AI Service Deployment and Resource Allocation Optimization Based on Human-Like Networking Architecture

Yuqi Hu, Qin Li, Yuhao Chai, Di Wu, Lu Lu, *Member, IEEE*, Nanxiang Shi, Yinglei Teng, *Senior Member, IEEE*, and Yong Zhang

这篇文章提出了类人网络架构HLN，在考虑QoAIS的情况下，解决AI 服务的agent放置问题和资源分配问题。这篇文章考虑了AI服务链，即AI任务中模块的依赖关系，文章的目标是在AI任务约束的情况下（包括精度、稳定性、接受率等），通过优化agent的部署和分配资源最小化系统时延。

问题：文章中agent的含义并不明确，agent存在的必要性是什么？功能是什么，都没有讲清楚。

评价：工作量饱和，但是有些乱，工作太杂，比如联合考虑QoAIS、服务部署和资源分配，约束太多，精度和资源约束应该考虑，但是某一部分消耗的约束感觉无此必要，可能因为不加会影响实验结果，但也只是广，没有讲深入，是一个架构探索式的文章，类似于数据面。稳定性是通过一个约束来保证的，讨论得很多，不是很有说服力。

可借鉴的点：精度的定义，数据质量的考虑。

***Abstract*—In the forthcoming sixth-generation (6G) era, edge-network-cloud collaboration is needed to support artificial intelligence as a service (AIaaS) with a strong demand for computing power. However, how to guarantee the Quality of AI Service (QoAIS) and utilize the edge-network-cloud collaboration to enhance the performance of artificial intelligence (AI) service is a big challenge. In this article, we propose an AI service management and network resource scheduling architecture based on human-like networking. Considering the Quality of Service (QoS) requirements and AI tasks, we propose a joint AI agent placement with deep neural network (DNN) deployment and dynamic bandwidth resource allocation algorithm (JAAPD-D). JAAPD-D is proposed to solve the short-term and long-term joint resource allocation problem which includes communication, computation, and memory resources in the network. We adjust the agent placement, DNN deployment, and schedule routing path to ensure effective service transmission in the long time interval and dynamically allocate bandwidth resources in the short time interval. We use Lyapunov optimization to ensure the system stability of the whole network, meet the QoS requirements of various services, and minimize the average end-to-end delay of services. Simulation results show that JAAPD-D outperforms existing algorithms in terms of delay, traffic accepted rate, network system throughput, and cost.**

***Index Terms*—Artificial intelligence as a service (AIaaS), deep neural network (DNN) inference, human-like networking (HLN), Lyapunov optimization, Quality of AI Service (QoAIS), resource management.**

I. Introduction

With the gradual maturation of artificial intelligence (AI) technology and the widespread adoption of AI applications, services and applications that use AI technology can be regarded as AI services, that is, AI as a service (AIaaS) [1], [2], [3], [4], [5]. AIaaS will play an increasingly significant role in the technological infrastructure of society, facilitating, promoting, and supporting functionalities across various applications. The futuristic sixth-generation (6G) communication network is expected to enable massive device connectivity and support emerging AI services, which foster the development of the AI of Things (AIoT) [6]. The boom of AIoT will notably exaggerate the data volumes, thereby imposing heightened pressures on network computing [7]. Emerging applications, including augmented reality (AR)/virtual reality (VR), autonomous driving, and the metaverse present elevated requirements for data processing rates and the capacity for training, inference, and decision making [8]. Consequently, it becomes crucial to address resource allocation and ensure Quality of Service (QoS) in practical networks when deploying these emerging AI services. The edge-network-cloud collaborative computing is considered an effective solution that not only addresses the limitations of terminal computing capabilities but also supports the robust demand for AI services requiring significant computational power.

Generally, AI services involve two phases: 1) model training and 2) inference [9]. Different from conventional services, AI services largely depend on the data provided by users. Therefore, the quality and quantity of user data determine the effectiveness of AI services, including interference accuracy and learning speed, etc. With the development of AI services, there arises a need for additional metrics to effectively characterize network performance and Quality of AI Service (QoAIS), particularly concerning inference efficiency and accuracy. Currently, deep neural network (DNN) has become an important part of AI applications [10], [11]. QoAIS is intricately influenced by the diverse array of DNN models. Consequently, the optimization of AI services requires a nuanced consideration of not only conventional metrics, such as bandwidth, throughput, energy consumption, and delay but also distinctive QoS metrics like model accuracy, inference delay, and inference ability. In this multidimensional QoS framework, the delicate equilibrium of multiobjective optimization among AI services is imperative for sustaining optimal performance [12], [13], [14], [15], [16], [17].

传统指标

According to the current AI service workflow, it tends to send the data to the central cloud for model training and

Manuscript received 30 January 2024; revised 11 March 2024; accepted 23 March 2024. Date of publication 8 April 2024; date of current version 9 July 2024. This work was supported in part by the Beijing University of Posts and Telecommunications-China Mobile Research Institute Joint Innovation Center. *(Corresponding author: Yong Zhang.)*

Yuqi Hu, Yuhao Chai, and Di Wu are with the School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: hyq@bupt.edu.cn; chaiyh@bupt.edu.cn; diwu@bupt.edu.cn).

Qin Li, Lu Lu, and Nanxiang Shi are with the Department of Basic Network Technology, China Mobile Research Institute, Beijing 100053, China (e-mail: liqinyjy@chinamobile.com; lulu@chinamobile.com; shinanxiang@chinamobile.com).

Yinglei Teng and Yong Zhang are with the School of Electronic Engineering and Beijing Key Laboratory of Work Safety Intelligent Monitoring, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: lilytengtt@gmail.com; yongzhang@bupt.edu.cn).

Digital Object Identifier 10.1109/JIOT.2024.3384546

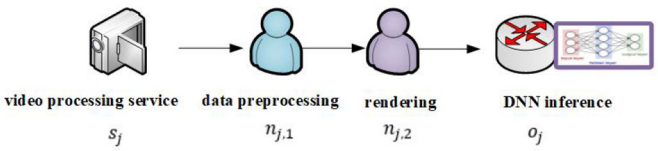


Fig. 1. Example of a task.

inference, and then return decisions to the users. However, as the volume and diversity of AI services gradually increase, leading to a significant growth in data size, the resulting high transmission delay becomes challenging to meet the requirements of ultralow latency scenarios, such as industrial Internet of Things (IoT). Additionally, the escalating communication costs and high bandwidth utilization pose further challenges. This type of transmission may result in data security and privacy issues [18]. The AI task can be processed if the required DNN model exists in the memory of the edge servers or the network node servers. Therefore, we contemplate the transmission of pretrained DNN models from the cloud to edge servers or network node servers to reduce end-to-end delay in service. However, the computation capabilities of edge servers are limited due to the constraints of edge hardware, making it challenging to handle complex AI services. In this article, AI services are considered as tasks, where the processing of a task needs coordination among multiple intelligent agent units. An **Agent** can be a function of any open system interconnection (OSI) layer. For example, it may represent network functions at the application layer, such as firewall (FW), deep packet inspection (DPI), deep flow inspection (DFI), etc., and it can also represent network functions at the network layer, such as routing table forwarding, congestion control, etc. Considering that AI service agents are all application layer network functions, Fig. 1 illustrates an example of a task. Agent in Fig. 1 represents the application layer network functions, namely data preprocessing and rendering. Subsequently, the task is accomplished through DNN inference. Taking the example of object detection tasks, requests can be served by DNN models like YOLO [19] or SSD [20], each with different resource-precision tradeoffs.

