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

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

***Abstract*—Industrial Metaverse is an emerging paradigm of modern industry that aims to create a fully immersive, hyper-spatial, and extremely interoperable virtual space for smart manufacturing, emphasizing a human-centric approach. Benefiting from edge computing and 5G networks, there has been a surge in the development of massive intelligent services (e.g., real-time detection) based on Convolutional Neural Networks (CNNs) deployed at the edge, which require plenty of computational resources. However, the huge volume of service requests can overload edge servers, thus decreasing the quality of service (QoS). Given the limited resources available at the edge, service offloading at the cloud can be employed to ensure the QoS. In this paper, we design MP-DOB, a load balance-aware offloading strategy with CNN model partitioning, which jointly optimizes the inference delay and energy consumption in the service process. Specifically, we partition a CNN model into sub-models in sequential and spatial manners. Then, part of these sub-models is considered to be offloaded at the cloud according to the game theory-based policy. Parallelly, the other part of sub-models is re-scheduled within the edge to further balance the load of edge servers. 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 is expected to revolutionize numerous industries through its immersive and interoperable integration. Similar to how people navigate the Internet with a mouse, virtualization technologies, such as virtual reality (VR), augmented reality (AR), mixed reality (MR), and digital twinning (DT), offer a means to access and explore the Metaverse [2]. By leveraging the aforementioned technologies to merge physical reality with virtual cyber-reality, the Metaverse establishes a human-centric multi-user platform where avatars achieve real-time communications and dynamic interaction. The Industrial Metaverse is being employed within the industry to promote the growth of Industry 4.0 and beyond, extending traditional manufacturing to an intelligent and human-centric stage [3]. Encapsulating a plethora of core components and enabling them to be interoperable in the virtual world, the Industrial Metaverse serves as a shared human-centric space for avatars to collaboratively, efficiently, and dynamically interact with the industrial elements.

In the Industrial Metaverse, a multitude of sensors, actuators, and IoT devices are involved in generating and transmitting vast amounts of data simultaneously, which can lead to significant process latency issues. Therefore, edge computing (EC), which brings the storage and computation of raw data closer to the data sources, is employed by Industrial Metaverse to enhance the ability of real-time information processing [4], thus improving the human-centric service quality. Leveraging the capabilities of EC, the Industrial Metaverse has the potential to facilitate enhanced artificial intelligence-based smart services in virtual manners, such as virtual intelligent industrial simulation [5]. However, the provision of such intelligent services through Industrial Metaverse relies on large-scale deep learning (DL) models, including convolutional neural networks (CNNs), that require substantial computation, memory, and energy resources [6]. In conventional cloud computing (CC), Industrial Metaverse services are directly offloaded to the cloud server (CS), while such a paradigm causes network congestion and instead reduces service latency. Meanwhile, given the constrained resources, the edge can not satisfy services without offloading part of them to the CS. Therefore, it is necessary to adopt a collaborative approach between EC and CC, where edge servers (ESs) can offload services to the CS through high-speed wireless 5G networks. This offloading mechanism helps alleviate the computational burden on the edge while also enhancing 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 are processed according to the offloading policy [8].

Despite the benefits of collaborative inference, offloading the entire DL model at the cloud results in network congestion due to the massive volume of data. This can lead to a significant decline in QoS if communication overhead is greater than the benefit from CC. To achieve more accurate and efficient service offloading for DL-based intelligent services in

Corresponding author: Lianyong Qi.
Xiaolong Xu and Sizhe Tang are co-first authors of the article.
Xiaolong Xu is with the School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China (email: xlxu@nuist.edu.cn).
Sizhe Tang is with the School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China (email: sizhe.tangz@gmail.com).
Xinkui Zhao is with the School of Software Technology, Zhejiang University, Hangzhou 310027, China (email: zhaoxinkui@zju.edu.cn).
Muhammad Bilal is with Hankuk University of Foreign Studies, Korea (South) (email: m.bilal@ieee.org).
Lianyong Qi is with the School of Computer Science, Qufu Normal University, Qufu 273165, China (emial: lianyongqi@qfnu.edu.cn).
Wanchun Dou is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (email: douwc@nju.edu.cn);
Qiang Ni is with Lancaster University, United Kingdom (Great Britain) (email: q.ni@lancaster.ac.uk).

a cloud-edge collaborative scenario, CNN model partitioning, a strategy that partitions a model into several parts in the granularity of layer, is leveraged in the inference stage [9]. Utilizing model partitioning, CNN models can be inferred parallelly and dynamically, thereby accelerating service completion and optimizing edge resource utilization. This reduces service latency, enabling avatars to interact with industrial elements in the metaverse with a rapid response.

