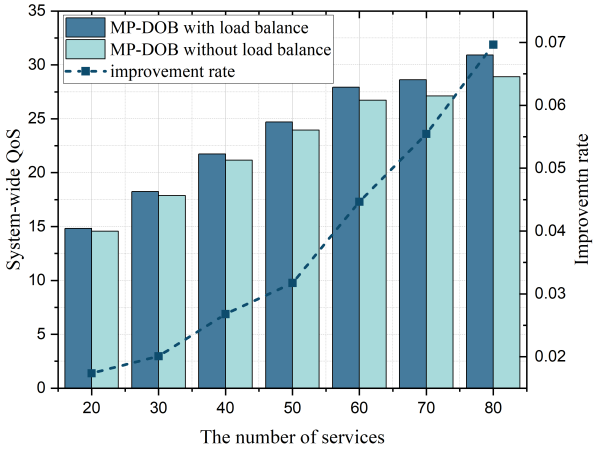

Fig. 7. Analysis of the effectiveness of load balance module

We next discuss the effectiveness of the load balance scheme applied in our MP-DOB. In the above comparative experiments, we have validated the strength of MP-DOB while neglecting to verify the effectiveness of load balance module. Thus, MP-DOB without load balance module will be compared with ultra MP-DOB in the aspect of system-wide QoS. As is illustrated in Fig. 7, the load balance module is effective in terms of different service scale and the improvement of QoS increases with the growth of service number. When the number of service request is not large, the resources in the edge is sufficient and the network status is not congested, leading to few requirements for balancing the load of ESs. However, the more services are requested, the more pressure is put on the ESs, and the balance between edge computing and network communication becomes more and more fragile. Therefore, a load balancing module is needed to further maintain this constraint relationship and make the offloading process of the entire system more stable and adaptable.

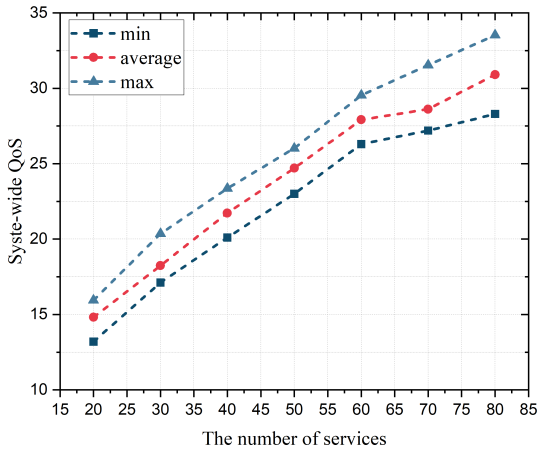

Fig. 8. Analysis of the robustness of MP-DOB

In Fig. 8, the minimum, average, and maximum system-wide QoS are compared to validate the robustness of MP-DOB. With the increase of service scale, the difference between the minimum and maximum QoS is getting larger, which is resulted by the local optimization due to the huge decision space. However, the average system-wide QoS keeps a constant increase, showing great robustness and adaptability.

To demonstrate the convergence of MP-DOB, Fig. 9 shows the variety of system-wide QoS in the duration of decision-making. At the beginning of algorithm execution, the system-wide QoS improves quickly and constantly, which proves the effectiveness of MP-DOB. Finally, the QoS converges to a value and remain stable after finite iterations, that is, the Nash equilibrium is reached.

VI. Conclusion

In this paper, MP-DOB, a game theoretic service offloading strategy equipped with CNN model partition and load balance is proposed to improve the QoS of DL-based human-centric

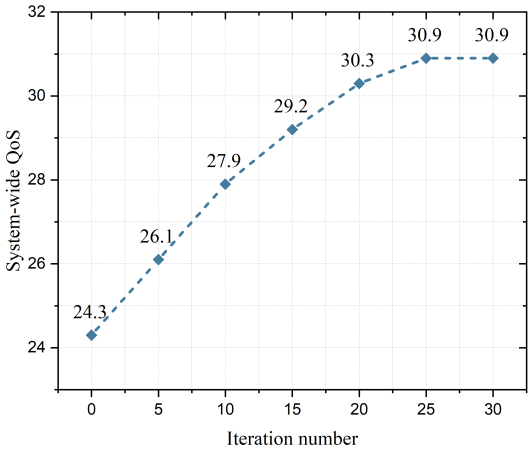

Fig. 9. Analysis of the convergence of MP-DOB

services in Industrial Metaverse. Benefiting from partitioning a large CNN into the head as well as tail and further dynamically segmenting tail parts, MP-DOB takes the advantage of the parallelism of CNN model and maximizes the computation as well as communication resources in the edge. The performance evaluation illustrates that MP-DOB could improve the system-wide QoS within linear logarithmic dimension complexity, which is more efficient and effective than existing strategies. In the future work, we will consider the scenarios and data of Industrial Metaverse scenarios. Moreover, refining MP-DOB into an all-in-one framework is also a promising work.