渲染、可视化

Divergences in network functions are discernible between AI services and conventional counterparts. Specifically, applications like VR and video processing services necessitate the integration of network functions associated with data preprocessing and rendering. These functions are strategically deployed in proximity to users, aiming to minimize service delays. In contrast, conventional services merely rely on standard network functions for packet processing [21]. Therefore, how to dynamically adjust the placement of different types of agents in AI services and find the optimal routing path in the transmission network requires further investigation.

In response to the growing demands for the QoAISs and the planning challenges of AI service chains (AISCs), the forthcoming 6G networks necessitate a novel framework to perceive, operate, and manage AI services within the Internet of Everything (IoE). Some studies have proposed to consider the network architecture of AI services [9], [22], [23], [24], but these efforts are merely preliminary exploration. The International Telecommunication Union (ITU) recently proposed introducing human-like features into the network and creating a new network architecture called human-like networking (HLN). This architecture leverages AI technology, network awareness technology, and other technologies applied in the network field, enabling the network to behave like a human [25]. Building upon the HLN concept, we are considering establishing a new architecture for AI services. The network nodes in HLN are referred to as autonomous nodes which have the capacity of computing and can support AI functionalities. Federated learning (FL) has been widely investigated and regarded as a distributed computing paradigm in which multiple distributed nodes undergo individual training before sharing parameters [26]. In contrast, the collaborative approach of AI agents in the human-like network proposed employs a chain-like structure to accomplish complex tasks. Nodes participating in this collaboration execute distinct subtasks, with no parameter sharing across agents. Therefore, the operational mode of the AI agents presented is entirely distinct from FL. To the best of our knowledge, this article marks the first contribution within the human-like network architecture regarding AI model placement and resource allocation.

Most existing researches focused on the placement of virtual network functions (VNFs) for traditional services and service function chain (SFC) routing with QoS constraints [13], [21], [27], [28], [29], [30], [31], but they hardly considered the existence of AI services in the network, the QoS requirements of AI services or the placement of DNN models. Moreover, many researchers studied the one-time placement of VNFs [5], [17], without considering queue backlog, long-term constraints on queue stability, and long-term utility. However, the network architecture we proposed is an HLN architecture, fundamentally aiming for algorithms to autonomously make long-term decisions within the network without the need for human intervention. This approach seeks to achieve network automation, and thus, when undertaking deployment optimizations, it is imperative to address the issue of long-term stability in the network system to ensure sustained QoS for service requirements over an extended period. Moreover, the frequent deployment of agents may also increase the deployment cost and lead to service interruption. Therefore, agents embedding and routing strategies in this article are optimized in a larger timescale to maintain network stability, reduce the cost of frequent agent deployment, and ensure service continuity.

In this article, we address the QoS requirements for AI tasks while considering constraints on network computing, memory, and bandwidth resources, as well as cost and agent placement rules. We explore a joint problem of agent placement and traffic routing for multiple AI service requests, aiming to ensure the overall system stability of the network and minimize the end-to-end average delay of service. In this problem, the adoption of modular chain processing for tasks can significantly streamline the transmission and data processing of AI tasks. Nodes with computational capabilities within the network can address the current challenges posed by insufficient computational power at edge nodes, thereby reducing end-to-end delay for tasks. This problem is formulated as a mixed-integer linear programming (MILP) problem.

流水线化

Given the NP-hard nature of the formulated MILP problem, we propose a joint placement scheme for AI agents based on DNN deployment and the dynamic bandwidth resource allocation algorithm. With the assistance of the Lyapunov optimization technique, we decouple the constraints of cost and queue stability in short-term network communication resource scheduling. We carry out both long-term and short-term resource allocation strategies, with the long-term resource allocation strategy aimed at maintaining the long-term stability of the network, preventing queue backlog congestion, and ensuring service stability. In the long time interval, we adjust agent placement, DNN deployment, and routing path to ensure service transmission and increase the traffic accepted rate. Dynamic bandwidth resource allocation is to ensure lower delay in a short time interval. The main contributions of this work can be summarized as follows.

1) We investigate AIaaS, such as image recognition, fault diagnosis, traffic prediction, etc., and propose an architecture for AI service management and network resource scheduling based on HLN. AI tasks are divided into multiple agents and sequentially deployed in the network. To the best of our knowledge, this article stands as the pioneering work delving into AI agent placement and resource allocation under the HLN architecture.
2) We formulate and analyze an agent placement and routing problem for joint AI tasks, leveraging the edge-network-cloud collaboration to minimize the long-term average delay of tasks while meeting the task inference accuracy requirements under constrained cost and resources, including communication, computation, and memory.
3) By using the Lyapunov optimization technique, we introduce the joint AI agent placement with DNN deployment and dynamic bandwidth resource allocation algorithm (JAAPD-D). This algorithm orchestrates resource allocation across multiple timescales. In the long term, we deploy all agents including DNN models and plan service routing paths. In the short term, bandwidth resources are allocated to reduce service end-to-end delay. We compare the performance of the proposed algorithm with that of existing algorithms. Simulation results show that JAAPD-D has better performance in terms of delay, traffic accepted rate, network throughput, and cost.

The remainder of this article is organized as follows. Section II introduces the system model. Section III presents related work, further elaborating on the novelty of our research. In Section IV, the problem is planned and transformed, and the optimization algorithm is proposed in Section V. The performance of the proposed algorithm is described in Section VI. Finally, Section VII concludes this article.

II. Related Work

The vibrant development and accelerating integration of AI and the IoT in the context of 6G scenarios call for the establishment of new network architectures for multiple AI services. Recently, Song et al. [22] introduced the frameworks of centralized and distributed AI-enabled IoT networks. Wu et al. [23] proposed an AI-native network slicing architecture for 6G networks, integrating AI into the SDN controller. Yang et al. [32] introduced the task-oriented native AI (TONA) network architecture to natively support network AI. By introducing task control and QoAIS assurance mechanisms in the 6G control layer, TONA achieves the finest service granularity at the task level, ensuring personalized Quality of Experience (QoE) for each user. In contrast, our proposed architecture for AI service management and network resource scheduling incorporates comprehensive perception technologies, including traffic awareness technologies (DPI, etc.), service awareness technologies (service provider interface, etc.), modeling technologies, transformation technologies, AI technologies, network automatic flexible control technologies, and big data technologies. The proposed architecture is essentially an upgrade of the SDN architecture, incorporating more AI technologies. Similar to SDN, it consists of the application layer, control layer, and infrastructure layer. However, the most notable difference from the SDN structure is that this architecture divides the original infrastructure layer into the operation and data subnetwork and the awareness subnetwork. Additionally, similar networks can learn from each other's knowledge and directly engage in resource orchestration, further enhancing intelligence.