However, due to the multi-layer and spatial structure of CNN, it is difficult to optimize the partitioning scheme in accordance with the real-time global status. An efficient and effective service offloading strategy is critical to achieving maximum system-wide Quality of Service (QoS) in a dynamic and resource-constrained system like the Industrial Metaverse. When there is a surge in service demand in a particular system area, the corresponding Edge Servers (ESs) inevitably faces overload. Meanwhile, the resources of its neighboring ESs may be underutilized. Therefore, a load balancing scheme within the edge is required to improve the overall utilization of edge resources and prevent overload.

To address the aforementioned issues, we proposed MP-DOB, a CNN model partitioning-based dynamic offloading strategy with load balance, to jointly optimize the execution latency and energy consumption in the service process. The main contributions of this paper are as follows:

- A distributed service offloading framework based on cloud-edge collaboration in the EC-enabled Industrial Metaverse is presented.
- Spatial CNN partitioning with a smaller granularity is adopted in the proposed strategy to enable parallel processing of CNN models and facilitate the identification of more optimized partial offloading strategies.
- Designed a game theoretic service offloading algorithm to jointly optimize the processing delay and energy consumption of edge devices in Industrial Metaverse.
- A load balance-aware service re-scheduling algorithm within edge is proposed to alleviate the overload of ESs and improve the stability of the proposed MP-DOB.

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. The performance analysis is provided in Section V, followed by the conclusion of this paper in Section VI.

II. Related Work

Benefiting from the rapid advancements of edge computing, the Industrial Metaverse has witnessed the emergence of several intelligent applications[10]. The demanding computation requirements of Industrial Metaverse applications make it difficult for individual edge servers (ESs) to perform inference tasks. Consequently, service offloading, a key technology in cloud-edge collaborative computing, has garnered considerable attention. Chen *et al.* [11] divided the original offloading decision-making task into two sub-problems and leveraged the greedy algorithm to achieve near-optimal cloud-edge and edge-edge collaboration. Guo *et al.* [12] designed a game theoretic and an approximation offloading strategy, to gain the optimal offloading decisions for the cloud-edge collaborative service offloading problem.

The multitude of service requests from devices makes the cloud-edge network highly dynamic and complex. This poses a challenge in making efficient offloading decisions. In order to address these issues, game theory has been widely applied to optimize the decision-making process of offloading strategies, with the aim of achieving good dynamics and convergence. Chen *et al.* [13] proposed a distributed offloading strategy based on potential game to improve the system-wide QoS in a multi-channel scenario. He *et al.* [14] proposed EUAGame, a potential game-based scheme, to efficiently solve a user allocation problem that used to be NP-hard.

Although an efficient and effective offloading decision strategy can improve the users' experience, the overload of servers is still a key problem. Hence, adopting load-balancing techniques to enhance the utility of server resources has garnered significant attention [15]. The strategy in paper [16] considered load balance and offloading jointly in edge vehicular networks aimed at alleviating server performance constraints caused by overloading. The load balancing aims to achieve the maximum offloading rate and service performance. Gao *et al.* [17] designed a load balancing-aware service offloading method to cope with computationally intensive services in the EC environment, aiming to minimize the processing time of user requests by optimizing the server selection scheme. The proposed architecture effectively maintained the real-time load of servers while ensuring robustness.

The large size of DL model-based services presents a significant challenge for existing cloud-edge network offloading strategies to achieve satisfactory results. To address this challenge, model partitioning [9] has been extensively investigated as a key technique for segmenting a CNN into several linear or spatial parts. Kang *et al.* [9] designed a regression strategy to select the most rewarding CNN layer as the partition point, thus accelerating the inference and alleviating constrained resources of edge devices. In the proposed strategy, the origin CNN model is partitioned into heads and tails, which are deployed on the terminal device and ES, respectively. In another work, Yang *et al.* [18] proposed a distributed CNN inference strategy to accelerate intelligent applications in end-edge-cloud scenario.

However, most of the existing research has not adequately considered the scale of inference models and the quantity of services. When a huge volume of services are requested, the high dimensionality of status and offloading decisions can make it challenging for algorithms to be trained within an acceptable time window. Additionally, existing model partitioning 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, tby modeling the optimization of the service offloading policy as a game, the original computational complexity resulting from the large decision space can be effectively reduced.

