Cost-Efficient Deep Neural Network Placement in Edge Intelligence-Enabled Internet of Things

HAO TIAN, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China, Nanjing, China and School of Software, Nanjing University of Information Science and Technology, Nanjing, Jiangsu, China, Nanjing, China
XIAOLONG XU*, School of Software, Nanjing University of Information Science and Technology, Nanjing, Jiangsu, China, Nanjing, China
HONGYUE WU, College of Intelligence and Computing, Tianjin University, Tianjin, China, Tianjin, China
QINGZHAN ZHAO, Geospatial Information Engineering Research Center, College of Information Science and Technology, Shihezi University, Shihezi, Xinjiang, China, Shihezi, China
JIANGUO DAI, Geospatial Information Engineering Research Center, College of Information Science and Technology, Shihezi University, Shihezi, Xinjiang, China, Shihezi, China
MAQBOOL KHAN, Department of IT and Computer Science, Pak-Austria Fachhochschule Institute of Applied Sciences and Technology, Haripur, Pakistan, Haripur, Pakistan

Edge intelligence (EI) integrates edge computing and artificial intelligence empowering service providers to deploy deep neural networks (DNNs) on edge servers in proximity to users to provision intelligent applications (e.g., autonomous driving) for ubiquitous Internet of Things (IoT) in smart cities, which facilitates the quality of experience (QoE) of users and improves the processing and energy efficiency. However, considering DNN is typically computational-intensive and resource-hungry, conventional placement approaches ignore the influence of multi-dimensional resource requirements (processor, memory, etc.), which may degrade the real-time performance. Moreover, with the increasing scale of geo-distributed edge servers, centralized decision-making is still challenging to find the optimal strategies effectively. To overcome these shortcomings, in this paper we propose a game theoretic DNN placement approach in EI-enabled IoT. First, a DNN placement optimization problem is formulated to maximize system benefits, which is proven to be $\mathcal{NP}$-hard and model the original problem as an exact potential game (EPG). Moreover, an EPG-based DNN model placement algorithm, named EPOL, is designed for edge

*Xiaolong Xu is corresponding author.

Authors' Contact Information: Hao Tian, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China, Nanjing, Jiangsu, China and School of Software, Nanjing University of Information Science and Technology, Nanjing, Jiangsu, China, Nanjing, Jiangsu, China; e-mail: withhaotian@gmail.com; Xiaolong Xu, School of Software, Nanjing University of Information Science and Technology, Nanjing, Jiangsu, China, Nanjing, Jiangsu, China; e-mail: njuxlxu@gmail.com; Hongyue Wu, College of Intelligence and Computing, Tianjin University, Tianjin, China, Tianjin, Tianjin, China; e-mail: hongyue.wu@tju.edu.cn; Qingzhan Zhao, Geospatial Information Engineering Research Center, College of Information Science and Technology, Shihezi University, Shihezi, Xinjiang, China, Shihezi, Xinjiang, China; e-mail: zqz_inf@shzu.edu.cn; Jianguo Dai, Geospatial Information Engineering Research Center, College of Information Science and Technology, Shihezi University, Shihezi, Xinjiang, China, Shihezi, Xinjiang, China; e-mail: djg_inf@shzu.edu.cn; Maqbool Khan, Department of IT and Computer Science, Pak-Austria Fachhochschule Institute of Applied Sciences and Technology, Haripur, Pakistan, Haripur, Pakistan; e-mail: maqbool.khan@scch.at.

ACM 1550-4867/2024/8-ART
https://doi.org/10.1145/3685930

servers to make sub-optimal strategies independently and theoretical analysis is possessed to guarantee the performance of EPOL. Finally, real-world dataset based experimental results corroborate the superiority and effectiveness of EPOL.

CCS Concepts: • **Theory of computation → Algorithmic game theory and mechanism design**; • **Networks → Network design and planning algorithms**; • **Computing methodologies → Distributed algorithms**.

Additional Key Words and Phrases: Edge intelligence, DNN placement, Game theory, Internet of Things

1 Introduction

Driven by the explosive growth of the Internet of Things (IoT) devices in recent years, including smartphones, wearable devices, and unmanned aerial vehicles (UAVs), the scale of generated data at the end increases exponentially [1, 2]. It is expected that the number of IoT connections will reach 25 billion around the world by 2025, up from 12 billion in 2019, according to GSMA's report [3]. At the same time, Artificial Intelligence (AI) technology has made a great breakthrough, especially in deep neural networks (DNNs) and its offshoots, e.g., convolutional neural networks (CNNs), recurrent neural network (RNNs), graph neural network (GNNs), and Transformer [4]. The developments of DNN upgrade the capabilities of IoT devices in environmental perception and data analytics, which spawns various smart applications (e.g., face recognition, object detection, AR/VR gaming) [5, 6]. With DNNs, these applications enable the improvement of the quality of experience (QoE) of users and provide intelligent decision-making for users. Generally, service providers, e.g., AWS, deploy the DNN models on the cloud data centers with abundant resources through multi-tenancy patterns to serve different tenants simultaneously and provision DNN services with charges. To perform the DNN execution on the cloud rather than locally, the processing pressure of devices will be alleviated, and the battery life will be increased.

Many delay-sensitive applications, such as autonomous driving that depends on the DNNs execution for video processing and traffic flow prediction, require processing in milliseconds level response [7, 8]. However, it is worth noting that DNNs execution is computationally-intensive and resource-hungry. For example, a classical ResNet-152 requires 11.57 GFLOPs (giga floating point operations), 226.06 MB memory footprint, and 240.77 MB parameter storage [9]. Moreover, a large amount of data collected from IoT devices have to be transmitted to the cloud data centers for processing. In view of this, the cloud computing paradigm enabled DNN execution incurs a great bandwidth consumption and unpredictable latency, which may degrade the QoE of users and quality of service (QoS) of networks [10]. To tackle such issues, edge computing pushes the computing and storage resources into the network edge, deploying edge servers on the infrastructures (such as base stations and access points) in proximity to IoT devices to offer flexible supply of distributed resources [11]. Edge computing enables devices to offload the computations to neighboring edge servers instead of fetching services from remote clouds, which decreases the latency and transmission overhead. Currently, lots of service providers have pushed the application instances from the cloud to the edge to serve users. For instance, Azure Edge Zones is a classical product from Microsoft that provides edge computing services to end-users and users can pay charges according to their practical demands [12].

With the integration of edge computing and AI, a promising paradigm referred to as edge intelligence [13] is emerging to facilitate network optimization and processing efficiency. Edge intelligence enables the AI functionalities to be deployed at the network edge to serve nearby IoT devices with lower computing latency and higher throughput. Recently, edge intelligence has attracted much attention in the literature in two main focuses, i.e., edge model training and edge model inference. On one hand, geographically distributed edge servers can be used as computing nodes to cooperatively train DNN models for IoT devices. Processing and power constrained IoT devices send data samples to several edge servers to perform DNN training and eventually learn high-quality DNN models [14, 15]. For example, a classical training mode, federated edge learning [16], unleashes the potential of collaborative training in the edge. In federated edge learning, the IoT device first performs the training of the local model, and then uploads the local gradient to the edge server to complete the global model aggregation and

edge server returns the results to the IoT device in each iteration step [17]. It not only effectively improves data privacy but also avoids the transmission delay caused by uploading a large amount of data to the centralized cloud. On the other hand, edge inference enables IoT devices to offload part of the high-overhead DNN computations to the edge servers for execution, so as to meet the requirements of real-time applications. For example, the model partition mechanism splits the DNN model into two parts, one part of the computation is done on the IoT device, and the other half is done on the edge server [18]. By fully exploring the computation and data transmission level of each layer in DNN, the optimal partition point is selected to minimize the inference delay and improve energy efficiency.

Although deploying DNN models on edge servers promotes the QoE and QoS, it is still facing three major challenges as follows. First, edge servers have limited resources compared to cloud data centers, e.g., CPU, DRAM, and disk. For a variety of DNN requests from devices, it is unrealistic for edge servers to place DNN models to handle all requests. On the other hand, placed DNNs that no user requests will cause the wasting of resources. Therefore, it is essential for edge servers to select appropriate DNNs to meet the demands of IoT devices and improve resource utilization. Second, from the perspective of service providers, the main focus is to increase their revenues for deploying DNN models on edge servers to serve multiple tenants with intelligent application instances. Due to the competition of resources between multi-edge servers and multi-devices, as well as the selfishness of each edge server that only considers its own requests, how to accomplish as many DNN computation requests as possible to maximize the system benefit while achieving minimal placement cost should be fully investigated. Third, considering the geographical distribution of edge servers, conventional centralized decision-making algorithms depend on the number of edge servers. When the number of edge servers reaches to large scale, it is difficult to be practical and the controlling cost for management of edge servers is also increased. Such that, how to realize a decentralized decision algorithm so that each edge server can make strategies independently without global information should be considered.

To tackle the aforementioned issue, a game theoretic DNN placement approach is introduced in edge intelligence. Specifically, we focus on the design of decentralized and effective DNN model placement strategies. First, the DNN placement problem is formulated as a constraint optimization problem to maximize the system benefit of edge servers taking into account multi-dimensional resources (i.e., processor, memory, and disk), which is proven to be $\mathcal{NP}$-hard. To solve the formulated problem, the original problem is modeled as a game, DMPG. Afterward, an EPG-based DNN model placement algorithm, named EPOL, is proposed to find the sub-optimal decisions that each edge server can make decision-making independently without any global information. A comprehensive theoretical analysis is given to prove that DMPG is an exact potential game that reaches an Nash equilibrium. Convergence analysis and complexity analysis are possessed to guarantee the theoretical performance of EPOL. Finally, comparative experiments are performed on a real-world edge computing dataset to evaluate the performance of EPOL algorithm and demonstrate that EPOL is superior to other existing algorithms.

The main contributions of this paper are concluded in the following:

- *Constrained DNN model placement optimization problem is formulated.* To maximize the system benefit of edge servers from the perspective of service providers, we formulate a DNN model placement optimization problem taking into account multi-dimensional resources (i.e., processor, memory, and disk), which is proven to be $\mathcal{NP}$-hard.
- *A cost-efficient DNN placement game DMPG is modeled and decentralized game theoretic algorithm EPOL is designed to decide the model placement.* To solve the formulated problem, the problem is modeled as a potential game to make placement strategies for each edge server to maximize the benefit and reduce the placement cost. According to the DMPG, we propose EPOL, an EPG-based DNN model placement algorithm to find the sub-optimal strategies for edge servers that make independent placement decisions without any global information.

- *Comprehensive theoretical analysis and extensive experiments are given.* First, we analyze the DMPG and prove it admits an Nash equilibrium with a convergence guarantee. Then, the performance comparison of EPOL is conducted with the other three existing algorithms and simulation results demonstrate the effectiveness of EPOL.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. Section 3 presents the system model and optimization problem formulation. In Section 4, the DMPG game is proposed. The design of EPOL algorithm is derived and convergence and complexity analysis of EPOL is also given to guarantee the theoretical performance in Section 5. Section 6 possesses comparative experiments to evaluate the effectiveness of EPOL. Consequently, we conclude the paper in Section 7.

2 Related Work

In this section, we will present the discussions on previous research topics with respect to edge intelligence and proactive placement at the edge as follows.

2.1 Edge Intelligence

Edge intelligence is the integration of edge computing and AI, which can not only improve the QoS and resource utilization at the edge, but also unleash the potential of pushing intelligent services on the edge in proximity to end-users. In view of this, edge intelligence has received much attention in recent years. Dai et al. [19] investigated the joint computation offloading and resource problem by utilizing deep reinforcement learning (DRL) based algorithm to derive an energy-efficient solution. Except for energy consumption, latency is another factor that greatly influences the QoS of users. Ale et al. [20] focused on the computation task deadline and power cost and then proposed a DRL-based edge server selection and computational resource allocation strategy to maximize the long-term utility. Furthermore, Song et al. [21] attempted to develop an edge caching method to improve the QoE in an edge vehicular computing environment. To enhance the hit ratio, an algorithm based on DRL was designed to efficiently solve the caching file update issues. Wang et al. [22] also devoted their efforts to improving device-to-device caching performance by Q-learning based method. Moreover, an attention-weighted mechanism was proposed to integrate with federated learning to improve the training efficiency of Q-learning agents, which jointly optimizes the fetching delay and hit ratio.