For AI tasks, in addition to traditional communication metrics, such as delay, throughput, and packet loss rate, it is imperative to consider specific QoS requirements, including but not limited to inference accuracy, inference delay, and model size. Recent research has delved into the tradeoffs between accuracy and delay for AI tasks, addressing the edge-cloud collaborative deployment of DNN models. Zhang et al. [10] introduced an edge-cloud architecture, optimizing task allocation and DNN model placement to maximize average inference accuracy using deep reinforcement learning (DRL) algorithms. Yao et al. [13] explored the joint optimization problem of DNN model caching and DNN request routing in edge collaboration scenarios. Their algorithm, based on stochastic rounding, aimed to maximize throughput under constraints, such as budget, accuracy, and latency. Wu et al. [14] investigated collaborative DNN inference between devices and edge servers for multiple AI services. To minimize service latency while meeting time-averaged accuracy requirements, the authors applied DRL to jointly optimize task sampling rate selection, task offloading, and edge computing resource allocation. Fan et al. [15] proposed a deployment problem for DNN models in edge computing scenarios, aiming to minimize total task processing delay and total error inference penalty while ensuring queue stability and task inference accuracy. However, these studies did not consider cost and task admission rate issues in real-world resource consumption. In our proposed solution, we comprehensively consider delay, inference accuracy, admission rate, and the overall network system's cost and stability.

Mobile edge computing (MEC) and FL are widely recognized as effective paradigms for addressing computational challenges [33], [34]. Furthermore, MEC primarily

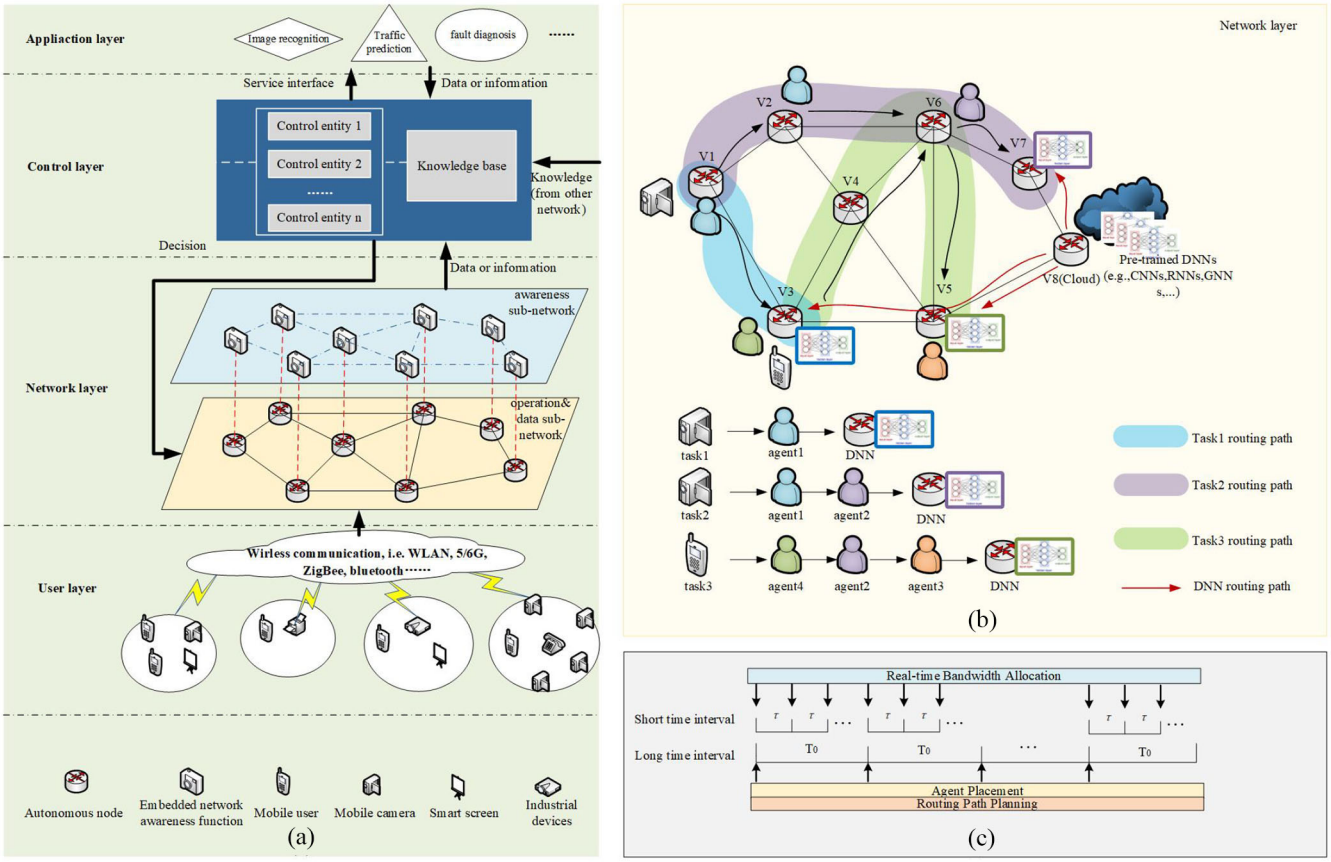


Fig. 2. AI service management and network resource scheduling architecture: (a) is the typical framework, (b) shows the flow of the task, and (c) illustrates the multitime scale model.

focuses on the challenge of resource allocation in wireless networks [35], [36], [37] and FL concentrates more on enhancing DNN accuracy, training, and inference speed [38], [39]. Conversely, our emphasis lies in orchestrating the deployment of AI tasks within the network to minimize task delay while meeting accuracy requirements, and concurrently achieving overall network stability objectives. Existing work primarily focuses on energy-efficient DNN inference for single users or DNN applications in edge scenarios. The deployment problem of DNN models for multiuser and multiapplication chained AI services has been largely overlooked. Premsankar and Ghaddar [40] modeled the energy-efficient placement of AI applications' services (i.e., DNN models) as a multiperiod optimization problem, jointly placing services and scheduling requests to minimize overall energy consumption and reduce latency. However, it still does not consider the chained AI services. Most studies concentrate on the placement of VNFs for traditional services and SFC routing [13], [21], [27], [28], [29], [30], [31], but they almost entirely neglect the presence of AI services in the network, let alone focusing on AI service QoS requirements and DNN model placement problems. Qiu et al. [41] proposed the optimization deployment of AISCs on edge servers, considering the placement of VNFs and blockchain-based VNFs (BVNFs). They aimed to form satisfactory AISCs with secure links while maximizing throughput for incoming requests and minimizing costs in real-time. Our work is different and complementary to [41] in at least three aspects: first, Qiu et al. only considered the edge-cloud collaborative scenario, while chained services require more attention to the network. We incorporate the network into consideration, addressing the edge-network-cloud collaborative scenario. Second, their focus is more on link security and reliability, neglecting the critical delay QoS requirements of AI services, which is our primary focus. Finally, their task processing is limited to an one-time deployment for incoming tasks, while we further consider the long-term stability of the network system, ensuring service meets long-term QoS requirements.