III. System Model

This section describes the architecture of the service offloading, CNN partitioning model, and service offloading model in the EC-empowered Industrial Metaverse.

A. The framework of Service Offloading

In the framework shown in Fig. 1, 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 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 EC, we adopt a quasi-static requesting manner in this paper to abstract the real-world environments and hence get more insightful analysis [19] [20]. In the 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 when time slots begin. In this paper, it is assumed that one ES receives one service from service requests, which efficiently narrows the solution space for offloading decisions. To demonstrate the offloading process clearly, we introduced the requested service, communication, and computing models next.

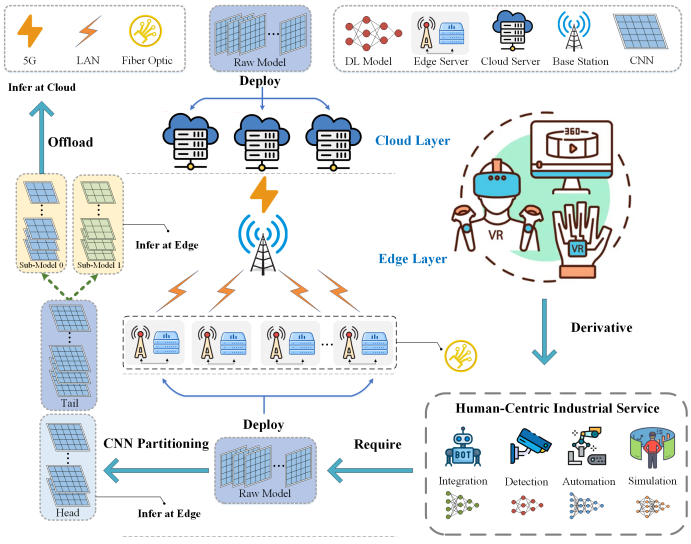

Fig. 1. A framework of intelligent service offloading in Industrial Metaverse

B. 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 into the head and tail sequentially. We define the partitioning layer as the parallel demarcation point (PDP). Head parts are processed by ESs locally, and the tail parts are further partitioned and considered to be offloaded. As to the mainstream CNN models, the data quantity of the first several layers tends to be large, thus increasing the communication cost once they are chosen as PDPs. However, the high inference complexity of the latter layers adds 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 output channel size of 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 costs can 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 decreases 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

$$Re_i = \kappa_1 \chi_i^{com} + \kappa_2 \chi_i^{mem} \quad (4)$$

$$Co_i = \chi_i^{trans} \quad (5)$$

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

$$\varrho_i = \sqrt{\frac{Re_i}{Co_i}}. \quad (6)$$

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 calculation of δ is illustrated specifically in the following section.

C. Communication Model

In the framework of 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 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 a deterioration in service performance. To achieve high QoS, the communication efficiency is optimized as well in the MP-DOB.

D. Computation Model

Upon collecting the service requests and ES statuses, the base station delivers this 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. In the following sections, we provide detailed explanations of these two modes.

1) Edge-Only Mode: In such a 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 denote 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 or energy consumption of the service, then the value of corresponding parameter shall increase accordingly.

Connected via fiber optics, ESs can communicate with each other, which enables them to collaborate to chase 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 the 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 to 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

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

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

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 ES,

communication energy and computing energy of the CS, which is denoted as

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

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

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

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.

E. Problem Definition

In such a 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

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

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

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

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

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 decreases accordingly. Therefore, the goal of MP-DOB is to achieve system-wide QoS maximization.

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

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 network congestion and service overload. The design of game theoretic offloading with CNN partition is shown in Fig. 2.