On the other hand, sinking AI services and applications on edge can reduce the processing delay and transmission cost where massive raw data have to send from the end to the remote cloud data centers [13, 23]. Two representative techniques for AI are training and inference. Due to the geo-distributed edge nodes, the distributed training on edge nodes is a promising solution. Ren et al. [16] studied federated edge learning that the local models are aggregated in edge nodes to accelerate the training process. Both considering the CPU and GPU scenarios, a federated edge learning framework was constructed to acquire better learning performance. Ma et al. [24] focused on the improvement of training waiting time and energy consumption. To this end, they designed an adaptive batch size approach for federated edge learning to enhance the stability and convergence performance of the global model on edge nodes. Besides, Laskaridis et al. [25] investigated the distributed inference on DNN models that the DNN is split into two parts and executed at end devices and edge servers respectively, which aims to optimize the inference delay and throughput to meet low-latency service demands.

2.2 Proactive Placement at Edge

For service providers, applications and services can be deployed on the edge servers at the network edge for users to satisfy various requirements with low latency and high reliability. Maia et al. [26] proposed an IoT service placement method with a meta-heuristic algorithm and linearization-based algorithm that optimizes the QoS violation of applications by considering QoS demands, resource availability, and workload level. Lv

et al. [27] devoted their efforts to microservice deployment issues on edge computing. A reward-sharing deep Q-learning algorithm was designed to optimize the transmission cost and load balance and further an elastic scaling algorithm was developed to enhance the service scalability. Ning et al. [28] studied the dynamic service placement problem taking into account the storage capacity of edge servers and service execution delay. Then several algorithms were proposed to improve the whole system utility. Li et al. [29] focused on the robustness of application deployment from the service providers' perspective. Specifically, the formulated robustness-oriented edge application deployment problem is solved by both integer programming and approximation algorithms. To improve both the revenue of service providers and edge servers, Deng et al. [30] proposed a two-stage spontaneous edge deployment algorithm with pricing approaches based on an incentive mechanism to enable service providers to select optimal edge servers to place applications. Gao et al. [31] considered the joint network selection and service placement optimization problem. To enhance the QoS and reduce the delay and cost, they derived an online framework with a two-phase algorithm to achieve neat-optimal effectiveness.

To the best of our knowledge, few works have investigated the proactive placement of DNNs at the edge. Different from previous work with conventional service placement on edge servers, DNN models are multi-dimensional resource-intensive (e.g., processor, memory, disk, and network). In view of this, from the perspective of service providers, we have to consider more factors that may influence the system benefit. Meanwhile, due to the geo-distributed features of edge servers, it is challenging to manage the centralized control for each node. Therefore, in this paper, we will present a decentralized DNN model placement approach that maximizes the system benefit of edge servers.

3 Preliminaries and Problem Formulation

3.1 Preliminaries

3.1.1 Deep Neural Network Models. A DNN is a type of artificial neural network that is designed to simulate the structure and function of the human brain, specifically in its ability to learn and process information. DNNs consist of multiple layers of interconnected nodes, called neurons or units, which are organized into an input layer, one or more hidden layers, and an output layer. With the rapid development of AI, DNN has derived many branches, including CNN, RNN, etc [32].

In specific, the input layer receives raw data, such as images, text, or numerical values, and passes it through the network. Each neuron in the input layer represents a feature or attribute of the data. The hidden layers, often consisting of multiple layers, perform complex computations on the input data. Each neuron in a hidden layer takes in inputs from the previous layer, applies weights to them, and processes them with an activation function to produce an output. This output is then passed to the next layer's neurons. The final layer of the network is the output layer, which provides the desired prediction or classification based on the input data. The output layer can have different configurations depending on the task at hand. For example, in image recognition, each neuron in the output layer might represent a different class or category that the network is trained to recognize. For example, as a classical CNN model, AlexNet [33] consists of 5 convolutional layers and 3 fully connected (FC) layers, where the last FC layer with softmax function is the output layer for prediction. DNNs have achieved remarkable success in various fields, including computer vision, natural language processing, speech recognition, and many more. They have outperformed traditional machine learning algorithms in tasks such as image classification, object detection, language translation, and speech synthesis.

In this paper, we focus on the novel DNN model placement strategies at the edge aiming to improve the QoS of users to satisfy requirements of intelligent applications.

3.1.2 Edge Intelligence Paradigm. Edge intelligence refers to the capability of performing real-time data processing, analysis, and decision-making directly on edge devices or edge servers in proximity to data source under the integration of edge computing and AI, rather than relying solely on centralized cloud-based resources.

The integration brings AI capabilities directly to the edge of network, enabling to optimize the edge network performance and the processing efficiency of intelligent applications. Recently, edge intelligence has attracted wide attention in the academic community, which mainly focuses on two aspects, i.e., Artificial Intelligence on Edge (AIE) and Intelligence-enabled Edge Computing (IEC) [13].

On one hand, edge deployment of AI applications mainly focuses on model training and inference at the edge. Firstly, edge servers, dispersed geographically, can be used as computing nodes to collaboratively train DNN models for end devices. For instance, the classical training mode, federated edge learning, unleashes the potential of collaborative training at the edge [34]. Federated edge learning not only effectively improves data privacy but also avoids the transmission delay caused by uploading a large amount of data to the cloud. In addition, edge inference allows end devices to offload the high-cost DNN computations to edge servers for application requirements, which minimizes the end-to-end execution latency and improves the battery efficiency of end devices.

On the other hand, IEC involves considering how to effectively utilize AI technologies and methods to optimize the many key issues in edge computing, including computation offloading, service caching, etc. Through the ability of DNNs to extract data features, it is possible to perceive the topology of dynamic networks and the resource status of edge servers. Meanwhile, DRL has real-time decision-making capabilities in dynamic environments through interaction with the environment, and can learn long-term optimal resource management and task scheduling strategies under time-varying scenarios to achieve intelligent control at the edge.

3.2 System Overview

As illustrated in Fig 1, an application scenario for DNN model placement in edge intelligence-empowered IoT is given. In this scenario, the set of IoT devices is denoted as $\mathcal{I} = \{i_m | m = 1, 2, \ldots, M\}$, and the set of edge servers is denoted as $\mathcal{E} = \{e_n | n = 1, 2, \ldots, N\}$. There are multiple kinds of DNN models that can be deployed in edge servers for specific requirements (e.g., object detection, person tracking). Let $\mathcal{D} = \{d_o | o = 1, 2, \ldots, O\}$ denote the set of DNN models. The necessary notations used in this paper are listed in TABLE 1.

Table 1. Summary of notations

Notation	Description
$\mathcal{I}$	Set of IoT devices
$\mathcal{E}$	Set of edge servers
$\mathcal{M}$	Set of DNN models
r_i	Set of DNN requests of IoT device i
k_i	Number of requests of IoT device i
d_i^j	Required DNN model of from request r_i^j
c_i^j	Required processing resources of request r_i^j
m_i^j	Memory footprint in runtime of request r_i^j
s_i^j	Occupied storage resources of request r_i^j
$x_{i,e}$	Connection association between device i and edge server e
a_e	DNN model placement decisions
G_e	Revenue of edge server e
C_e	Placement cost of edge server e
S_e	Placement capacity of edge server e

IoT device i can generate DNN execution request denoted as $r_i = \left\{r_i^1, r_i^2, \ldots, r_i^{k_i}\right\} (i \in \mathcal{I})$, where k_i is the number of requests on device i. More specifically, DNN request $r_i^j (1 \leq j \leq k_i)$ mainly includes four parts, i.e.,

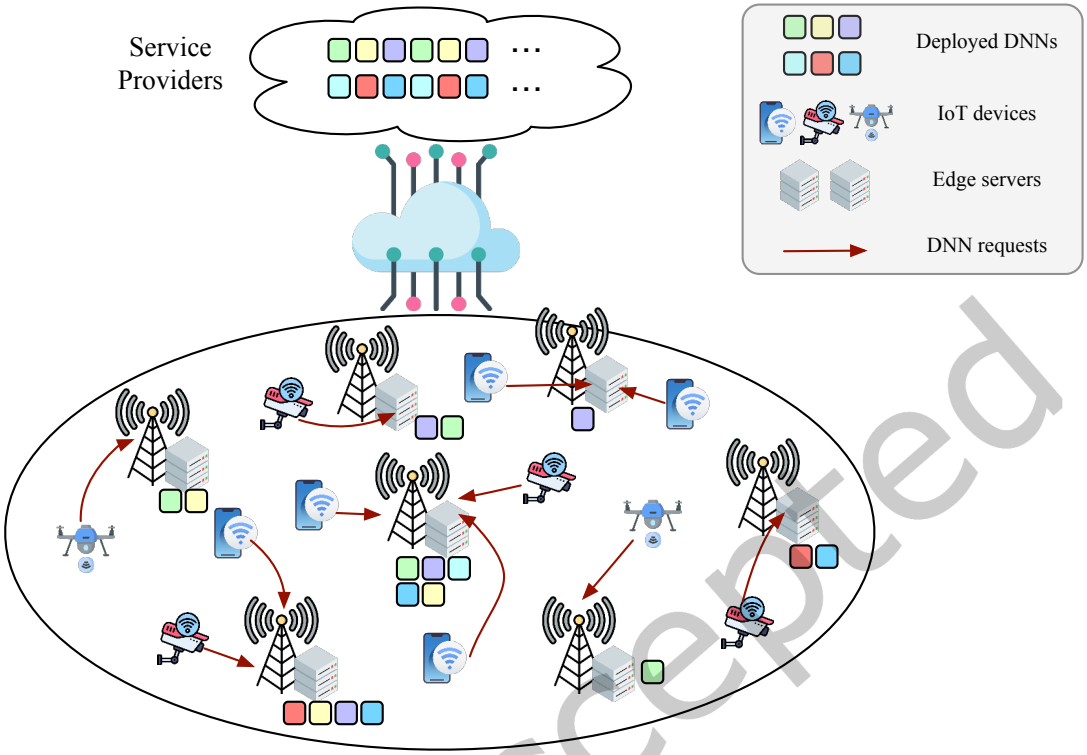


Fig. 1. An application scenario for DNN model placement in edge intelligence-empowered IoT.

$r_i^j = \{d_i^j, c_i^j, m_i^j, s_i^j\}$, where d_i^j is the required DNN model, c_i^j is the required processing resources, m_i^j is the memory footprint in runtime, and s_i^j is the occupied storage resources. For service providers, they tend to deploy more DNN models with as less as edge servers to meet massive demands from devices. However, due to the constrained computing, memory, and storage resources, the main goal for service providers is to employ an efficient DNN model placement approach gaining as much profit as possible.

3.3 Processing Delay Model

Request r_i generated from device i will connect to edge server e for satisfying multiple DNN execution demands via wireless communication, e.g., Wi-Fi or cellular networks (5G). The binary variable $x_{i,e}$ denote the connection association between device i and edge server e, which is given by

$$x_{i,e} = \begin{cases} 1, \text{if device } i \text{ connects to edge server } e, \\ 0, \text{otherwise.} \end{cases} \tag{1}$$

It is notable that each device can achieve connection with no more than one edge server within its communication coverage at one time. Meanwhile, the edge server can handle multiple or zero requests from different IoT devices simultaneously.

At each edge server, it can only deploy limited DNN models to serve requests with finite storage capacity. Therefore, let $a_e = \emptyset \cup \{d_1, d_2, \ldots, d_{l_e}\}$ represent the DNN model placement decision of edge server e, where l_e

is the size of deployed DNN models. Note that the size of deployed DNN models may be zero on edge servers when there is no request to the edge server. From the service provider's perspective, the key consideration is to increase the profit. For example, when some edge servers have no requests, they may turn into an idle state to boost the whole system utilization.