AI服务的依赖关系

III. System Model

A. Network Architecture

Based on HLN architecture, we develop the AI service management and network resource scheduling architecture as shown in Fig. 2(a). The typical framework mainly includes four layers: 1)user layer; 2) network layer; 3) control layer; and 4) application layer [25]. User layer consists of user devices, such as mobile users and smart screens, which are used to collect user information. Network layer consists of network devices and includes two subnetworks: 1) the operation and data subnetwork and 2) the awareness subnetwork. The operation and data subnetwork transmits data and receives

instructions from the control layer, and the awareness subnetwork senses the network contents, and sends this information to the control layer. Control layer includes the knowledge base and multiple control entities. The control entities are responsible for processing the information, generating knowledge, and making decisions. Knowledge base is responsible for storing useful data and receiving knowledge from other similar networks. Application layer contains image recognition, fault diagnosis, etc., which receives information from the control layer and provides it to the control layer.

In this network architecture, all kinds of AI services are considered as tasks, and a task requires multiple agent units to cooperate and complete. These agents are placed on autonomous nodes that do not require manual configuration and can operate on any layer of the networking stack, for example, routers, switches, personal computers, etc. Awareness function is embedded in the autonomous node, which can sense the network content. The awareness subnetwork is composed of awareness function. Awareness function has three aspects: 1) network traffic awareness; 2) network resource awareness; and 3) network demand awareness, which can know the distribution of various network services, the utility of network resources, and the requirements from users, respectively, [25]. Specific information awareness methods can be implemented through open interfaces for network resource awareness and network traffic monitoring [42], [43]. Furthermore, application awareness can be achieved through deep learning mechanisms [44]. In summary, awareness functions can sense network traffic, resources, and user demands, and collect necessary information to allocate network resources and deploy AI tasks, thus achieving the goal of self-awareness.

In this article, the AI task processing is taken into consideration. Each AI task processing involves different functions, such as data preprocessing, training, inference, etc. For example, if there is a video service, video rendering and other functions may also need to be considered. We decompose a complete and complex AI service into different agents, i.e., AI service is a task function chain. Different agents can be processed on different network nodes. Furthermore, we do not specifically pay attention to the way of data preprocessing (such as resizing, normalization, augmentation, etc.) or data training (such as RNN, CNN, etc.) but only focus on the demanding resources of data preprocessing or data training, to conduct the overall arrangement of service in the network.

As shown in Fig. 2(b), in the regional operation and data subnetwork, there are distributed autonomous nodes with computation capabilities besides the centralized node (cloud) with huge computation resources. In the awareness subnetwork, each node has the awareness function for user traffic, network resources, and service demands. Therefore, when multiple AI tasks arrive, the awareness subnetwork will first sense different service requirements. The control entity will arrange the task sequence after receiving the service information and resources and determine the required DNN model of each AI service according to its requirements at the same time. Then, the control entity determines agent placement and DNN model placement, and selects the routing path for each service. AI service traffic that requires high computation capabilities can be selected to autonomous nodes for calculation according to service requirements, and the central node can deploy the pretrained DNN model on the autonomous network nodes with computation capabilities for inference. Generally, we regard DNN inference as the final step of AI service, and the DNN model placing node is the target node of the task. Taking Task 3 as an example from Fig. 2(b), it enters the edge node V3. At this point, the awareness subnetwork assesses the incoming traffic volume and service requirements. Based on the current network resources, it plans accordingly. Following the algorithmic decisions, Agent 1 is placed on node V3, Agent 2 on node V6, and Agent 3 along with the DNN model on node V5. Subsequently, on the data and operation subnetworks, traffic data is uploaded according to the routing path, while the DNN model is downloaded from the cloud. The DNN model performs service inference within autonomous nodes and ultimately returns the results. This completes the entire AI service inference process. The main notations used in this article are summarized in Table I.

B. Multitime Scale Model

To consider the long-term utility of the network system, we assume that the operation of the network system is time-slotted. As shown in Fig. 2(c), let $\mathcal{T} = \{1, \ldots, t, \ldots, T\}$ represent the time-slot system. The length of short time interval is τ and the length of long time interval is T_0, where each long time interval T_0 contains h short time intervals, i.e., $T_0 = h\tau$. Let p represent the long time interval. The tasks arrive randomly at each time slot according to the Poisson distribution.

C. Agent Placement Model

The graph $G^P(V^P, E^P)$ represents the actual network, where V^P and E^P denote the sets of node and link, respectively. In the network, we consider that the awareness subnetwork is $G^a(V^a, E^a)$. The subnetwork is a virtual network, mainly for traffic and resource awareness. We use V^a to indicate autonomous node set, i.e., a node using autonomous functions. Autonomous nodes can not only communicate with neighboring nodes but also have computation resources that can process computing tasks. E^a indicates the virtual link set. The virtual link between each two nodes consists of multiple physical links and passes through multiple nodes. The memory and computation resources of each node are different. The memory size of the node v is denoted by C_v^{mem} and $v \in V^a$. The computation capacity of the node v is denoted by C_v^{cpu} and $v \in V^a$. Let $vu \in E^a$ denote the link connecting node v and u. The total bandwidth of the link between two adjacent nodes v and u is expressed as C_{vu}^{link} and $vu \in E^a$.

In this model, we focus mainly on users of AI services. AI users $j \in \mathcal{K}_{AI} = \{1, 2, \ldots, K_{AI}\}$ have QoS requirements for inference accuracy. Each AI user has different task requirements, and each task needs to be completed by multiple agents in cooperation, and the final destination node of the task is the node where the DNN model exists. Each task is processed by a series of agents in order. Let $N_j = \{n_{j,1}, n_{j,2}, \ldots n_{j,|N_j|}\}$ be the