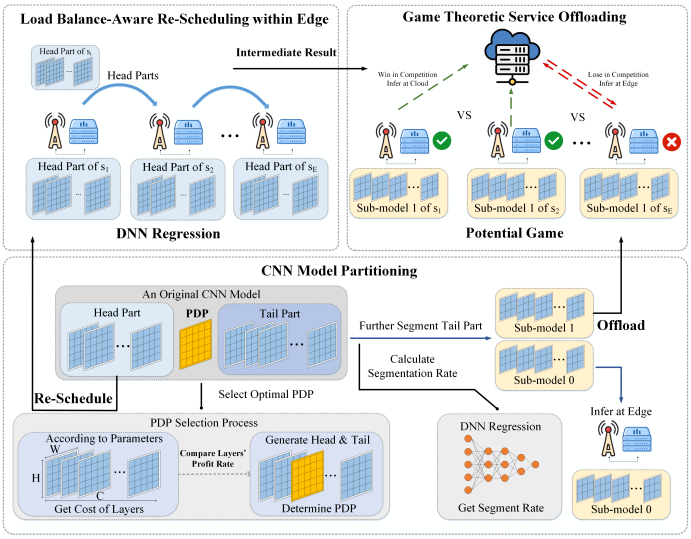

Fig. 2. The Design of Game Theoretic Offloading with CNN Partitioning

A. CNN Model Partitioning

Since the indicator ϱ_i is used to measure the appropriateness of choosing l_i as the PDP. Thus, the goal of the adaptive CNN model partitioning 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), 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 model, thus saving the computing time of the PDP selecting algorithm. However, since the service requests of industrial devices and the computing as well as the memory capacity of ESs are dynamically changing, κ_1 and κ_2 also fluctuate to suit different needs. For instance, bearing a 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 ES is low in memory, then it calls 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 the 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 deep neural network (DNN) to gain the best δ at each time slot. Specifically, the network status, current global ES states, CNN model parameters are input into the DNN, and then an optimal δ is predicted as the segmentation rate. Since the DNN for predicting the δ is shallow, the execution time could be ignored in 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. Algorithm 1 illustrates the specific steps of CNN model partitioning.

Algorithm 1 Adaptive CNN Model Partitioning

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 service 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 results in a decrease in the data transmission rate. Thus, the total processing delay is increased due to the low data rate. Meanwhile, the energy consumption by transmission becomes higher, which worsens the problem of resource scarcity in the edge network. Therefore, the system-wide QoS has decreased accordingly and can not satisfy the service requests. Once the QoS of offloading mode decreases to even lower than the one of the edge-only computing mode, the ES does not choose to offload the tail part to the CS. In such a 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 individual service, chasing a profitable offloading policy means they do 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 results in the higher utilization of network resources and 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 is expanded below.

1) Game Formulation: According to paper [13] [14], 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 service 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 through their intelligence and self-interested traits. Thus, the distributed solution is applied to tackle the primary NP-hard centralized optimization problem. 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 can be determined when no player wants to update the target.

Then, the distributed offloading decision game is formulated. Once receiving the global status θ_{-i} from the base station, all the ESs can determine the 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 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 searching for the optimal offloading policy can be achieved by determining the 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. In such a case, 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 service 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,

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

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

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

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, the player can adopt the service offloading mode and the profit can be achieved. Otherwise, the service shall be processed on the ES.

2) Structural Properties: Next, we leverage a powerful tool, a 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 change in finite decision changes results in a Nash equilibrium. Before proving that the distributed service 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 potential game-based distributed service offloading strategy, which is shown 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 to 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 service offloading game is reached, that is, there is no better offloading policy leading to a higher system-wide QoS.

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-Aware Re-Scheduling within Edge

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 re-schedule and 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 $\{0,1\}^{|E|}$ to narrow the solution space, where 1 means that the ES offloads the head-part inference to the nearest ES while 0 means that no offloading. Considering the scale of solution space, we leverage the 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, it inputs collected statuses into the DNN and then delivers the generated optimal policy to each ES. Algorithm 3 demonstrates the process of the load balance module in detail.

Algorithm 3 Load Balance-Aware Service Re-Scheduling within 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, comparative experiments and numerical analysis are presented to evaluate the efficiency of MP-DOB. We summarize the QoS of all services as the system-wide QoS to measure overall service satisfaction from 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 is 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 industrial center. The transmission power q_i is set as 150 mWatts and the background noise σ_0 equals -100 dBm. Moreover, we assume that 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 a load balance component. The latter comparison is aimed at validating the effect of the load balance component when there are massive concurrent service requests. The comparative strategies are as follows.

1) Entirely Local Computing: Entirely local computing (ELC) is a traditional manner where the ESs process all the service requests on their own without offloading it to the CS for acceleration.

2) Entirely Cloud Computing: Contrary to ELC, entirely cloud computing (ECC) enables all ES to offload their service requests to the CS via wireless channels.

3) Randomly Binary Distributed Service Offloading: In this offloading mode, the 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. Such strategy is referred to as BDO.