Definition 1 (Placement Decision Profile). *For IoT devices $\mathcal{I} = \{i_m | m = 1, 2, \ldots, M\}$, edge servers $\mathcal{E} = \{e_n | n = 1, 2, \ldots, N\}$, and DNN models $\mathcal{D} = \{d_o | o = 1, 2, \ldots, O\}$, the placement decision profile of all edge servers is denoted as $\mathcal{A} = \{a_1, a_2, \ldots, a_e\}$.*

Remark 1. *In the constructed application scenario for DNN model placement in edge intelligence-empowered IoT, the placement decision of DNNs are scheduled in advance with the offline manner. Firstly, DNN model placement involves determining the optimal locations for deploying DNN models on edge servers. This process requires a comprehensive understanding of the edge network environments, including the capabilities of edge servers, the demands of IoT devices, and the specific requirements of DNN computations. Such information needs to be gathered and analyzed in advance to ensure optimal decision-making. Moreover, DNN model placement is a complex mission that involves significant computational overhead. Given the potentially large scale of the network and high dimensionality of the decision space, finding the optimal solution can be time-consuming, which might not be feasible in real-time scenarios.*

Furthermore, the variable $y_{e,i^j} \in \{0, 1\}$ denotes whether the required model d_i^j is deployed on edge server e. If $y_{e,i^j} = 1$, the request will be processed on the edge server. Otherwise, the request r_i^j will be executed locally with less computing power and higher battery burden. The detailed presentation is give by

$$y_{e,i^j}(a_e) = \begin{cases} 1, \text{if } d_i^j \in a_e, \\ 0, \text{otherwise.} \end{cases} \tag{2}$$

Therefore, there are two cases when $y_{e,i^j}(a_e)$ equals 1 or 0, which are discussed in details as follows.

Case 1. *When $y_{e,i^j} = 1$, that means the requested DNN is deployed on edge server e. Such that the DNN execution is conducted on the edge server instead of on-device processing, where the processing delay for request r_i^j can be calculated as*

$$t_i^j = \sum_{e \in \mathcal{E}} x_{i,e} \frac{c_i^j}{f_e}, \tag{3}$$

where f_e is the available computation resources of edge server e. Note that the typical computation measurements for the DNN model are the number of FLOPs, which is mainly incurred by the layer execution (e.g., convolution operation, FC operation, and self-attention operation) [35]. Therefore, in this paper, we perform the number of FLOPs as the metric to represent the number of required computations for DNN execution.

Case 2. *Otherwise, if $y_{e,i^j} = 0$, that means the requested model is missed on edge server e. In this case, the request r_i^j has to perform locally. The latency is given by*

$$t_i^j = \frac{c_i^j}{g_i}, \tag{4}$$

where g_i is the available computation resources on device i. Especially, if the placement decision a_e is $\emptyset$, that is, there is no model placed on edge server e. Therefore, the IoT device who tends to generate to edge server e will perform the DNN execution locally. The processing time is the same as Eq. (4).

Based on above two cases, the total processing of request r_i is

$$t_i = \sum_{i \in \mathcal{I}} \left[x_{i,e} \frac{\sum_{j=1}^{k_i} y_{e,i^j} c_i^j}{f_e} + \frac{\sum_{j=1}^{k_i} (1 - y_{e,i^j}) c_i^j}{g_i} \right]. \tag{5}$$

Although when the edge servers fail the DNN request where IoT devices will perform locally, we also consider this situation not improve the both QoE of users and the benefit of edge servers. The local processing time conducted at devices is referred to as the punishment for edge servers. Therefore, for edge server e, the total execution latency is based on the IoT devices that request to edge server, which is derived as

$$T_e = \sum_{i \in \mathcal{I}} t_i. \tag{6}$$

3.4 Energy Consumption Model

During the DNN execution, the energy cost is another significant factor for edge servers. The power consumption of edge server e is influenced by the workload level and occupied computing resources. First, we derive the presentation of the power of edge servers for DNN execution, which is given by [36]

$$p_e = \begin{cases} p_e^{idle} + (p_e^{max} - p_e^{idle}) \frac{\sum_{i \in \mathcal{I}} x_{i,e} \sum_{j=1}^{k_i} y_{e,ij}}{h_e}, \text{if } \sum_{i \in \mathcal{I}} x_{i,e} \sum_{j=1}^{k_i} y_{e,ij} \leq h_e, \\ p_e^{max}, \text{otherwise.} \end{cases} \tag{7}$$

where p_e^{idle} is the power of edge server in idle state, p_e^{max} is the power of edge server in full workload state, and h_e is the maximum workload threshold of edge server e. When the deployed model exceeds the maximum threshold of edge servers, the consumed power is the full workload state for edge servers. Otherwise, the power will depend on the current workload level. On the other hand, if the required model is non-deployed on edge servers, even though there are no computing, memory, and storage resource consumption and incurred power cost, it is not the optimal solution for edge servers. When y_{e,i^j} is 0 (i.e., $d_i^j \notin a_e$), the local processing will be conducted. Such that the energy cost for performing on-device is given by

$$p_i = \sum_{e \in \mathcal{E}} \sum_{j=1}^{k_i} (1 - y_{e,i^j}) \kappa_i, \tag{8}$$

where κ_i is the power of device i on execution.

Overall, for edge server e, the energy consumption for processing all requests (both perform on edge server and IoT device) can be calculated as follows

$$E_e = (p_e + p_i) T_e. \tag{9}$$

3.5 Edge Server Benefit Model

By considering the resource consumption on edge servers, we formulate the edge server benefit model. This model is primarily dependent on two components: the revenue of edge servers by placing DNN models to serve users and the placement cost of edge server to utilize computing and memory resources.

First, the revenue of edge server e is based on the user who consumes resources to perform DNN computations on the edge server. Users pay a fee for the DNN-enabled services, and the fee contributes to the revenue of edge servers. The revenue of edge server is based on two key components, i.e., the fixed profit and variable income that is a function of the resources consumed by users performing DNN computations, which is derived by [37]

$$G_e(a_e) = (R + \alpha \sum_{i \in \mathcal{I}} \sum_{j=1}^{k_i} c_i^j + \beta \sum_{i \in \mathcal{I}} \sum_{j=1}^{k_i} m_i^j) \frac{\sum_{i \in \mathcal{I}} x_{i,e} \sum_{j=1}^{k_i} y_{e,i^j}(a_e)}{\sum_{i \in \mathcal{I}} k_i}, \tag{10}$$

where R is a fixed profit representing a base income that the server receives irrespective of its load. The variable income is a direct function of the resources consumed by the users. α is the price of processing resources and β is the price of memory usage. This revenue model developed a comprehensive understanding for the financial dynamics of edge server utilization. The more resources a user consumes, the more they contribute to the variable income of the edge server. This creates an incentive for the edge server to accommodate as many users as it can handle, maximizing the utilization of its resources.

On the other hand, during the DNN execution, edge servers also take on the placement cost, including processing latency, energy cost, memory footprint consumption, and storage occupancy of deployed DNN model parameter files. In the placement cost of latency and energy cost, the failed requests are also taken into consideration as the punishment. Thus, the placement cost is

$$C_e(a_e) = T_e(a_e) + E_e(a_e) + \sum_{i \in \mathcal{I}} x_{i,e} \sum_{j=1}^{k_i} y_{e,i^j}(a_e) m_i^j + \sum_{d \in a_e} s_d(a_e), \tag{11}$$

where $s_d(a_e)$ is the storage cost of placed model on edge server e. Such that, the system benefit of edge server e is calculated as the revenues minus placement cost, i.e.,

$$v_e(a_e) = G_e(a_e) - C_e(a_e). \tag{12}$$

By combining these two models, we derive the overall benefit of edge servers. Specifically, the sum of benefit is essentially revenue obtained from the edge server's usage minus the costs associated with its placement and maintenance. The benefit model $V(\mathcal{A})$ of all edge servers is calculated by

$$V(\mathcal{A}) = \sum_{e \in \mathcal{E}} v_e(a_e). \tag{13}$$

3.6 Problem Formulation

Considering the defined decision profile, the main objective is to maximize the system benefit for all edge servers. The optimization problem is derived as

$$\mathcal{P}1: \max_{\mathcal{A}=\{a_1,a_2,\ldots,a_e\}} V(\mathcal{A}) \tag{14a}$$

$$s.t. \quad \sum_{e \in \mathcal{E}} \sum_{i \in \mathcal{I}} x_{i,e} \leq 1, \tag{14b}$$

$$a_e \in \emptyset \cup \{d_o | d_o \in \mathcal{D}\}, \tag{14c}$$

$$\sum_{d \in a_e} s_d \leq S_e, \tag{14d}$$

where constraint (14b) denotes that each IoT device requests no more than one edge server at a time that avoids the collision caused by several edge servers processing requests from the same device simultaneously, constraint (14c) represents the space of decisions $\mathcal{A}$, and constraint (14d) indicates that the placed DNNs on each edge servers do not exceed the capacities of edge servers.

3.7 Problem Analysis

We analyze the hardness of formulated optimization problem.

Theorem 1. *The formulated DNN placement optimization problem $\mathcal{P}1$ is $\mathcal{NP}$-hard.*

Proof. To prove the problem $\mathcal{P}1$ is $\mathcal{NP}$-hard, we conduct the polynomial-time reduction from the classical knapsack problem (KP), which is known to be $\mathcal{NP}$-hard [38]. The KP is described as follows: given the number of products N and a knapsack with capacity W, each product n has its weight w_n and value v_n. The problem is which products to pack into the knapsack such that the total weights of these products do not exceed the knapsack capacity W and the total value is maximum. The decision variable x_n denotes whether the product n

is put into the knapsack, i.e., $x_n = 1$ means that put the product n into the knapsack. Otherwise, product n is discarded. Thus, the KP is formulated as in the following.

$$\mathcal{P}2: \max_{\{x_1,x_2,\ldots,x_N\}} \sum_{i=1}^{N} x_i v_i \tag{15a}$$

$$s.t. \quad x_i \in \{0,1\}, \forall i = 1,2,\ldots,N, \tag{15b}$$

$$\sum_{i=1}^{N} x_i w_i \leq W, \tag{15c}$$

where constraint (15b) denotes each product that uses no more than once, and constraint (15c) indicates the total weight of the placed products cannot exceed the capacity of the knapsack.

Now we reduce an arbitrary problem $\mathcal{P}2$ instance to an instance of $\mathcal{P}1$. Let's treat product i as a DNN model and a knapsack as an edge server. We extend the problem of one knapsack to M knapsack problems. The extended M knapsack problem is the sum of the problem of one knapsack, where each knapsack is independent. Moreover, the total values of one knapsack are treated as the benefit of one edge server. The constraint (15b) can be aligned with constraint (14c) through an equivalent representation, i.e., a set of products to put into a knapsack. The constraint (15c) and constraint (14d) are equivalent and both mean that cannot exceed the capacity. Note that such reduction is obviously in a polynomial time. Thus, the problem $\mathcal{P}1$ can be reduced from the KP problem, which is also $\mathcal{NP}$-hard. □

4 DNN Model Placement Game

Considering the above-mentioned problem is $\mathcal{NP}$-hard, to solve the challenges achieving maximum system benefit, the game theory-based approach is employed. In specific, game theory is a promising optimization technique, due to its ability to model and analyze strategic interactions between multiple actors. In the context of DNN placement in edge intelligence environments, the formulated optimization problem Eq. (14) involves multiple players, i.e., edge servers, competing for limited resources. One key advantage of using game theory is its ability to capture the strategic decision-making process of self-interested player. Compared to classical optimization techniques, e.g., convex optimization [39], which are powerful tools for solving optimization problems, game theory provides a more comprehensive approach that incorporates strategic decision-making and considers the dynamics and interactions between multiple actors. For instance, convex optimization techniques may not fully exploit efficient methods in the distributed environments with resource competition between multiple edge servers. Therefore, in this section, we first formulate the DNN model placement problem as an exact potential game, then theoretic game property analysis is derived in detail.