TABLE I
SUMMARY OF NOTATIONS

Network	
$G^P(V^P, E^P)$	Actual physical network, where V^P and E^P represent node sets and link sets respectively
$G^a(V^a, E^a)$	Network awareness subgraph, where V^a represents the autonomous node and E^a represents the virtual link
C_v^{cpu},C_v^{mem},C_{vu}^{link}	The computation resource of the node v, the memory resource of the node v, and the total bandwidth of the link between two adjacent nodes v and u
$\varphi_v^{cpu}(t)$,$\varphi_v^{mem}(t)$,$\varphi_{uv}^{link}(t)$	The remaining computation resources, memory resources of node v and link bandwidth of link uv respectively
$\beta_{j,n}^{cpu}$	The computation consumption of agent type n
M^{cpu}, M^{mem}	The consumption cost of computation resource, memory consumption cost
$Z_v(t)$	The resource cost of node v at time t
Z^r	The upper limit of single node cost
AI task	
$\mathcal{K}_{AI}$	The set of AI task
$A_j(p)$	The rate of the user's arrival traffic data
N_j	The order set of agents of AI task j
η_j	The requirements of accuracy
R_j	The requirements of bandwidth
$\omega_j(p)$	The quality of data of AI task j
$x_v^{jn}(p)$	The binary variable indicates whether the agent n of task j is embedded in node v at p
$y_{uv}^{jnn'}(p)$	The binary variable indicates whether the user j traffic flow passes through the node u carrying agent n and routes to the node v carrying the next agent n' at p
$l_{uv}^{jnn'}(p)$	Number of hops on the route from node u to node v in task j in the actual physical network at p
$\delta_v^{jn}(p)$	The binary variable indicates whether task j's agent n is migrated to node v at p
DNN model	
$x_v^{jm}(p)$	The binary variable indicates whether the DNN model m of task j is embedded in node v at p
$\delta_v^{jm}(p)$	The binary variable indicates whether the DNN model m of task j is migrated to node v at time p
$\beta_{j,m}^{cpu}(p)$,$\beta_{j,m}^{mem}(p)$	The computation consumption of agent type n and the memory consumption of agent type n
$l_{cv}^{jm}(p)$,$l_{uv}^{jm}(p)$	Number of hops on the route from cloud c to node v in task j and from node u migrating to node v in the actual physical network
ε_j^m	Accuracy factors related to different DNN models
$S_j^m(p)$	The size of DNN m
$r_j^m(t)$	The requirement of DNN m transmission rate

order set of agents, i.e., $n_{j,1} \to n_{j,2} \to \ldots$. For AI user j, task is defined as $W_j(p) = \{A_j(p), s_j(p), o_j(p), N_j, R_j, \eta_j^r, \omega_j(p)\}$, where $A_j(p)$ represents the arrival rate of the user's traffic data, $s_j(p)$ and $o_j(p)$ represent the source and destination nodes, respectively. R_j indicates the user's transmission rate requirements. η_j^r indicates the accuracy requirement or error of user j because different AI services usually have different minimum accuracy thresholds or maximum error thresholds, which can be integrated into an unified expression form through inequality transformation. $\omega_j(p)$ indicates the quality of data. The DNN model $m \in M$ can be placed on the destination node $o_j(p)$. $x_v^{jn}(p) = 1$ denotes that agent n of task j is placed on node v in the long time interval p, and $x_v^{jm}(p) = 1$ denotes that DNN model m is placed on node v in the long time interval p. $y_{uv}^{jnn'}(p) = 1$ represents that task j flow passes through the node u carrying agent n and routes to the node v carrying the next agent n'

$$C_1 : y_{uv}^{jnn'}(p) = x_u^{jn}(p) * x_v^{jn'}(p) \quad \forall n, n' \in N_j \ \forall u, v \in V^a \ \forall uv \in E^a. \tag{1}$$

Let $\beta_{j,n}^{\text{cpu}}$ denote the computation consumption of agent n. $\beta_{j,m}^{\text{mem}}(p)$ is the memory consumption of the pretrained DNN model m deployed on the node required by user j, and $\beta_{j,m}^{\text{cpu}}(p)$ is the CPU resources allocated to user j when using the DNN model m to perform inference in the long time interval p. $r_j(t)$ is the link bandwidth allocated to user j. $\varphi_v^{\text{cpu}}(t)$, $\varphi_v^{\text{mem}}(t)$, and $\varphi_{uv}^{\text{link}}(t)$ denote the remaining computation resources, memory resources of node v, and link bandwidth of link uv, respectively

$$C_2 : \sum_{j\in K_{AI}} \sum_{n\in N_j} \beta_{j,n}^{\text{cpu}} x_v^{jn}(p) + \sum_{j\in K_{AI}} \sum_{m\in M} \beta_{j,m}^{\text{cpu}}(p) x_v^{jm}(p) \le \varphi_v^{\text{cpu}}(t) C_v^{\text{cpu}} \ \forall v \in V^a \tag{2}$$

$$C_3 : \sum_{j\in K_{AI}} \sum_{m\in M} \beta_{j,m}^{\text{mem}}(p) x_v^{jm}(p) \le \varphi_v^{\text{mem}}(t) C_v^{\text{mem}} \ \forall v \in V^a \tag{3}$$

$$C_4 : \sum_{j\in K_{AI}} \sum_{n\in N_j} r_j(t) y_{uv}^{jnn'}(p) \le \varphi_{uv}^{\text{link}}(t) C_{uv}^{\text{link}} \ \forall uv \in E^a \tag{4}$$

$$C_5 : r_j(t) \ge R_j \ \forall j \in K_{AI} \tag{5}$$

where C_2, C_3, and C_4 represent computation, memory, and communication bandwidth resource constraints, respectively. C_5 is the service bandwidth requirement.

D. Cost Model

The cost is expressed by the resource utilization rate of the node, which includes the computation resource utilization rate and memory resource utilization rate of the node. The higher the consumption of resources and the smaller the remaining resources of the node, the higher the cost. Considering the development of optical communication technology, the link bandwidth is sufficient to meet the communication requirements. If there are enough bandwidth resources, bandwidth will be allocated to the services that need to be transmitted as much as possible to minimize the transmission delay. The resource cost of node v can be expressed as follows:

$$Z_v(t) = M^{\text{cpu}} \frac{\sum_{j\in K_{AI}} \sum_{n\in N_j} \beta_{j,n}^{\text{cpu}} x_v^{jn}(p) + \sum_{j\in K_{AI}} \sum_{m\in M} \beta_{j,m}^{\text{cpu}}(p) x_v^{jm}(p)}{\varphi_v^{\text{cpu}}(t) C_v^{\text{cpu}}} + M^{\text{mem}} \frac{\sum_{j\in K_{AI}} \sum_{m\in M} \beta_{j,m}^{\text{mem}}(p) x_v^{jm}(p)}{\varphi_v^{\text{mem}}(t) C_v^{\text{mem}}} \ \forall v \in V^a \tag{6}$$

$$C_6 : \bar{Z}_v = \lim_{T\to\infty} \frac{1}{T} \sum_{t\in T} Z_v(t) \le Z^r \tag{7}$$

where M^{cpu} is the consumption cost of computation resource, M^{mem} represents memory consumption cost, and Z^r is the upper boundary of single node cost. C_6 indicates that the average cost of the node v is under the cost constraints in the long time interval.

E. Inference Accuracy Model

The quantity and quality of AI user data will affect its accuracy, such as image recognition and other services. The reduction of data or the poor quality of collected data will lead to the reduction of its recognition accuracy. Different types of DNN models will also affect the accuracy of AI services in the system. Generally, DNN models have two types: 1) compressed (light-weight) DNN and 2) uncompressed (full-size)

DNN [14], [15]. Compressed DNNs are typically obtained through model compression techniques, such as model weight pruning and quantization [45]. The compressed DNN has fewer model weights, a smaller size, and requires fewer resources. For the same data set, the inference accuracy of uncompressed DNN is higher than that of compressed DNN because it has more neurons and layers. However, a more accurate DNN model will occupy more computation and memory resources, so making a tradeoff between accuracy and resource consumption is necessary.