4) Approximation Collaborative Computation Offloading Algorithm: In paper [29], an 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) Centralized Optimization for Task Offloading based on Model Partition: MASS [30] is proposed to accelerate DL tasks by offloading them from devices to edge servers after partitioning the original models. This strategy formulates the offloading decision into a non-linear optimization problem and solves it with centralized methods.

6) Distributed Offloading with Potential Game: In [13], a distributed computing offloading (DCO) is proposed to minimize the process latency through the game between users. The competing resources are channels and edge computing forces.

7) 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 [31]. After determining the offloading policy in the respect of the whole system, the algorithm then optimizes the resource allocation to further improve service satisfaction. Such strategy is referred to 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 varies according to the working hours. Thus, in addition to the effectiveness, the offloading strategy shall have great adaptability as well.

Fig. 3 demonstrates the system-wide QoS of MP-DOB, BDO, ELC, and ECC. With the growth of the service request scale, the system-wide QoS of MP-DOB almost always doubles that of ELC. When the number of services is not large, such an indicator 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

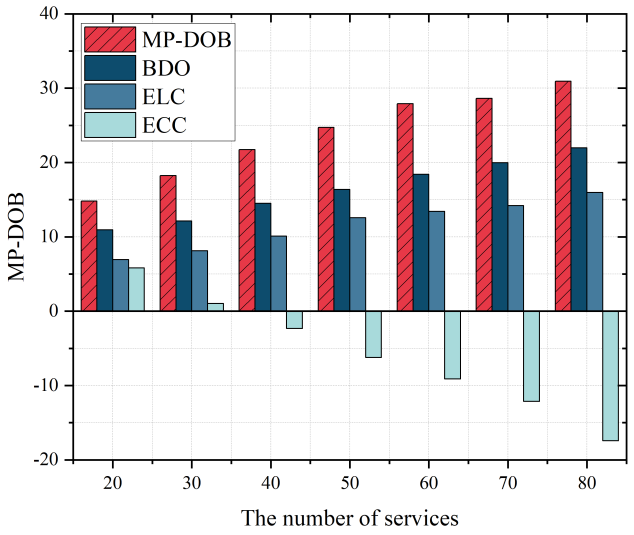

Fig. 3. Comparison analysis of system-wide QoS of MP-DOB, BDO, ELC, and ECC

more significant as the number of services rises, since BDO only implements cloud-edge collaborative offloading crudely. Obviously, MP-DOB outperforms BDO, ELC, and ECC in the effectiveness and stability. Moreover, benefiting from MP-DOB, the stability of the service is also guaranteed in the face of tremendous services.

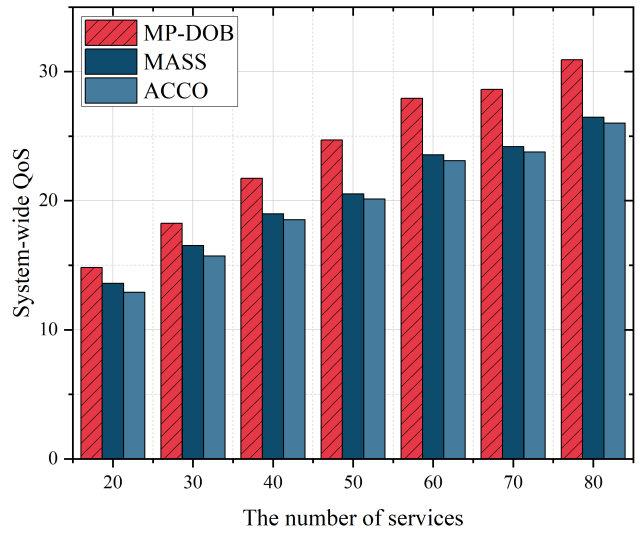

Fig. 4. Comparison analysis of system-wide QoS of MP-DOB, MASS, and ACCO

According to Fig. 4, the system-wide QoS of MP-DOB is only slightly more than that of MASS and ACCO when there are few services. Nevertheless, the QoS of MP-DOB is far superior to that of MASS and ACCO with the increase of service scale though the QoS of both are all 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 MASS and ACCO find near-optimal offloading policies. 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.