4.1 Game Formulation

The DNN model placement game (DMPG) is modeled as

$$\mathcal{G} = \{\mathcal{E}, \mathcal{A}, \{\mathcal{U}_e\}_{e\in\mathcal{E}}\}, \tag{16}$$

where $\mathcal{E} = \{1,2,3,\ldots,N\}$ is set of game players (i.e., edge server), $\mathcal{A}$ is the decision space, $\{\mathcal{U}_e\}$ is the utility function of player e. Specifically, the decision space $\mathcal{A}$ is the set of decisions of all players. The decision space for player e is $\mathcal{A}_e$, and each decision in decision space is $a_e \in \emptyset \cup \{d_o | d_o \in \mathcal{D}\}$. In addition, let a_{-e} denote the decision of all players except player e where $a_{-e} = \{a_1, a_2, \ldots, a_{e-1}, a_{e+1}, \ldots, a_N\}$. Therefore, the $\mathcal{A}$ can be derived as

$$\mathcal{A} = \prod_{e=1}^{N} \mathcal{A}_e = \mathcal{A}_1 \otimes \mathcal{A}_2 \otimes \cdots \otimes \mathcal{A}_N, \tag{17}$$

where $\otimes$ is the Cartesian Product. Moreover, the utility function of player e is defined as

$$\mathcal{U}_e(a_e, a_{-e}) = v_e(a_e, a_{-e}). \tag{18}$$

From Eq. (18), the utility value of player e includes two parts, i.e., the revenue of edge server and the placement cost. The aim of each edge server is to maximize its utility value to get optimal benefit.

4.2 Analysis of Nash Equilibrium

Definition 2 (Nash Equilibrium). *A DNN placement decision profile $a^* = (a_1^*, a_2^*, \ldots, a_N^*)$ of all edge servers is an Nash Equilibrium if and only if no player can get a better utility value by unilaterally changing its own decision while other players keep their decisions the same, i.e.,*

$$\mathcal{U}_e(a_e^*, a_{-e}^*) \geq \mathcal{U}_e(a_e, a_{-e}^*), \forall e \in \mathcal{E}, a_e^* \neq a_e. \tag{19}$$

Nash Equilibrium (NE) enables that decision $a_e^* \in a^*$ for edge server e is the best decision for DNN model placement. In other words, it is a situation where each player's decision is optimal, given the decisions of the other players. That is, no edge server can benefit by changing their strategy if the strategies of the others remain unchanged. In view of this, ensuring the existence of NE is the focus to derive the optimal solutions to the formulated problem. As a typical classification of non-cooperative games, the potential game possesses at least one pure-strategy NE [40]. Therefore, the definition of EPG and analysis of EPG properties are discussed in the following.

Definition 3 (Exact Potential Game). *Any proposed game $\mathcal{G}$ is an EPG if and only if an unique global exact potential function Φ exists such that for $e \in \mathcal{E}$, it satisfies*

$$\mathcal{U}_e(a_e', a_{-e}) - \mathcal{U}_e(a_e, a_{-e}) = \Phi(a_e', a_{-e}) - \Phi(a_e, a_{-e}), \forall a_e, a_e' \in \mathcal{A}_e. \tag{20}$$

Definition 3 indicates that the property of potential function. If the formulated game is proven to be an EPG, it can be concluded there exists NE. To this end, we first give the definition of the exact potential function $\Phi(\mathbf{a})$, which is

$$\Phi(a_e, a_{-e}) = \sum_{e \in \mathcal{E}} v_e(a_e, a_{-e}). \tag{21}$$

The potential function $\Phi(\mathbf{a})$ is the sum of the utility function values of all edge servers.

Theorem 2. *The formulated DMPG is an EPG with an exact potential function Φ and exists at least one pure-strategy NE.*

Proof. Given the exact potential function Φ and decision profile a_e and a_e' ($\forall a_e, a_e' \in \mathcal{A}_e, e \in \mathcal{E}$), we have

$$\begin{aligned} &\Phi(a_e', a_{-e}) - \Phi(a_e, a_{-e}) \\ &= \sum_{e \in \mathcal{E}} v_e(a_e', a_{-e}) - \sum_{e \in \mathcal{E}} v_e(a_e, a_{-e}). \end{aligned} \tag{22}$$

Based on the Eq. (12), the benefit of each edge server depends on the revenue and placement cost. Note that for edge servers, the total sum of benefits can be divided into two parts: i) the current edge server e; ii) and other edge servers. Such that, the Eq. (22) can also be presented as

$$\begin{aligned} &\Phi(a_e', a_{-e}) - \Phi(a_e, a_{-e}) \\ &= G_e(a_e', a_{-e}) - C_e(a_e', a_{-e}) + \textstyle\sum_{j \in \mathcal{E}, j \neq e} \left[G_j(a_j, a_{-e}) - C_j(a_j, a_{-e}) \right] \\ &- \left\{ G_e(a_e, a_{-e}) - C_e(a_e, a_{-e}) + \textstyle\sum_{j \in \mathcal{E}, j \neq e} \left[G_j(a_j, a_{-e}) - C_j(a_j, a_{-e}) \right] \right\}. \end{aligned} \tag{23}$$

When other edge servers ($\forall j \in \mathcal{E}$ and $j \neq e$) under the decision profile a_e and a'_e, the benefits of them under its own decision a_j are unchanged. Such that, we further have

$$\begin{aligned}
&\Phi(a'_e, a_{-e}) - \Phi(a_e, a_{-e}) \\
&= G_e(a'_e, a_{-e}) - C_e(a'_e, a_{-e}) - [G_e(a_e, a_{-e}) - C_e(a_e, a_{-e})] \\
&= v_e(a'_e, a_{-e}) - v_e(a_e, a_{-e}) \\
&= \mathcal{U}_e(a'_e, a_{-e}) - \mathcal{U}_e(a_e, a_{-e}).
\end{aligned} \tag{24}$$

There is $\Phi(a'_e, a_{-e}) - \Phi(a_e, a_{-e}) \equiv \mathcal{U}_e(a'_e, a_{-e}) - \mathcal{U}_e(a_e, a_{-e})$. Therefore, we prove that DMPG is an EPG, and every EPG possesses at least one pure-strategy NE according to [40].

□

Consequently, with the finite improvement property [41], it can conduct an iterative procedure within a finite number of iterations to find the NE points to acquire sub-optimal decisions. Through local and global interactions with the environment, each player can only update its own decisions of model placement in a decentralized manner. In the next section, a decentralized EPG-based DNN placement algorithm will be described in detail to solve the formulated DMPG problem.

5 An EPG-based DNN Placement Algorithm

5.1 EPOL Overview

Considering the DNN model placement problem is proven as $\mathcal{NP}$-hard, we then design an EPG-based DNN model placement algorithm, named EPOL, to achieve the maximal system benefit of edge servers. As illustrated in Fig. 2, the algorithm framework of EPOL is presented. EPOL consists of two stages, i.e., Initialization & Configuration and Runtime Execution. In the Initialization & Configuration stage, the EPOL manages the randomized strategies of edge servers and pre-load the information of DNN models, including computation cost, memory usage, and parameters storage of models. Afterwards, in the Runtime Execution, edge servers conduct the iterative process to acquire the optimal placement strategies with maximized system benefit under the competitions of decision update between other players. The details of EPOL algorithm is illustrated in the following.

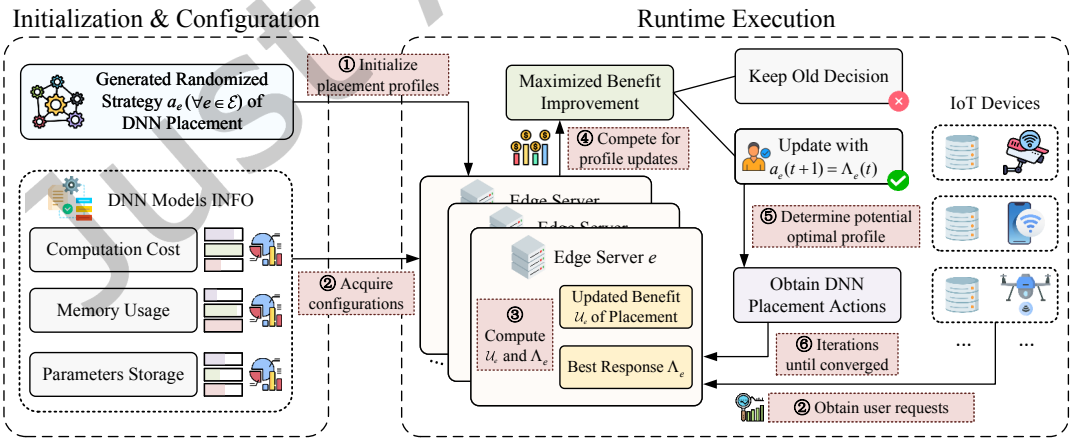


Fig. 2. The framework of EPOL.

5.2 The details of EPOL

In the EPOL, finite improvement property enables to be reached to the NE where each edge server will acquire an optimal DNN placement strategy to increase its benefit under a limited number of iterations. In each iteration, players will generate new decisions but only one play has the opportunity to update the decision. At the end of iterative steps, there is no edge server that can update its strategy to achieve higher benefits where no other edge servers change their own decisions, and a stable status called an NE is reached. Specifically, if edge servers continuously improve benefits by changing their strategies one by one, eventually they will reach a NE point where no more improvements are possible. Such finite improvement property ensures that players have no incentive to unilaterally change their decisions because any update would result in a decrease in the potential function and hence a worse outcome for at least one player.

Note that each edge server selected optimal decision updates mainly depends on the best response (BR), and the definition of BR is given as follows [42].

Definition 4 (Best Response). *Given the strategy a_{-e} of other players, the best response of player e is the set where the player can obtain the maximum utility value, which can be expressed as*

$$\Lambda_e(t) = \{a_e^* | a_e^* = \underset{a_e^* \in \mathcal{A}_e}{argmax}\ \mathcal{U}_e(a_e^*, a_{-e})\}. \tag{25}$$

The BR refers to a strategy that yields the highest possible payoff given the strategies chosen by other players in a game. The BR can be expressed as a function that maps the strategies chosen by other players to the optimal strategy for the player. It allows players to anticipate and respond to the actions of others, leading to the concept of NE, where no player has an incentive to unilaterally deviate from their strategy.

The details of the EPOL are presented in Algorithm 1. The information of IoT devices, edge servers, DNN models, and requests from devices are as the input. After the algorithm, the output is the best DNN placement decision of edge servers. The EPOL consists of two phases. First, in the initialization phase (lines 1 to 3), the number of iterations is set. The decision profile $\mathcal{A}$ is started with the randomized placement strategies and we also set the best decision $\mathcal{A}^*$ as empty. In the repeat phase, each edge server will perform iterative executions to update decisions. Specifically, in one iteration, edge server e will find all the devices that request to edge server e (i.e., $x_{i,e} = 1$), and the total sum of benefit under current decision $a_e(t)$ is calculated (lines 7 to 9). After that, edge server e searches the BR $\Lambda_e(t)$ according to Eq. (25) to obtain the maximal benefit value (lines 11). If the acquired BR $\Lambda_e(t)$ is the same as the current decision $a_e(t)$, edge server e will hold the old a_e to the next iteration. Otherwise, the $\Lambda_e(t)$ must be superior to current decision $a_e(t)$, and edge server e will compete to update its optimized result (lines 12). Then, each node will broadcast its maximal benefit $\mathcal{U}_e'$ to other nodes for competition. Generally, there are two ways to implement the data transmission, i.e., decentralized manner and centralized manner. In the decentralized manner, each node sends the data to each with peer to peer. On the other hand, the centralized manner enables the controller to manage the data transmission where controller collects all the data from edge servers and distribute them to target nodes. In this paper, to simplify the analysis, we assume that there is no transmission overhead for edge server to communicate with others. Note that, in EPOL, only one edge server gains the highest benefit improvement under old decision a_e and BR $\Lambda_e(t)$ that can win the competition and updates its own strategy with $a_e(t+1) = \Lambda_e(t)$, where $a_e(t+1)$ is the strategy in the next state (lines 14 to 16). On the other hand, the rest of edge servers that do not achieve the update opportunity continue will maintain the old decisions, i.e., $a_e(t+1) = a_e(t)$ (lines 18). When the old decision profile $a(t-1) = \{a_1(t-1), a_2(t-1), \ldots, a_N(t-1)\}$ is the same as current $a(t) = \{a_1(t), a_2(t), \ldots, a_N(t)\}$, the repeat phase will finish and the best decision profile $\mathcal{A}^*$ will be obtained.