We are focusing on pretrained DNN models, where the accuracy factor can be determined, and the inference accuracy also depends on the quality of the traffic data being inferred. Since the type of DNN model and the data quality of task are mutually independent and uncoupled, we represent the accuracy of AI services as the product of accuracy value respect to the type of DNN model and accuracy value of data quality [14], [15], [41]. Let $g(x)$ be the accuracy function related to the quality of data 精度等于模型精度乘以数据质量

$$\eta_j(p) = \varepsilon_j^m \cdot g\big(\omega_j(p)\big) \tag{8}$$

where ε_j^m is the accuracy factors related to different DNN model m. Note that this function readily extends to scenarios employing alternative inference methods, as the accuracy values concerning data quality and DNN types can be empirically obtained through practical experimentation.

AI users' interference accuracy needs to meet the accuracy requirements

$$C_7 : \eta_j(p) \geq \eta_j^r \ \forall j \in K_{AI}. \tag{9}$$

F. Delay Model

The end-to-end delay for each AI service comprises the deployment delay of agents, the data transmission delay, and the inferring delay at the destination node. It is assumed that the packet size will be negligible after task processing, thereby allowing the omission of backhaul delay. The following section provides a detailed description of these delays.

1) Agent Deployment Delay: For the agents in the task, replacing the agent will generate a new instantiation delay. In practice, the instantiation delay is very short and is influenced by different devices and platforms. However, the instantiation delay is often significantly smaller than the transmission delay. Without losing generality, we assume that the instantiation delay of each agent can be set as a constant D_{ins} [21]. The agent deployment delay can be expressed as

$$D^{PL,jn}(t) = \begin{cases} \sum_{v\in V^a}\sum_{n\in N_j} D_{\text{ins}} x_v^{jn}(p), & t = 1 \\ \sum_{v\in V^a}\sum_{n\in N_j} D_{\text{ins}} x_v^{jn}(p)\delta_v^{jn}(p), & t = p,\ t \neq 1 \\ 0, & \text{else} \end{cases} \tag{10}$$

$$\delta_v^{jn}(p) = \begin{cases} 1, & x_v^{jn}(p) - x_v^{jn}(p-1) = 1 \\ 0, & \text{else} \end{cases} \tag{11}$$

where $\delta_v^{jn}(p)$ is the change of the node v carrying agent n in the long time interval p.

For the DNN model, data transmission is required for model migration or redeployment, which is calculated in transmission delay.

2) Transmission Delay: The service transmission delay of AI services is related to the size of service data and allocated bandwidth, i.e., the link transmission rate. However, it is worth noting that the service traffic is not constant after passing through different agents. For example, when the service data is preprocessed, the data will generally become smaller. Therefore, the traffic change factor through agent n can be expressed as α_n, and the cumulative change rate of the jth AI service can be expressed as $\rho_j = \prod_{n\in N_j}\alpha_n$. The transmission delay of adjacent agents is

$$D_n^{TS,j}(t) = \sum_{uv\in E^a}\sum_{n,n'\in N_j} y_{uv}^{jnn'}(p) l_{uv}^{jnn'}(p) \frac{\prod_{i=n_{j,1}}^{n}\alpha_i A_j(p)\tau}{r_j(t)} \tag{12}$$

where $l_{uv}^{jnn'}(p)$ is the number of route hops of the selected path from node u to v in the physical network. $\prod_{i=n_{j,1}}^{n}\alpha_i A_j(p)\tau$ represents the cumulative change rate of service data from agent $n_{j,1}$ to n.

For AI users, there is also a DNN model transmission delay from the cloud to the autonomous node in the network

$$D_m^{TS,j}(t) = \min\Bigg\{ \sum_{c,v\in V^a}\sum_{cv\in E^a}\sum_{m\in M} x_v^{jm}(p) l_{cv}^{jm}(p)\frac{S_j^m(p)}{r_j^m(t)} \sum_{u,v\in V^a}\sum_{uv\in E^a}\sum_{m\in M}\delta_v^{jm}(p)x_v^{jm}(p)l_{uv}^{jm}(p)\frac{S_j^m(p)}{r_j^m(t)} \Bigg\} \quad \forall t \in [p, p+T_0-1] \tag{13}$$

where $l_{cv}^{jm}(p)$ is the number of route hops of the selected path from cloud node c to v in the physical network and $l_{uv}^{jm}(p)$ is the number of route hops from node u to v. $S_j^m(p)$ indicates the size of DNN m and $r_j^m(t)$ denotes the rate at which DNN models are transmitted. The first item in (13) represents the delay of the DNN model transmitting from the cloud to the autonomous node. Different DNN models have been pretrained according to the large-scale model and data. After selecting the corresponding model according to the services requirements, it is not necessary to retrain the model by uploading data, but directly use the existing model for inference, so the DNN model in the network does not need to be distributed from the cloud every time. At this point, we need to consider the problem of DNN model migration. The latter item in the above formula represents the delay of DNN model migration. We choose the scheme with a smaller delay to transmit and deploy the model. It is worth noting that if the DNN deployment does not change, a new DNN transmission delay will not be generated at this time, but memory resources will be consumed.

Given the full awareness function of HLN, service requirements are sensed upon the arrival of traffic. Subsequently, model selection and routing path planning are executed. Simultaneously, when the task is uploaded, the requisite DNN

model is distributed. Consequently, for user j, its overall transmission delay is

$$D^{TS,j}(t) = \max\left\{D_n^{TS,j}(t), D_m^{TS,j}(t)\right\}. \tag{14}$$

推理时延等于计算时延

3) Inferring Delay: Without the loss of generality, we assume that the considered task inferring delay effectively refers to the processing delay incurred on the computational nodes [12], [14], [15]. The inferring delay is related to the computation capacity of the node. Different nodes have different computation and memory capabilities. The delay of inferring can be expressed as

$$D^{PR,j}(t) = \sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p)\frac{\rho_j A_j(p)\tau L}{\beta_{j,m}^{\text{cpu}}(p)} \quad \forall t \in \left[p, p+T_0-1\right] \tag{15}$$

where L represents the processing density (in CPU cycles /bit) [46].

End-to-end delay of user j is expressed as

$$D_j^E(t) = D^{PL,jn}(t) + D^{TS,j}(t) + D^{PR,j}(t). \tag{16}$$

G. Queuing Model

The system queue backlog model corresponding to AI user j in slot t can be expressed as

$$Q^j(t+1) = \max\left[Q^j(t) - r_j(t)\tau, 0\right] + \rho_j A_j(p)\tau \tag{17}$$

$$Q^{AI}(t) = \left[Q^1(t), Q^2(t), \ldots, Q^{K_{AI}}(t)\right] \tag{18}$$

where $Q^{AI}(0) = 0$ and $Q^{AI}(t)$ represents a vector composed of $Q^j(t)$.