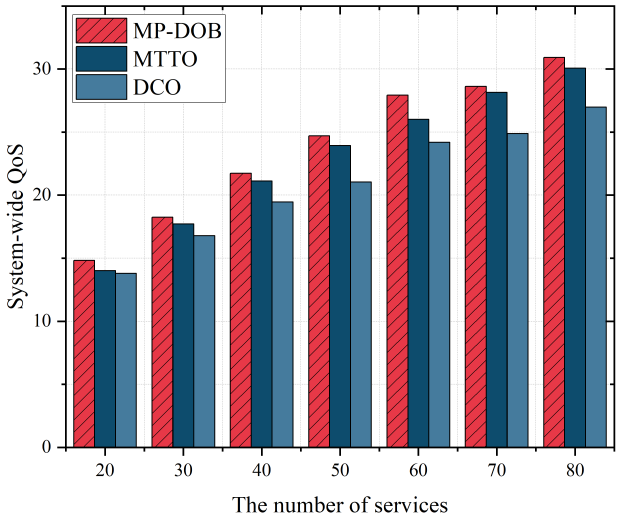

Fig. 5. Comparison analysis of system-wide QoS of MP-DOB, MTTO, and DCO

As is demonstrated in Fig. 5, such three strategies all have achieved steady growth in system QoS as the service scale increases, but MP-DOB always has the best results. Benefiting from the CNN partitioning module, MP-DOB 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 a 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. The offloading rate of MP-DOB equals the offloaded 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 above three cases, the distribution of tail parts of original services is almost the same, which means they all reach the saturation finally. With the increase in service number, they still keep a similar trend while MP-DOB always occupies the top if the number of services is large. For MTTO and ACCO, the resources of the cloud-edge environment are in shortage and the high concurrency leads to channel congestion with the growth of service number, which indicates uneven resource utilization. That is, compared with these

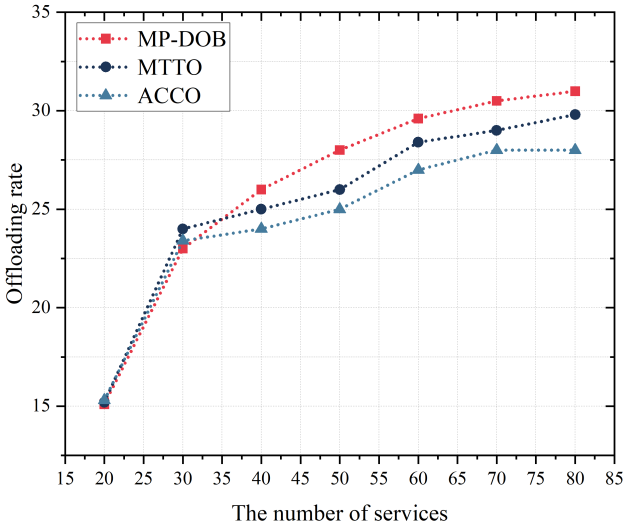

Fig. 6. Comparison analysis of offloading rate of MP-DOB, MTTO, and ACCO

two strategies, MP-DOB achieves a more reasonable system-wide resource allocation, considering the energy consumption. However, the resources of the whole system are limited, so the offloading methods involved all tend to saturate at the end.

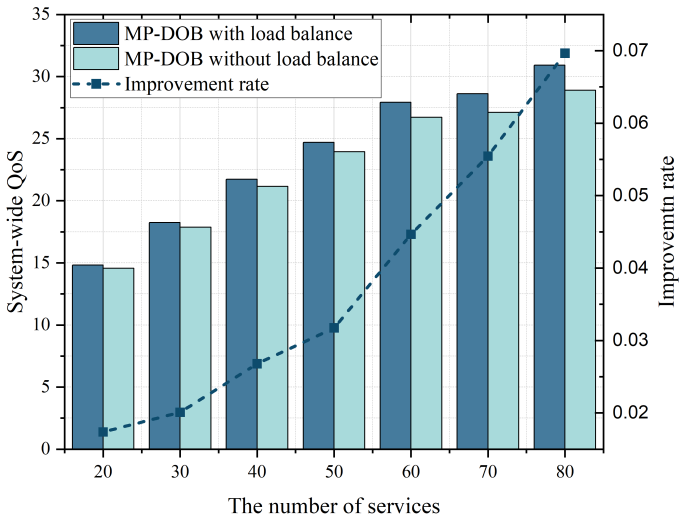

Fig. 7. Analysis of the effectiveness of load balance module

In the above comparative experiments, we have validated the strength of MP-DOB while neglecting to verify the efficacy of the load balance module. Thus, MP-DOB without a load balance module is compared with ultra MP-DOB in the aspect of system-wide QoS. As illustrated in Fig. 7, the load balance module is effective in terms of different service scales and the improvement of QoS grows with the service number. When the service request scale 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 EC and network communication becomes more and more fragile. Therefore, a load balancing module is needed to re-schedule head parts of services and make the offloading process of the entire system more stable and adaptable.