Algorithm 1: EPG-based DNN Model Placement Algorithm (EPOL)

Input: $\mathcal{I}, \mathcal{E}, \mathcal{D}, r_i(\forall i \in \mathcal{I})$;
Output: $\mathcal{A}^* = \{a_1^*, a_2^*, \ldots, a_e^*\}$;
Initialize number of iterations T;
Initialize random decision $a_e, \forall e \in \mathcal{E}$;
Initialize $\mathcal{A}^* = \{\}$;
for *Iteration* $t = 1$ *to* T **do**
  for *Each edge server* $e \in \mathcal{E}$ **do**
    for *Each IoT device* $i \in \mathcal{I}$ **do**
      if $x_{i,e}$ *equals* 1 **then**
        Calculate the cumulative sum of benefit with decision a_e according to Eq. (18);
      end
    end
    Compute the BR $\Lambda_e(t)$ with maximal benefit value $\mathcal{U}_e'$ according to Eq. (25);
    if $\Lambda_e(t) \neq a_e$ **then**
      Broadcast the acquired maximal benefit $\mathcal{U}_e'$ to other edge servers to contend the decision update opportunity;
      if *e wins the competition with highest benefit improvement* **then**
        Update the decision $a_e(t+1) = \Lambda_e(t)$ based on BR;
      end
      else
        Stay the old decision $a_e(t+1) = a_e(t)$;
      end
    end
    else
      Stay the old decision $a_e(t+1) = a_e(t)$;
    end
    if $a(t-1)$ *and* $a(t)$ *are the same* **then**
      $\mathcal{A}^* = a(t)$;
      Stop the iteration and break;
    end
  end
end
Return $\mathcal{A}^*$;

5.3 Theoretical Analysis

5.3.1 Convergence analysis. DMPG as an EPG is convergent to an NE within a limited number of iterations with the finite improvement property. In this section, the convergence analysis of EPOL is derived to guarantee the algorithm performance. The upper bound of iteration times is held by Theorem 3.

Theorem 3. *To admit an NE, the iteration times* Ξ *of EPOL satisfies the following inequality, i.e.,*

$$\Xi \leq |\mathcal{E}| \left\lfloor \frac{R + |\mathcal{I}|k_{max}(\alpha c_{max} + \beta m_{max} - m_{min}) - T_{min} - E_{min} - l_{min}s_{min}}{Q_{min} + P_{min} + V_{min}} \right\rfloor. \tag{26}$$

Proof. Consider the best situation of placement decisions for edge servers, i.e., the edge servers can handle all requests from devices with the maximum revenue and minimum placement cost. Assume that $c_{max} \triangleq max_{\{i\in\mathcal{I},1\leq j\leq k_i\}} c_i^j$, $m_{max} \triangleq max_{\{i\in\mathcal{I},1\leq j\leq k_i\}} m_i^j$, , $E_{min} \triangleq min_{\{e\in\mathcal{E}\}} E_e$, and $s_{min} \triangleq min_{\{d\in a_e\}} s_d$. According to Eq. (21), the upper bound of potential function is derived as

$$\begin{aligned}
\Phi(\mathcal{A}) &= \sum_{e\in\mathcal{E}} R + \alpha \textstyle\sum_{i\in\mathcal{I}} \sum_{j=1}^{k_i} c_i^j + \beta \sum_{i\in\mathcal{I}} \sum_{j=1}^{k_i} m_i^j - (T_e + E_e + \textstyle\sum_{i\in\mathcal{I}} \sum_{j=1}^{k_i} m_i^j + \sum_{d\in a_e} s_d) \\
&\leq \sum_{e\in\mathcal{E}} R + \alpha \textstyle\sum_{i\in\mathcal{I}} \sum_{j=1}^{k_i} c_{max} + \beta \sum_{i\in\mathcal{I}} \sum_{j=1}^{k_i} m_{max} \\
&- (T_{min} + E_{min} + \textstyle\sum_{i\in\mathcal{I}} \sum_{j=1}^{k_i} m_{min} + \sum_{d\in a_e} s_{min}) \\
&\leq \sum_{e\in\mathcal{E}} R + |\mathcal{I}|k_{max}(\alpha c_{max} + \beta m_{max} - m_{min}) - T_{min} - E_{min} - l_{min}s_{min} \\
&= |\mathcal{E}| \left[R + |\mathcal{I}|k_{max}(\alpha c_{max} + \beta m_{max} - m_{min}) - T_{min} - E_{min} - l_{min}s_{min} \right].
\end{aligned} \tag{27}$$

Based on Theorem 2, after the decision changes from a_e to a'_e, the value of the potential function increases, which is presented by

$$\begin{aligned}
&\Phi(a'_e, a_{-e}) - \Phi(a_e, a_{-e}) \\
&= G_e(a'_e, a_{-e}) - C_e(a'_e, a_{-e}) - [G_e(a_e, a_{-e}) - C_e(a_e, a_{-e})] \\
&= (\alpha \textstyle\sum_{i\in\mathcal{I}} \sum_{j=1}^{k_i} c_i^j + \beta \sum_{i\in\mathcal{I}} \sum_{j=1}^{k_i} m_i^j) \frac{\sum_{i\in\mathcal{I}} x_{i,e} \sum_{j=1}^{k_i} [y_{e,i^j}(a'_e) - y_{e,i^j}(a_e)]}{\sum_{i\in\mathcal{I}} k_i} \\
&+ [T_e(a_e) - T_e(a'_e)] + [E_e(a_e) - E_e(a'_e)] + \sum_{i\in\mathcal{I}} x_{i,e} \sum_{j=1}^{k_i} m_i^j [y_{e,i^j}(a'_e) - y_{e,i^j}(a_e)] > 0
\end{aligned} \tag{28}$$

Let

$$Q_e \triangleq (\alpha \textstyle\sum_{i\in\mathcal{I}} \sum_{j=1}^{k_i} c_i^j + \beta \sum_{i\in\mathcal{I}} \sum_{j=1}^{k_i} m_i^j) \frac{\sum_{i\in\mathcal{I}} x_{i,e} \sum_{j=1}^{k_i} [y_{e,i^j}(a'_e) - y_{e,i^j}(a_e)]}{\sum_{i\in\mathcal{I}} k_i}, \tag{29}$$

$$P_e \triangleq [T_e(a_e) - T_e(a'_e) + E_e(a_e) - E_e(a'_e)], \tag{30}$$

$$V_e \triangleq \sum_{i\in\mathcal{I}} x_{i,e} \sum_{j=1}^{k_i} m_i^j [y_{e,i^j}(a'_e) - y_{e,i^j}(a_e)]. \tag{31}$$

Then we can derive the following inequality

$$\Phi(a'_e, a_{-e}) - \Phi(a_e, a_{-e}) = Q_e + P_e + V_e \geq Q_{min} + P_{min} + V_{min} \tag{32}$$

With above, it is proven that to reach an NE of EPOL the iteration times will not be more than $|\mathcal{E}| \left\lfloor \frac{R+|\mathcal{I}|k_{max}(\alpha c_{max}+\beta m_{max}-m_{min})-T_{min}-E_{min}-l_{min}s_{min}}{Q_{min}+P_{min}+V_{min}} \right\rfloor$.

□

5.3.2 Complexity analysis.

Theorem 4. *Given K requests on IoT devices, and Z decisions in game DMPG, the time complexity of EPOL algorithm is upper bound by*

$$O(\Xi|\mathcal{E}|(|\mathcal{I}|K + Z + |\mathcal{E}|)) \tag{33}$$

Proof. For Algorithm 1, in each iteration slot, all edge servers have to perform decision updates. For lines 6 to 9, the time complexity is $O(|\mathcal{I}|K)$ to handle all requests from the device to the edge server and then sum the benefit value. The BR computation in lines 11 is with the complexity of $O(Z)$. In lines 12 to 20, edge servers contend the decision update opportunity with the communication to other nodes, where the complexity is $O(|\mathcal{E}|)$. In addition, the algorithm will be terminated after Ξ iteration steps. Therefore, the time complexity of EPOL is $O(\Xi|\mathcal{E}|(|\mathcal{I}|K + Z + |\mathcal{E}|))$. □

6 Experimental Evaluation

In this section, we demonstrate the implementation and comparative evaluation of EPOL. All the experiments are conducted on a PC with Intel i7-8700 (6 cores, 3.20GHZ), 32GB DRAM, and NVIDIA GeForce GTX 1080, and Python 3.9.5 and PyTorch 1.12.0 are employed as the simulation environment. The parameters are presented in TABLE 2.

Table 2. Experiment Parameters

Parameter	Value
Number of IoT devices	[250,300,350,400,450,500]
Number of edge servers	[30,60,90]
Computing capacity of edge server	9e12 FLOPS [43]
Computing capacity of device	5e9 FLOPS [43]
Placement storage capacity of edge server	[300,600,900,1200,1500,1800] MB
Max power of edge server	300 Watt [39]
Min power of edge server	100 Watt [39]
Execution power of device	6.6 Watt
Number of requests	[250,500,750,1000,1250,1500]

6.1 Dataset

We utilize the real-world EUA dataset [41] of base station and end-user within Metropolitan Melbourne, Australia, covering over $9000km^2$. In the experiments, the region of high intensive density in the Melbourne CBD area is adopted, as shown in Fig. 3 in latitude and longitude. Specifically, the red dots represent the location of end-users, and the blue markers are the location of base stations. Statistically, a total of 816 end-users and 538 base stations are covered in the selected Melbourne CBD area.

6.2 Experiment Settings

To evaluate the performance of the proposed EPOL, we select 10 classical DNN models, i.e., AlexNet, VGG-16, ResNet-101, GoogleNet, DenseNet-201, ConvNet, MobileNet V2, Inception V3, Vision Transformer-B, and Swin Transformer-B. The details of required computing resources, memory usage, and storage cost are illustrated in TABLE 3. In addition, to verify the effectiveness of EPOL in the DNN models placement to meet the real-world environments, the different versions of DNNs are considered in the experiments. Specifically, for each DNN, it has multiple model versions with different performance to meet practical requirements of users (e.g., response

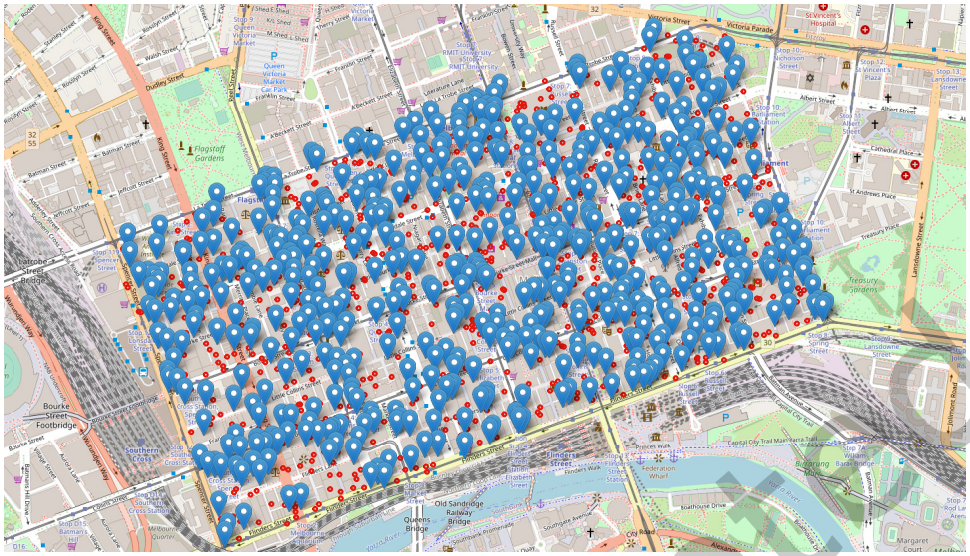


Fig. 3. The distribution of end-users and base stations in Melbourne CBD area of EUA dataset.

time, computation overhead, and battery life) [44]. To this end, based on these 10 conventional DNN models, the experiments let each DNN spawns 10 types of model versions with different computation cost, memory usage, and parameter storage. Each version is created by compressing the computation cost, memory usage, and parameter storage of the original model by a random proportion (compression ratio from 0 to 66.7%) [45]. Therefore, there are 100 types of DNN models in total for edge servers to decision the approximate placement strategies.