All computation resources of nodes with autonomous functionalities are integrated. The control layer uniformly manages resources and allocates them to AI users. Therefore, AI services maintain task buffers for each user to store computing tasks that have arrived but have not yet been executed. The memory resource is assumed to have sufficient capacity. Then, the AI user processing queue can be expressed as

$$F^j(t+1) = \sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p)\left\{\max\left[F^j(t) - \beta_{j,m}^{\text{cpu}}(p), 0\right] + \min L\left[Q^j(t), r_j(t)\right]\right\} \tag{19}$$

$$F^{AI}(t) = \left[F^1(t), F^2(t), \ldots, F^{K_{AI}}(t)\right] \tag{20}$$

where $F^{AI}(0) = 0$ and $F^{AI}(t)$ represents a vector composed of $F^j(t)$.

To meet the long-term stability requirements of the network, we have the following constraints:

$$C_8: \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathrm{E}\left\{Q^{AI}(t)\right\} \le \infty \tag{21}$$

$$C_9: \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathrm{E}\left\{F^{AI}(t)\right\} \le \infty. \tag{22}$$

For the long-term stability of the network system, the longer the queue is, the more congested the task is, and the more time slots fail to meet the requirements. Therefore, we formalize the challenge of long-term queue stability using the Lyapunov model.

IV. Problem Formulation and Transformation

The average end-to-end delay of AI services in the network system is the long-term optimization goal of the system. Based on the service delay of a single time slot, the end-to-end delay of the system can be expressed as

$$D^E(t) = \frac{1}{|K_{AI}|}\sum_{j\in K_{AI}} D_j^E(t) \tag{23}$$

$$C_{10}: D_T \ge D^E(t). \tag{24}$$

A. Problem Formulation

In this article, aiming at minimizing the average end-to-end delay of AI services, while satisfying the inference accuracy of AI services and the constraints of network resource and cost, we formula the multitime scale joint agent placement, routing path planning and network resource scheduling optimization problem as follows:

$$\begin{aligned} P0: \min_{\{\mathbf{x}(p),\mathbf{y}(p),\boldsymbol{\beta}^{\mathbf{mem}}(p),\boldsymbol{\beta}^{\mathbf{cpu}}(p),\mathbf{r}(t)\}} & \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathrm{E}\{D_T\} \\ \text{s.t.} \quad & C_1 - C_{10} \\ & C_{11}: \sum_{v\in V^a} x_v^{jn}(p) = 1 \ \forall j \in K_{AI} \ \forall n \in N_j \\ & C_{12}: \sum_{v\in V^a} x_v^{jm}(p) = 1 \ \forall j \in K_{AI} \ \forall m \in M \end{aligned} \tag{25}$$

where $\mathbf{x}(p)$ represents the sets of $x_v^{jm}(p)$ and $x_v^{jm}(p)$. $\mathbf{y}(p)$, $\boldsymbol{\beta}^{\mathbf{mem}}(p)$, $\boldsymbol{\beta}^{\mathbf{cpu}}(p)$, and $\mathbf{r}(t)$ represent the sets of $y_{uv}^{jnn'}(p)$,$\beta_{j,m}^{\text{mem}}(p)$, $\beta_{j,m}^{\text{cpu}}(p)$, and $r_j(t)$, respectively. C_1 represents the constraint of link selection. C_2, C_3, and C_4 represent computation, memory, and communication bandwidth resource constraints, respectively. C_5 is the service bandwidth requirement. C_7 is the demand for service accuracy. C_8 and C_9 are the long-term stability demands of the network. C_{11} indicates that the agent n of task j must be placed on only one autonomous node. C_{12} means that the DNN model m selected by task j must also be placed on only one autonomous node. Problem $P0$ is an MILP problem because: 1) the variables are continuous-discrete-mixed and are deeply coupled, where $\mathbf{x}(p)$ and $\mathbf{y}(p)$ are discrete variables, $\boldsymbol{\beta}^{\mathbf{mem}}(p)$ and $\boldsymbol{\beta}^{\mathbf{cpu}}(p)$ are continuous variables, and there is deep coupling between $\mathbf{x}(p)$ and $\mathbf{y}(p)$ and 2) the objective function is nonconvex. Existing research has already proven that integer programming is NP-complete [47], and MILP is NP-hard [48]. Therefore, the problem we propose is an NP-hard problem. To address this problem, we first utilize Lyapunov optimization to transform the problem and then propose a heuristic algorithm to obtain suboptimal solutions for the problem.

B. Lyapunov Optimization Model Transformation

C_6 is the constraint on the cost of agent placement and DNN placement in the long time interval. To solve this problem, we transform C_6 into a queue stability problem. Virtual queues can be represented as

$$E_v(t+1) = \max\left[E_v(t) - Z^r, 0\right] + Z_v(t). \tag{26}$$

We consider using Lyapunov optimization to solve the problems above. Let $\mathbf{\Theta}(t) = [Q^{AI}(t), F^{AI}(t), E(t)]$, the Lyapunov function is defined as

$$L(\mathbf{\Theta}(t)) = \frac{1}{2}\left\{\sum_{j\in K_{AI}}\left[Q^j(t)\right]^2 + \sum_{j\in K_{AI}}\left[F^j(t)\right]^2 + \sum_{v\in V^a}\left[E_v(t)\right]^2\right\}. \quad (27)$$

The Lyapunov shift under long time period p is defined as

$$\Delta_{T_0}(L(\mathbf{\Theta}(p)\ ) = \mathbb{E}\{L(\mathbf{\Theta}(p+T_0)) - L(\mathbf{\Theta}(p))|\mathbf{\Theta}(p)\} \quad (28)$$

which means that given the current queue state, the expectation of the change of the Lyapunov function from the long time slot p to slot $p+T_0$.

The drift minimization method uses the current service state and the current queue backlog to stabilize the system without prior knowledge of the service. By adding the system control parameter V, we obtain the Lyapunov drift-plus-penalty function as

$$\Delta_{T_0}(L(\mathbf{\Theta}(p))) + V\mathbb{E}\left\{D^E(t)|\mathbf{\Theta}(p)\right\} \quad (29)$$

where V is a non-negative weight parameter used to balance the queue backlog and optimization objectives.

Theorem 1: Drift-plus-penalty is subject to the following constraints:

$$\begin{aligned}
&\Delta_{T_0}(L(\mathbf{\Theta}(p))) + V\mathbb{E}\left\{D^E(p)|\mathbf{\Theta}(p)\right\} \le \\
&C - \sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}} Q^j(t)\mathbb{E}\left[r_j(t)\tau - \rho_j A_j(p)\tau|\mathbf{\Theta}(p)\right] \\
&\quad - \sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}}\sum_{v\in V^a}\sum_{m\in M} F^j(t)\mathbb{E}\left\{x_v^{jm}(p)\left[\beta_{j,m}^{\text{cpu}}(p) - Lr_j(t)\right]|\mathbf{\Theta}(p)\right\} \\
&\quad - \sum_{t=p}^{p+T_0-1}\sum_{v\in V^a} E_v(t)\mathbb{E}\left[Z^r - Z_v(t)|\mathbf{\Theta}(p)\right] \\
&\quad + V\mathbb{E}\left\{\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}} D_j^E(t)|\mathbf{\Theta}(p)\right\} \\
&= C + \sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}} Q^j(t)\rho_j A_j(p)\tau \\
&\quad - \sum_{t=p}^{p+T_0-1}\sum_{v\in V^a} E_v(t)Z^r + \Lambda. \quad (30)
\end{aligned}$$

where C satisfies the constraint as follow:

$$\begin{aligned}
C \ge \frac{1}{2}\Bigg\{&\left[\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}} r_j(t)\tau\right]^2 + \left[\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}} \rho_j A_j(p)\tau\right]^2 \\
&+ \left[\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}}\sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p)\beta_{j,m}^{\text{cpu}}(p)\right]^2 \\
&+ \left[\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}}\sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p) r_j(t) L\right]^2 \\
&+ \left[\sum_{t=p}^{p+T_0-1}\sum_{v\in V^a} Z^r\right]^2 + \left[\sum_{t=p}^{p+T_0-1}\sum_{v\in V^a} Z_v(t)\right]^2\Bigg\}. \quad (32)
\end{aligned}$$