ACKNOWLEDGMENTS

This research is supported by the Major Research plan of the National Natural Science Foundation of China, no. 92267104, Natural Science Foundation of Jiangsu Province of China under Grant BK20211284, no. BK20211284, and Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps, no. 2017DB005.

REFERENCES

[1] M. Xu, W. C. Ng, W. Y. B. Lim, J. Kang, Z. Xiong, D. Niyato, Q. Yang, X. S. Shen, and C. Miao, "A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies,and challenges," 2022. [Online]. Available: https://arxiv.org/abs/2203.05471

[2] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, and X. Shen, "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2022.

[3] S. Trinks and C. Felden, "Edge computing architecture to support real time analytic applications : A state-of-the-art within the application area of smart factory and industry 4.0," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2930–2939.

[4] M. Xu, D. Niyato, J. Kang, Z. Xiong, C. Miao, and D. I. Kim, "Wireless edge-empowered metaverse: A learning-based incentive mechanism for virtual reality," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 5220–5225.

[5] X. Yao, J. Zhou, J. Zhang, and C. R. Boër, "From intelligent manufacturing to smart manufacturing for industry 4.0 driven by next generation artificial intelligence and further on," in *2017 5th International Conference on Enterprise Systems (ES)*, 2017, pp. 311–318.

[6] R. S. Peres, X. Jia, J. Lee, K. Sun, A. W. Colombo, and J. Barata, "Industrial artificial intelligence in industry 4.0 - systematic review, challenges and outlook," *IEEE Access*, vol. 8, pp. 220 121–220 139, 2020.

[7] K. Peng, H. Huang, M. Bilal, and X. Xu, "Distributed incentives for intelligent offloading and resource allocation in digital twin driven smart industry," *IEEE Transactions on Industrial Informatics*, pp. 1–11, 2022.

[8] J. Wang, W. Jian, and B. Fu, "Edge-to-cloud collaborative for qos guarantee of smart cities," *IFAC-PapersOnLine*, vol. 55, no. 11, pp. 60–65, 2022, iFAC Workshop on Control for Smart Cities CSC 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322011387

[9] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 615–629. [Online]. Available: https://doi.org/10.1145/3037697.3037698

[10] J. Wu, Z. Cao, Y. Zhang, and X. Zhang, "Edge-cloud collaborative computation offloading model based on improved partical swarm optimization in mec," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, 2019, pp. 959–962.

[11] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[12] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[13] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2020.

[14] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based load balancing for fast datacenter networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 465–478. [Online]. Available: https://doi.org/10.1145/2785956.2787507

[15] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2019.

[16] Y. Gao and Z. Li, "Load balancing aware task offloading in mobile edge computing," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2022, pp. 1209–1214.

[17] A. A. Abdelltif, E. Ahmed, A. T. Fong, A. Gani, and M. Imran, "Sdn-based load balancing service for cloud servers," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 106–111, 2018.

[18] S. Teerapittayanon, B. McDanel, and H. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 328–339.

[19] G. Iosifidis, L. Gao, J. Huang, and L. Tassiulas, "An iterative double auction for mobile data offloading," in *2013 11th International Symposium and Workshops on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2013, pp. 154–161.

[20] C.-C. Hsu, J. M. Chang, and Y.-W. Chen, "Joint optimization for cell configuration and offloading in heterogeneous networks," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[21] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1423–1431.

[22] J. Wallenius, J. S. Dyer, P. C. Fishburn, R. E. Steuer, S. Zionts, and K. Deb, "Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead," *Management science*, vol. 54, no. 7, pp. 1336–1349, 2008.

[23] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, p. 23–32, apr 2013. [Online]. Available: https://doi.org/10.1145/2479942.2479946

[24] C. Xian, Y.-H. Lu, and Z. Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in *2007 International Conference on Parallel and Distributed Systems*, 2007, pp. 1–8.

[25] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[26] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124–143, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0899825696900445

[27] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[28] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[29] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[30] Q. Jiang, X. Xu, Q. He, X. Zhang, F. Dai, L. Qi, and W. Dou, "Game theory-based task offloading and resource allocation for vehicular networks in edge-cloud computing," in *2021 IEEE International Conference on Web Services (ICWS)*, 2021, pp. 341–346.