Table 3. Details of selected DNN models (224 × 224 resolutions).

Model	Computing cost (GFLOPs)	Memory usage (MB)	Parameter storage (MB)
AlexNet	0.72	4.19	244.40
VGG-16	15.5	109.39	553.42
ResNet-101	7.84	161.75	178.2
GoogleNet	1.51	30.03	52.02
DenseNet-201	4.37	219.59	80.06
ConvNext	0.65	256.31	354.37
MobileNet V2	0.32	74.25	14.02
Inception V3	2.85	36.65	108.65
Vision Transformer-B	0.17	119.47	346.27
Swin Transformer-B	0.08	210.55	351.07

In addition, EPOL is compared with three existing algorithms as follows.

- **BRKGA** [26]. BRKGA is a genetic-based algorithm that is utilized to solve the service placement problems, which is based on the biased random-key chromosomes presentation mechanism and an elite strategy to improve the generation performance.

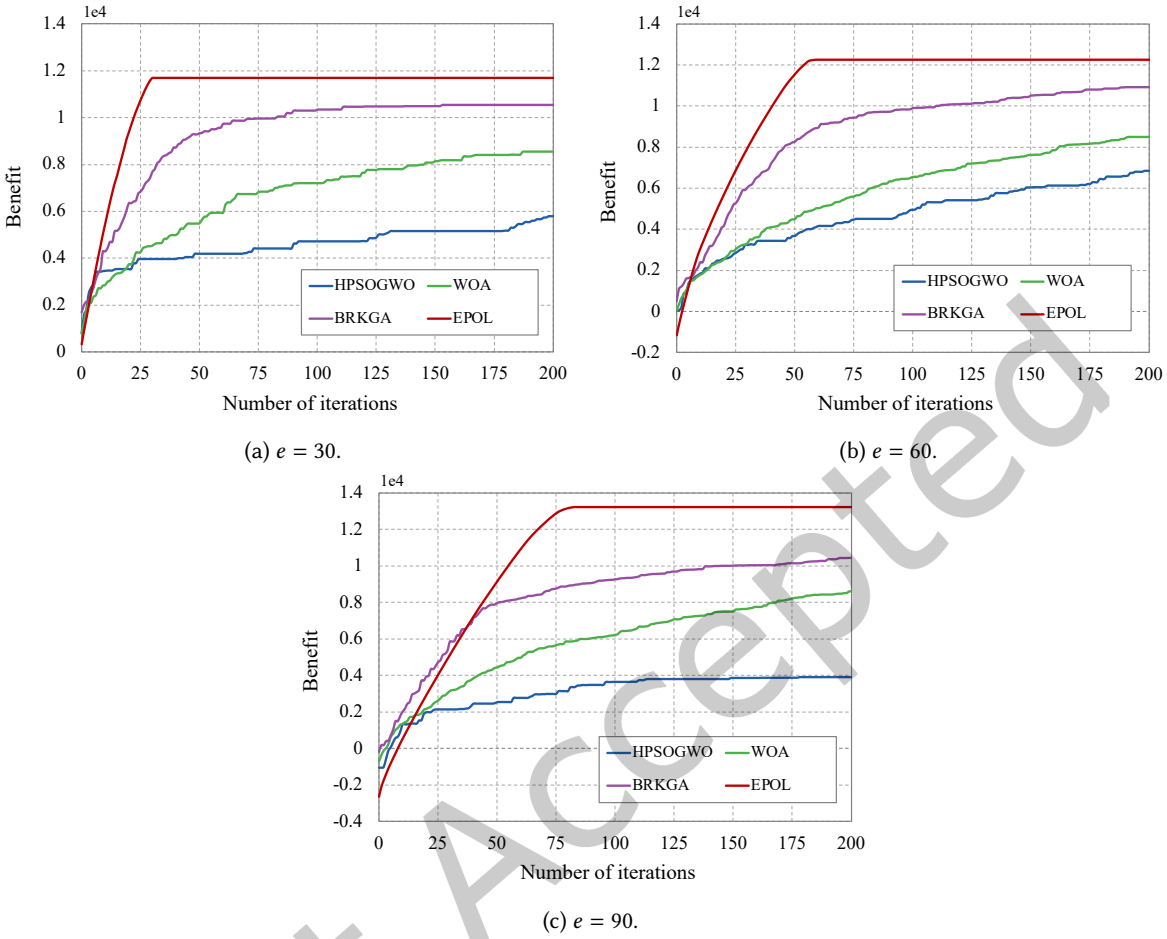
- **WOA** [46]. WOA is a heuristic algorithm that traditionally solves feature selection problems. With the nature-inspired meta-heuristic optimization abilities and discrete search characteristics of binary space, WOA can also be effective for model placement problems.
- **HPSOGWO** [47]. HPSOGWO is a hybrid particle swarm optimization (PSO) and grey wolf optimizer (GWO) based algorithm that can improve the ability of exploitation in PSO with the ability of exploration in GWO. With the combination of PSO and GWO, HPSOGWO outperforms the pure PSO and GWO in solution quality and stability.

In the experiments, four algorithms are initialized with a random decision and then conduct the iterative steps to obtain optimal DNNs placement results within finite steps. Specifically, at the starting point of four algorithms, the randomized numbers of deployed DNNs are derived first. Afterwards, based on the deployed numbers of DNNs, the valid placement decisions of four algorithms are generated where iteration step $t = 0$. Finally, each approach is finished after 200 steps. Moreover, we assume that IoT devices will generate their requests to its nearest edge server in the experiments.

6.3 Performance Analysis

6.3.1 Convergence performance. Fig. 4 demonstrates the convergence performance of four algorithms when e=30, 60 and 90, respectively. As can be seen from Fig. 4a, when the number of edge servers is 30, EPOL reaches convergence after 32 iterations, while BRKGA reaches convergence after 152 iterations. In addition, the convergence performance of WOA and HPSOGWO is not very stable. After 200 iterations, the edge server benefit of EPOL is 101%, 37%, and 11% higher than HPSOGWO, WOA, and BRKGA, respectively. As shown in Fig. 4b, when the number of edge servers is 60, EPOL converges after 59 iterations, and the benefit of edge servers is increased by 5% compared with that when e = 30. Moreover, after 200 iterations, the benefit of EPOL achieves 79%, 44%, and 12% higher than HPSOGWO, WOA, and BRKGA, respectively. Moreover, Fig. 4c shows the convergence performance of the four algorithms when the number of edge servers is further expanded to 90. Specifically, EPOL converges when the number of iterations is 83. After 200 iterations, the benefit of EPOL gains 240%, 54% and 27% increase than HPSOGWO, WOA, and BRKGA. In addition, compared with e=30 and e=60, the benefit of EPOL increased by 8% and 13%, respectively. In a word, it can be observed that the converged iterations of EPOL increase linearly with the expansion of the number of edge servers, and with the increase of edge servers, the gains also increase. This is because based on the finite improvement property, at each iteration, there is one and only one edge server that will update the current policy until the algorithm ends.

6.3.2 Performance with different number of IoT devices. Fig. 5 presents the comparison of the effects of four algorithms on the benefit of edge servers by the number of different IoT devices under e=30, 60, and 90 respectively. From the figures, it can be found that the benefit increases linearly as the number of devices increases. Specifically, Fig. 5a shows the revenue of the four algorithms when e=30. When the number of devices is 500, the benefit of EPOL is 6%, 13%, and 3% higher than HPSOGWO, WOA, and BRKGA, respectively. When the number of devices went from 250 to 500, the benefit of EPOL increased by 52%. As shown in Fig. 5b, compared with e=30, when the number of edge servers is 60, the benefit of EPOL increases by 33%. This is because as the number of edge servers increases, the diversity of model placements makes it more likely that end users will get a better service experience. When the number of devices reaches 500, EPOL achieves 18%, 14%, and 3% higher on benefits than HPSOGWO, WOA, and BRKGA. In addition, Fig. 5c shows the payoff of the four algorithms when e=90. When the number of devices reaches 500, EPOL gains 108%, 16%, and 3% higher on benefit than HPSOGWO, WOA, and BRKGA, respectively. Therefore, with the increase of IoT devices, the benefit of edge servers is increasing. When the number of edge servers expands, it can further improve the benefit. In conclusion, EPOL consistently outperforms the baseline methods across different number of IoT devices, demonstrating its efficiency and scalability. The benefits of edge servers increase with the rise of IoT devices, and EPOL is well-equipped to optimize benefits.



(a) e = 30. (b) e = 60.

(c) e = 90.

Fig. 4. Convergence performance under different number of edge servers (e is the number of edge servers).

6.3.3 Performance with different number of requests. Fig. 6 shows the comparison of four algorithms with the different number of requests for the benefit under e=30, 60, and 90, respectively. As can be observed in Fig. 6a, when the number of requests reaches 1000, EPOL can obtain the maximum payoff, and it is 11%, 10%, and 4% higher than HPSOGWO, WOA, and BRKGA, respectively. As the number of requests gets larger, however, the payoff becomes volatile. This is because as the diversity of IoT device requests for DNN tasks increases, it becomes more challenging to maximize edge server revenue within a limited placement capacity. Fig. 6b shows the performance of four algorithms when e=60. Specifically, when the number of requests reaches 1000, EPOL can gain 20%, 17%, and 7% higher benefits than HPSOGWO, WOA, and BRKGA, respectively. When the number of requests reaches 1500, the edge server payoff of EPOL is 3%, 26%, and 8% better than BRKGA, WOA, and HPSOGWO. In addition, Fig. 6c shows the performance when e=90. When the number of requests per device reaches 1500, EPOL achieves the maximum revenue and is 116%, 30%, and 14% higher than HPSOGWO, WOA, and BRKGA. From the above, it can be seen that the benefits are increasing even as the number of requests increases. However, when the number of device requests reaches a certain condition, the revenue of the edge