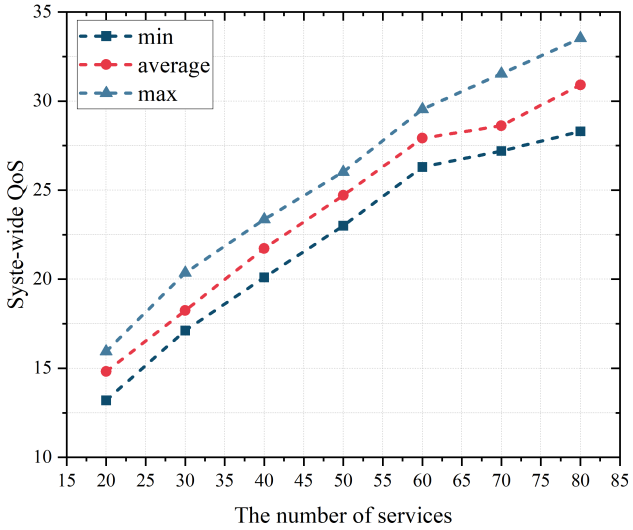

Fig. 8. Analysis of the robustness of MP-DOB

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.

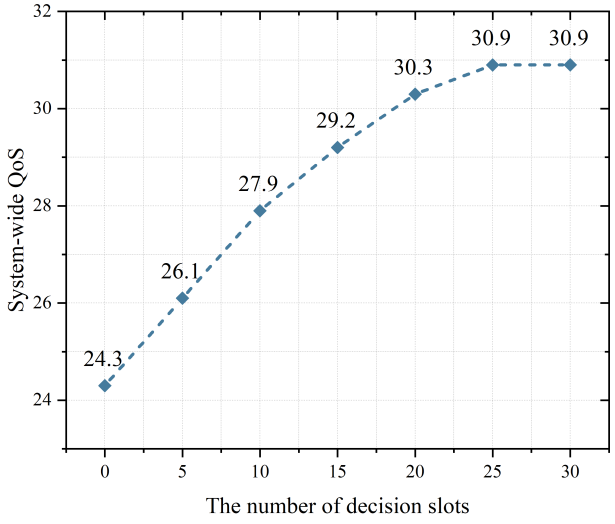

Fig. 9. Analysis of the convergence of MP-DOB

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.

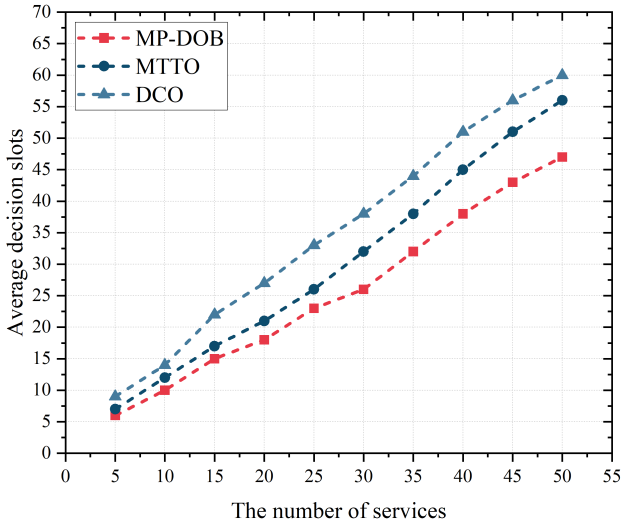

Fig. 10. Comparison of average required decision slots of MP-DOB, MTTO, and DCO

As is shown in Fig. 10, with the number of services rising, the average decision slots of the compared three strategies grow almost nearly and MP-DOB always requires the least slot quantity. Such a phenomenon demonstrates that the performance of MP-DOB is not much affected by the number of services. Hence, MP-DOB is proven to have great adaptability.