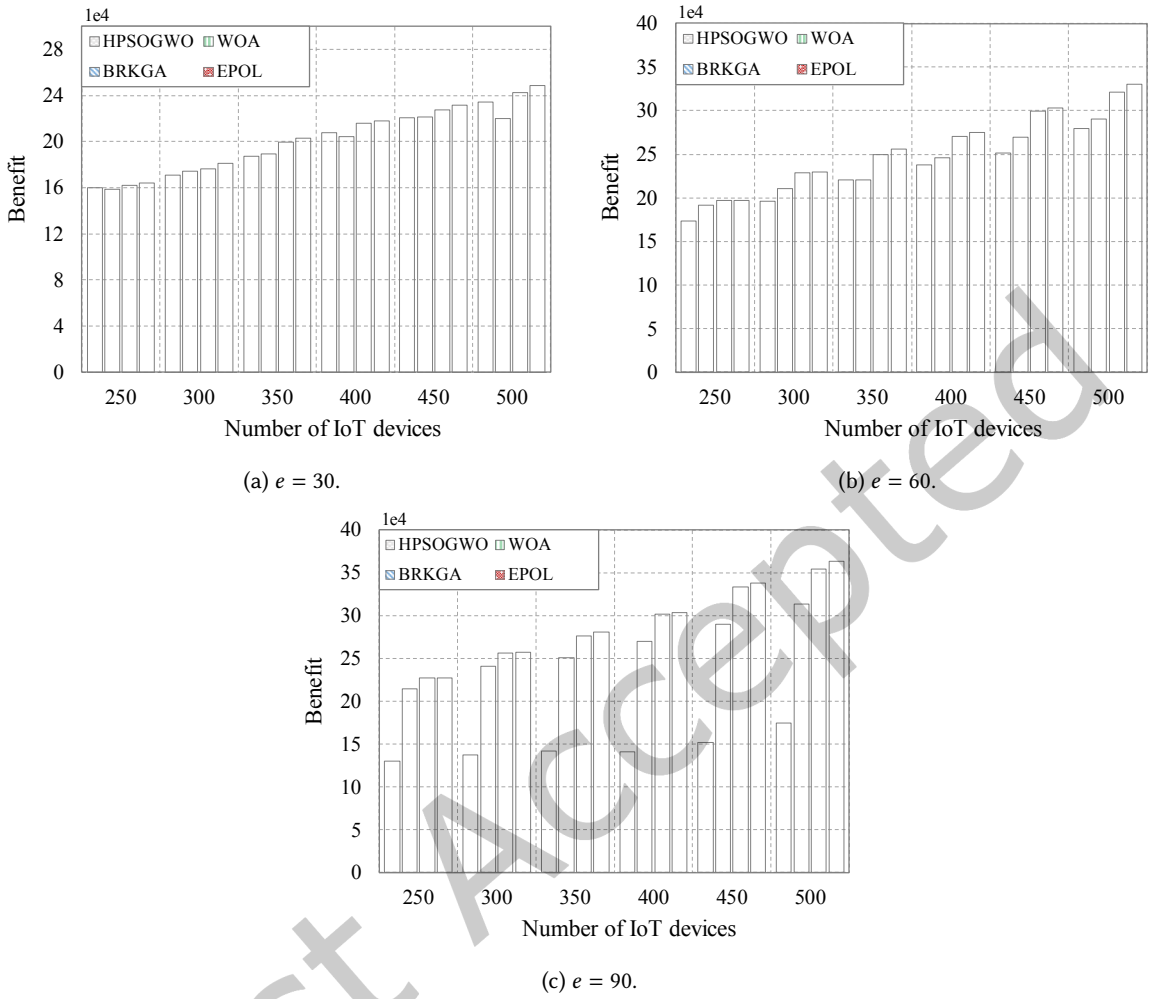
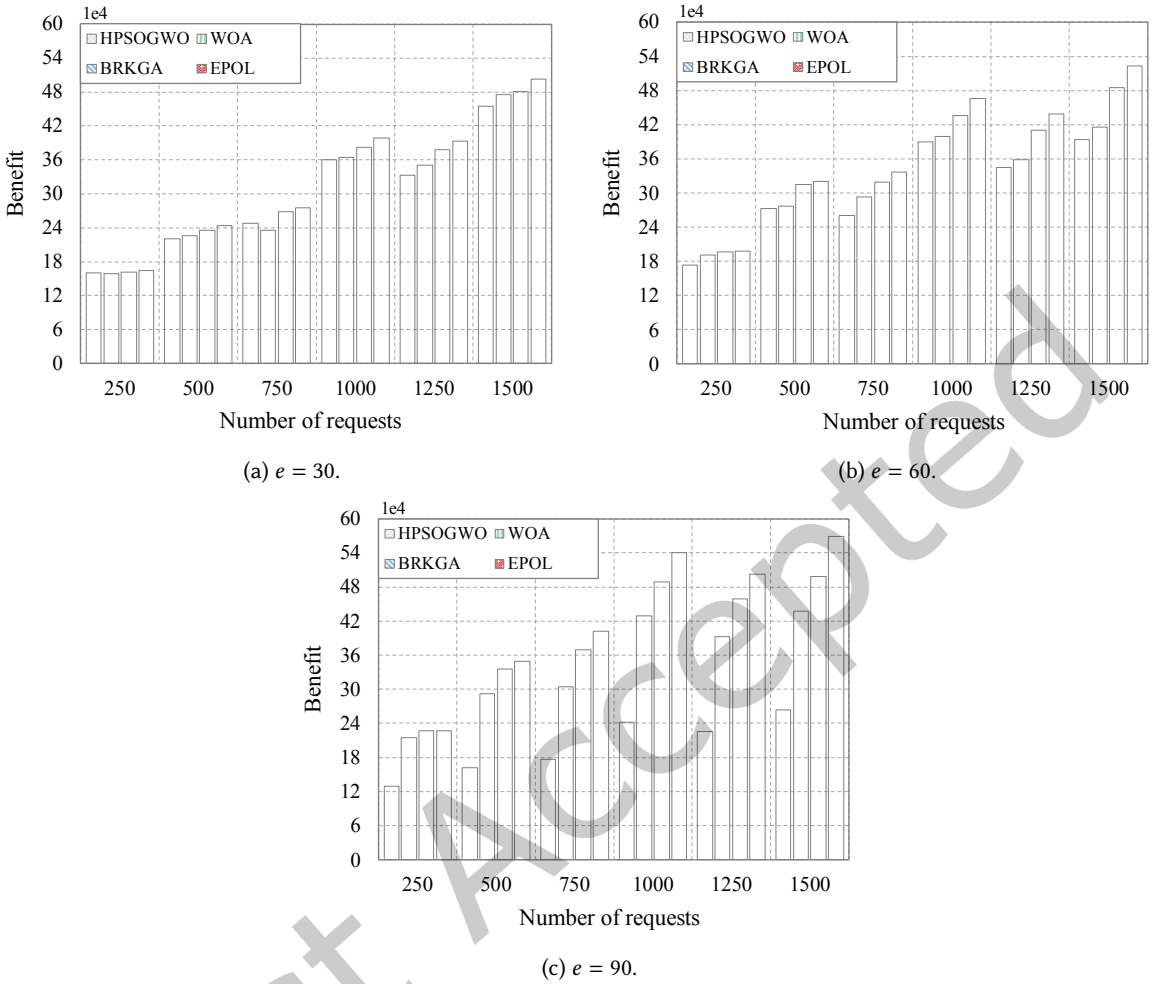


Fig. 5. Performance of benefit with number of IoT devices under different number of edge servers (e is the number of edge servers).

server will become unstable. As a result, EPOL demonstrates effective performance in optimizing edge server benefits. While the benefits increase with the number of device requests, the revenue becomes unstable beyond a certain threshold. Nonetheless, EPOL consistently outperforms the baselines, showcasing its efficiency and adaptability.

6.3.4 Performance with different placement capacities. Fig. 7 illustrates the comparison of the impact of different edge server placement capabilities on benefit under e=30, 60, and 90. Specifically, Fig. 7a shows the benefit of the algorithms when e=30. When the placement capacity is 900 MB, the edge server payoff of EPOL is 44%, 32%, and 27% higher than HPSOGWO, WOA, and BRKGA, respectively. When the placement capacity of edge servers went from 900 to 1800 MB, EPOL's benefit increased by 35%. In addition, it can be seen from the figures that

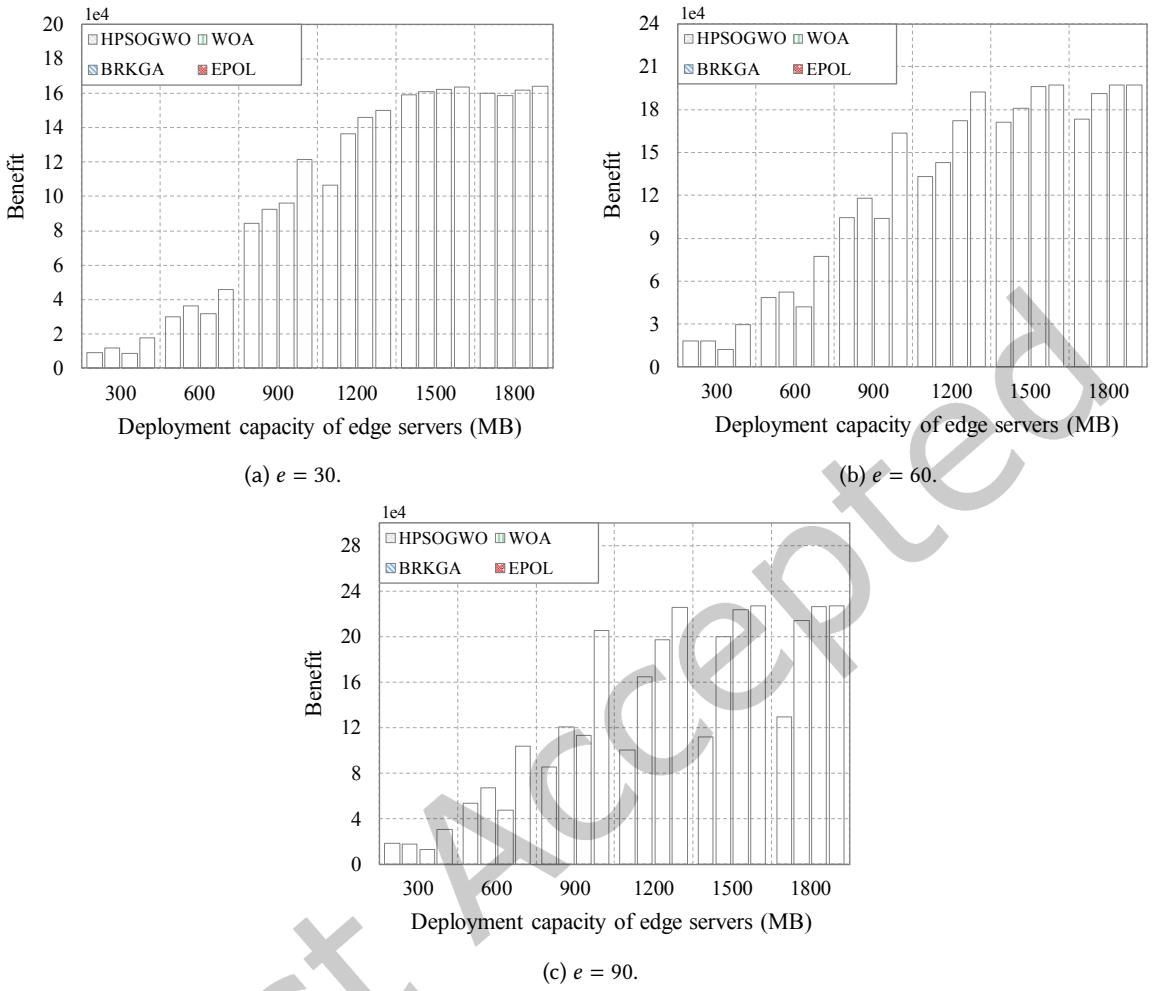


(a) e = 30. (b) e = 60.

(c) e = 90.

Fig. 6. Performance of benefit with number of requests under different number of edge servers (e is the number of edge servers).

with the increase in placement capacity, the gap in benefit between the four algorithms becomes smaller and smaller. When the placement capacity is 1800 MB, EPOL is only 2.3%, 3.2%, and 1.1% higher than HPSOGWO, WOA, and BRKGA. This is because when the placement capacity is large enough, more models can be deployed. As shown in Fig. 7b, when e=60 and placement capacity is from 300 to 1800 MB, the performance of EPOL is increased by 6 times. Specifically, EPOL acquires 45%, 34%, and 12% higher performance than HPSOGWO, WOA, and BRKGA, respectively, when deploying capacity is 1200 MB. Fig. 7c shows the payoff of four algorithms when e=90. As the placement capacity is 1200 MB, EPOL is 125%, 37%, and 14% better than HPSOGWO, WOA, and BRKGA, respectively. All in all, it can be observed that the payoff of EPOL goes up linearly with the increase of edge server placement capacity. However, when the placement capability reaches a certain threshold, the gains will tend to be stable. As a result, service providers need to balance storage overhead with edge server benefits.



(a) $e = 30$. (b) $e = 60$.

(c) $e = 90$.

Fig. 7. Performance of benefit with placement capacity under different number of edge servers (e is the number of edge servers).

7 Conclusion

In this paper, we investigated the DNN placement problem in edge intelligence environments. To maximize the benefit of edge servers from the perspective of service providers, an constrained optimization problem was first formulated, which is proven to be $\mathcal{NP}$-hard. Moreover, we transformed the original problem into an EPG, DMPG, and theoretical analysis is given to prove that the DMPG admits an NE. To find the near-optimal solutions, a decentralized DNN placement algorithm EPOL was proposed to support edge servers to perform the strategies independently. Convergence analysis and complexity analysis were possessed to guarantee the upper bounds of EPOL. At last, we conducted extensive experiments on the real-world EUA-dataset in Melbourne, Australia, to evaluate the performance of EPOL which was superior to three existing algorithms in the system benefit improvements of edge servers. In the future, we intend to extend our work to practical large-scale DNN placement

scenarios in edge computing systems. Furthermore, we will further take into account the mobility of IoT devices and personalized requirements from different users, and realize more robust placement approaches.