VI. Conclusion

In this paper, we propose MP-DOB, a service offloading strategy based on game theory, which incorporates CNN model partitioning and load balancing to improve the QoS of DL-based human-centric services in the Industrial Metaverse. To enhance the efficiency of the offloading and load balancing strategy, MP-DOB divides a large CNN model into the head and tail, and dynamically segments the tail parts. This technique takes advantage of the parallelism of the CNN model and maximizes the utilization of computation and communication resources at the edge. The performance evaluation illustrates that MP-DOB could improve the system-wide QoS within linear logarithmic dimension complexity, surpassing the efficiency and efficacy of existing strategies. In our future work, we plan to explore the scenarios and data of Industrial Metaverse in more detail and investigate their impact on the performance of MP-DOB. We also plan to refine MP-DOB into a comprehensive architecture able to handle common DL-based human-centric services with different complexity levels and requirements.

Acknowledgments

This research is supported by the Major Research plan of the National Natural Science Foundation of China under Grant 92267104, the Natural Science Foundation of Jiangsu Province of China under Grant BK20211284, and the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps under Grant 2020DB005.

References

[1] M. Xu, W. C. Ng, W. Y. B. Lim, J. Kang, Z. Xiong, D. Niyato, Q. Yang, X. Shen, and C. Miao, "A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 656–700, 2023.

[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*, vol. 25, no. 1, pp. 319–352, 2023.

[3] Z. A. E. Houda, B. Brik, A. Ksentini, L. Khoukhi, and M. Guizani, "When federated learning meets game theory: A cooperative framework to secure iiot applications on edge computing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 7988–7997, 2022.

[4] P. Huang, L. Zeng, X. Chen, K. Luo, Z. Zhou, and S. Yu, "Edge robotics: Edge-computing-accelerated multirobot simultaneous localization and mapping," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 14 087–14 102, 2022.

[5] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: Architecture, advances and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[6] G. Premsankar and B. Ghaddar, "Energy-efficient service placement for latency-sensitive applications in edge computing," *IEEE internet of things journal*, vol. 9, no. 18, pp. 17 926–17 937, 2022.

[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*, vol. 19, no. 3, pp. 3133–3143, 2023.

[8] M. Gao, R. Shen, J. Li, S. Yan, Y. Li, J. Shi, Z. Han, and L. Zhuo, "Computation offloading with instantaneous load billing for mobile edge computing," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1473–1485, 2022.

[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.

[10] K. Li, Y. Cui, W. Li, T. Lv, X. Yuan, S. Li, W. Ni, M. Simsek, and F. Dressler, "When internet of things meets metaverse: Convergence of physical and cyber worlds," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4148–4173, 2023.

[11] L. Chen, J. Wu, J. Zhang, H.-N. Dai, X. Long, and M. Yao, "Dependency-aware computation offloading for mobile edge computing with edge-cloud cooperation," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2451–2468, 2022.

[12] 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.

[13] 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.

[14] 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.

[15] H. Guo, X. Zhou, Y. Wang, and J. Liu, "Achieve load balancing in multi-uav edge computing iot networks: A dynamic entry and exit mechanism," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18 725–18 736, 2022.

[16] 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.

[17] 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.

[18] S. Yang, Z. Zhang, C. Zhao, X. Song, S. Guo, and H. Li, "Cnnpc: End-edge-cloud collaborative cnn inference with joint model partition and compression," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4039–4056, 2022.

[19] L. Yuan, Q. He, S. Tan, B. Li, J. Yu, F. Chen, and Y. Yang, "Coopedge+: Enabling decentralized, secure and cooperative multi-access edge computing based on blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 894–908, 2023.

[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] R. Lin, T. Xie, S. Luo, X. Zhang, Y. Xiao, B. Moran, and M. Zukerman, "Energy-efficient computation offloading in collaborative edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21 305–21 322, 2022.

[23] W. Feng, S. Lin, N. Zhang, G. Wang, B. Ai, and L. Cai, "Joint c-v2x based offloading and resource allocation in multi-tier vehicular edge computing system," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 432–445, 2023.

[24] S. Yue, J. Ren, N. Qiao, Y. Zhang, H. Jiang, Y. Zhang, and Y. Yang, "Todg: Distributed task offloading with delay guarantees for edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 7, pp. 1650–1665, 2022.

[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] X. Gao, R. Liu, and A. Kaushik, "Virtual network function placement in satellite edge computing with a potential game approach," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1243–1259, 2022.

[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] D. Li, Z. Ke, and X. Zhou, "Mass: Multi-edge assisted fast object detection for autonomous mobile vision in heterogeneous edge networks," in *Proceedings of the 17th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, ser. Q2SWinet '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 61–68.

[31] 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.