Acknowledgments

This research was supported by the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps under grant No. 2020DB005, the Postgraduate Research and Practice Innovation Program of Jiangsu Province under grant No. KYCX22_1219, and also by the Natural Science Foundation of Jiangsu Province of China under grant No. BK20211284. This work was also supported by the National Natural Science Foundation of China under Grant No.92267104.

References

[1] Junghoon Kim, Taejoon Kim, Morteza Hashemi, David J. Love, and Christopher G. Brinton. 2022. Minimum Overhead Beamforming and Resource Allocation in D2D Edge Networks. *IEEE/ACM Transactions on Networking* 30, 4 (2022), 1454–1468.

[2] Xiaokang Zhou, Wang Huang, Wei Liang, Zheng Yan, Jianhua Ma, Yi Pan, I Kevin, and Kai Wang. 2024. Federated distillation and blockchain empowered secure knowledge sharing for Internet of medical Things. *Information Sciences* 662 (2024), 120217.

[3] Yuanming Ren, Shihao Shen, Yanli Ju, Xiaofei Wang, Wenyu Wang, and Victor CM Leung. 2022. EdgeMatrix: A Resources Redefined Edge-Cloud System for Prioritized Services. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 610–619.

[4] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. 2023. A Survey on Vision Transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 1 (2023), 87–110.

[5] Haochen Hua, Yutong Li, Tonghe Wang, Nanqing Dong, Wei Li, and Junwei Cao. 2023. Edge computing with artificial intelligence: A machine learning perspective. *Comput. Surveys* 55, 9 (2023), 1–35.

[6] Yousaf Bin Zikria, Muhammad Khalil Afzal, Sung Won Kim, Andrea Marin, and Mohsen Guizani. 2020. Deep learning for intelligent IoT: Opportunities, challenges and solutions. *Computer Communications* 164 (2020), 50–53.

[7] Aakash Khochare, Aravindhan Krishnan, and Yogesh Simmhan. 2021. A Scalable Platform for Distributed Object Tracking Across a Many-Camera Network. *IEEE Transactions on Parallel and Distributed Systems* 32, 6 (2021), 1479–1493.

[8] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen. 2022. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles* 7, 3 (2022), 652–674.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[10] Jing Chen, Jia Chen, Kuo Guo, Renkun Hu, Tao Zou, Jun Zhu, Hongke Zhang, and Jingjing Liu. 2024. Fault Tolerance Oriented SFC Optimization in SDN/NFV-Enabled Cloud Environment Based on Deep Reinforcement Learning. *IEEE Transactions on Cloud Computing* 12, 1 (2024), 200–218.

[11] Xiong Wang, Jiancheng Ye, and John C.S. Lui. 2024. Online Learning Aided Decentralized Multi-User Task Offloading for Mobile Edge Computing. *IEEE Transactions on Mobile Computing* 23, 4 (2024), 3328–3342.

[12] Ziya Chen, Qian Ma, Lin Gao, and Xu Chen. 2024. Price Competition in Multi-Server Edge Computing Networks Under SAA and SIQ Models. *IEEE Transactions on Mobile Computing* 23, 1 (2024), 754–768.

[13] Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y Zomaya. 2020. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal* 7, 8 (2020), 7457–7469.

[14] Hai Jin, Lin Jia, and Zhi Zhou. 2020. Boosting edge intelligence with collaborative cross-edge analytics. *IEEE Internet of Things journal* 8, 4 (2020), 2444–2458.

[15] Xiaokang Zhou, Qiuyue Yang, Qiang Liu, Wei Liang, Kevin Wang, Zhi Liu, Jianhua Ma, and Qun Jin. 2024. Spatial–Temporal Federated Transfer Learning with multi-sensor data fusion for cooperative positioning. *Information Fusion* 105 (2024), 102182.

[16] Jinke Ren, Guanding Yu, and Guangyao Ding. 2020. Accelerating DNN training in wireless federated edge learning systems. *IEEE Journal on Selected Areas in Communications* 39, 1 (2020), 219–232.

[17] Xiaokang Zhou, Xuzhe Zheng, Xuesong Cui, Jiashuai Shi, Wei Liang, Zheng Yan, Laurence T. Yang, Shohei Shimizu, and Kevin I-Kai Wang. 2023. Digital Twin Enhanced Federated Reinforcement Learning With Lightweight Knowledge Distillation in Mobile Networks. *IEEE Journal on Selected Areas in Communications* 41, 10 (2023), 3191–3211.

[18] Letian Zhang, Lixing Chen, and Jie Xu. 2021. Autodidactic neurosurgeon: Collaborative deep inference for mobile edge intelligence via online learning. In *Proceedings of the Web Conference 2021*. 3111–3123.

[19] Yueyue Dai, Ke Zhang, Sabita Maharjan, and Yan Zhang. 2020. Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond. *IEEE Transactions on Vehicular Technology* 69, 10 (2020), 12175–12186.

[20] Laha Ale, Ning Zhang, Xiaojie Fang, Xianfu Chen, Shaohua Wu, and Longzhuang Li. 2021. Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning. *IEEE Transactions on Cognitive Communications and Networking* 7, 3 (2021), 881–892.
[21] Chunhe Song, Wenxiang Xu, Tingting Wu, Shimao Yu, Peng Zeng, and Ning Zhang. 2021. QoE-driven edge caching in vehicle networks based on deep reinforcement learning. *IEEE Transactions on Vehicular Technology* 70, 6 (2021), 5286–5295.
[22] Xiaofei Wang, Ruibin Li, Chenyang Wang, Xiuhua Li, Tarik Taleb, and Victor CM Leung. 2020. Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching. *IEEE Journal on Selected Areas in Communications* 39, 1 (2020), 154–169.
[23] Xiaokang Zhou, Wei Liang, I Kevin, Kai Wang, Zheng Yan, Laurence T Yang, Wei Wei, Jianhua Ma, and Qun Jin. 2023. Decentralized P2P federated learning for privacy-preserving and resilient mobile robotic systems. *IEEE Wireless Communications* 30, 2 (2023), 82–89.
[24] Zhenguo Ma, Yang Xu, Hongli Xu, Zeyu Meng, Liusheng Huang, and Yinxing Xue. 2023. Adaptive Batch Size for Federated Learning in Resource-Constrained Edge Computing. *IEEE Transactions on Mobile Computing* 22, 1 (2023), 37–53.
[25] Stefanos Laskaridis, Stylianos I Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D Lane. 2020. SPINN: synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th annual international conference on mobile computing and networking*. 1–15.
[26] Adyson M Maia, Yacine Ghamri-Doudane, Dario Vieira, and Miguel F de Castro. 2019. Optimized placement of scalable iot services in edge computing. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 189–197.
[27] Wenkai Lv, Quan Wang, Pengfei Yang, Yunqing Ding, Bijie Yi, Zhenyi Wang, and Chengmin Lin. 2022. Microservice Deployment in Edge Computing Based on Deep Q Learning. *IEEE Transactions on Parallel and Distributed Systems* 33, 11 (2022), 2968–2978.
[28] Zhaolong Ning, Peiran Dong, Xiaojie Wang, Shupeng Wang, Xiping Hu, Song Guo, Tie Qiu, Bin Hu, and Ricky YK Kwok. 2020. Distributed and dynamic service placement in pervasive edge computing networks. *IEEE Transactions on Parallel and Distributed Systems* 32, 6 (2020), 1277–1292.
[29] Bo Li, Qiang He, Guangming Cui, Xiaoyu Xia, Feifei Chen, Hai Jin, and Yun Yang. 2022. READ: Robustness-Oriented Edge Application Deployment in Edge Computing Environment. *IEEE Transactions on Services Computing* 15, 3 (2022), 1746–1759.
[30] Shuiguang Deng, Yishan Chen, Gong Chen, Shouling Ji, Jianwei Yin, and Albert Y. Zomaya. 2023. Incentive-Driven Proactive Application Deployment and Pricing on Distributed Edges. *IEEE Transactions on Mobile Computing* 22, 2 (2023), 951–967.
[31] Bin Gao, Zhi Zhou, Fangming Liu, Fei Xu, and Bo Li. 2021. An online framework for joint network selection and service placement in mobile edge computing. *IEEE Transactions on Mobile Computing* 21, 11 (2021), 3836–3851.
[32] Zhiqiang Geng, Zhiwei Chen, Qingchao Meng, and Yongming Han. 2021. Novel transformer based on gated convolutional neural network for dynamic soft sensor modeling of industrial processes. *IEEE Transactions on Industrial Informatics* 18, 3 (2021), 1521–1529.
[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).
[34] Xiaokang Zhou, Qiuyue Yang, Xuzhe Zheng, Wei Liang, Kevin I-Kai Wang, Jianhua Ma, Yi Pan, and Qun Jin. 2024. Personalized Federated Learning With Model-Contrastive Learning for Multi-Modal User Modeling in Human-Centric Metaverse. *IEEE Journal on Selected Areas in Communications* 42, 4 (2024), 817–831.
[35] Thaha Mohammed, Carlee Joe-Wong, Rohit Babbar, and Mario Di Francesco. 2020. Distributed inference acceleration with adaptive DNN partitioning and offloading. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 854–863.
[36] Xiangqiang Gao, Rongke Liu, and Aryan Kaushik. 2022. Virtual network function placement in satellite edge computing with a potential game approach. *IEEE Transactions on Network and Service Management* 19, 2 (2022), 1243–1259.
[37] Tao Fang, Dan Wu, Jiaxin Chen, Chao Yue, and Meng Wang. 2021. Joint Distributed Cache and Power Control in Haptic Communications: A Potential Game Approach. *IEEE Internet of Things Journal* 8, 18 (2021), 14418–14430.
[38] Harvey M Salkin and Cornelis A De Kluyver. 1975. The knapsack problem: a survey. *Naval Research Logistics Quarterly* 22, 1 (1975), 127–144.
[39] Shutong Chen, Lei Jiao, Lin Wang, and Fangming Liu. 2019. An Online Market Mechanism for Edge Emergency Demand Response via Cloudlet Control. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 2566–2574.
[40] Shengyi Wang, Liang Du, and Jason R Marden. 2022. Learning in Potential Games for Electric Power Grids: Models, Dynamics, and Outlook. *IEEE Systems Journal* 16, 3 (2022), 5079–5091.
[41] Qiang He, Guangming Cui, Xuyun Zhang, Feifei Chen, Shuiguang Deng, Hai Jin, Yanhui Li, and Yun Yang. 2019. A game-theoretical approach for user allocation in edge computing environment. *IEEE Transactions on Parallel and Distributed Systems* 31, 3 (2019), 515–529.
[42] Lichao Yang, Heli Zhang, Xi Li, Hong Ji, and Victor CM Leung. 2018. A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing. *IEEE/ACM Transactions on Networking* 26, 6 (2018), 2762–2773.
[43] Tao Zhou, Xiangping Liu, Zuowei Xiang, Haitong Zhang, Bo Ai, Liu Liu, and Xiaorong Jing. 2024. Transformer Network Based Channel Prediction for CSI Feedback Enhancement in AI-Native Air Interface. *IEEE Transactions on Wireless Communications* (2024).
[44] Jingyan Jiang, Ziyue Luo, Chenghao Hu, Zhaoliang He, Zhi Wang, Shutao Xia, and Chuan Wu. 2021. Joint model and data adaptation for cloud inference serving. In *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 279–289.

[45] Shiwei Ding, Lan Zhang, Miao Pan, and Xiaoyong Yuan. 2024. PATROL: Privacy-Oriented Pruning for Collaborative Inference Against Model Inversion Attacks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 4716–4725.
[46] Mohamed A Tawhid and Abdelmonem M Ibrahim. 2020. Feature selection based on rough set approach, wrapper approach, and binary whale optimization algorithm. *International journal of machine learning and cybernetics* 11, 3 (2020), 573–602.
[47] Narinder Singh and SB Singh. 2017. Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance. *Journal of Applied Mathematics* 2017 (2017).

Received 27 August 2022; revised 7 May 2024; accepted 25 July 2024