Proof: See the Appendix. ∎

The original problem $P0$ can be transformed into the problem of minimizing drift function

$$\begin{aligned}
P1:\ &\min_{\{\mathbf{x}(p),\mathbf{y}(p),\boldsymbol{\beta}^{\text{mem}}(p),\boldsymbol{\beta}^{\text{cpu}}(p),\mathbf{r}(t)\}} \Lambda \\
&\text{s.t.}\quad C_1 - C_{12}. \quad (33)
\end{aligned}$$

V. Algorithms Design

In this section, to overcome the NP-hard challenges of mixed-integer programming problems, we decouple the optimization problem $P1$ into a long-term agent and DNN model placement and routing path planning problem and real-time bandwidth resource allocation problem, and propose the joint AI service agent placement and DNN deployment algorithm (JAAPD). During agent placement for a long period of time, the system allocates minimum communication, computation, and memory resources required by users, i.e., the user bandwidth requirements are considered to meet the minimum transmission rate requirements and the minimum computation resource requirements to meet the service accuracy. We design algorithms for each problem to obtain the optimal solution, respectively. The original mixed integer programming can be converted into a binary integer programming problem (BIP).

$$\begin{aligned}
\Lambda = &-\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}}\mathbb{E}\left[Q^j(t)r_j(t)\tau|\mathbf{\Theta}(p)\right] - \sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}}\sum_{v\in V^a}\sum_{m\in M} F^j(t)\mathbb{E}\{x_v^{jm}(p)\left[\beta_{j,m}^{\text{cpu}}(p) - Lr_j(t)\right]|\mathbf{\Theta}(p)\} \\
&+ \sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}}\sum_{v\in V^a} E_v(t)\mathbb{E}\left[M^{\text{cpu}}\frac{\sum_{n\in N_j}\beta_{j,n}^{\text{cpu}}x_v^{jn}(p) + \sum_{m\in M}\beta_{j,m}^{\text{cpu}}(p)x_v^{jm}(p)}{\varphi_v^{\text{cpu}}(t)C_v^{\text{cpu}}} + M^{\text{mem}}\frac{\sum_{m\in M}\beta_{j,m}^{\text{mem}}(p)x_v^{jm}(p)}{\varphi_v^{\text{mem}}(t)C_v^{\text{mem}}}|\mathbf{\Theta}(p)\right] \\
&+ \frac{V}{|K_{AI}|}\mathbb{E}\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}}\Bigg\{\sum_{v\in V^a}\sum_{n\in N_j} D_{\text{ins}}x_v^{jn}(p)\delta_v^{jn}(p) + \max\left\{\sum_{uv\in E^a}\sum_{n,n'\in N_j} y_{uv}^{jnn'}(p)l_{uv}^{jnn'}(p)\frac{\prod_{i=n_{j,1}}^{n}\alpha_i A_j(p)\tau}{r_j(t)}\right. \\
&\left.\min\left\{\sum_{c,v\in V^a}\sum_{cv\in E^a}\sum_{m\in M} x_v^{jm}(p)l_{cv}^{jm}(p)\frac{S_j^m(p)}{r_j^m(t)}, \sum_{u,v\in V^a}\sum_{uv\in E^a}\sum_{m\in M}\delta_v^{jm}(p)x_v^{jm}(p)l_{uv}^{jm}(p)\frac{S_j^m(p)}{r_j^m(t)}\right\}\right\} + \sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p)\frac{A_j(p)\tau}{\beta_{j,m}^{\text{cpu}}(p)M}\Bigg\} \quad (31)
\end{aligned}$$

A. Agent Placement and Routing Path Planning Problem

The subproblems of agent and DNN placement are

$$SP1: \min_{\{\mathbf{x}(p),\mathbf{y}(p),\boldsymbol{\beta}^{\mathbf{mem}}(p),\boldsymbol{\beta}^{\mathbf{cpu}}(p)\}} \Lambda \quad \text{s.t.} \quad C_1 - C_{12}. \tag{34}$$

According to the task process proposed in Section II, task planning has four phases.

First, the process begins with the simultaneous arrival of multiple tasks. The proposed AI network architecture possesses the capability to independently sense the requirements of each task. Subsequently, it determines the scheduling order of multiple tasks at the control layer and selects the appropriate DNN model. It is worth noting that different DNN models require different computation and memory resources. Once the DNN model is determined, the corresponding resource consumption is established, denoted as $\boldsymbol{\beta}^{\mathbf{mem}}(p)$ and $\boldsymbol{\beta}^{\mathbf{cpu}}(p)$. After that, agents in different tasks are deployed according to the actual network resources over a long period, and the corresponding DNN model location is determined. Then, upload the service and distribute the DNN model according to the deployed agent and DNN model simultaneously. Finally, dynamic bandwidth allocation is performed in a short period to minimize the delay.

To address the challenge, we initially employ a heuristic algorithm to determine the placement and prioritization order of each AI service. Specifically, we prioritize all services based on the accuracy requirements of AI services. Given that different DNN models demand varying computation and memory resources, higher accuracy requirements typically entail larger DNN models. Random planning sequences could potentially lead to subsequent service nodes lacking the capability to support a larger DNN model, resulting in the inability to meet accuracy requirements and subsequent service denials. To enhance the network's capacity to accommodate more services, we adopt a prioritization approach based on the accuracy requirements of AI services and select the smallest DNN model that can meet the accuracy requirements of AI services.

After the DNN model type is determined, the variables $\mathbf{x}(p)$ and $\mathbf{y}(p)$ in $SP1$ solely represent the nodes and edges that carry agents and process the tasks in the graph, respectively. The optimization problem can be reformulated into a minimum weight path problem within the directed weighted graph. For each task, a new directed weighted graph is constructed based on the actual physical network status following the prior planning order, aiming to derive the optimal placement and routing strategy for agents and DNN models. Fig. 3 illustrates this process using an AI task as an example. The task needs to go through several different agents to perform task inference in the DNN model. Each agent can choose to be placed on any autonomous node. The weights of nodes and edges are determined by the variables $\mathbf{x}(p)$ and $\mathbf{y}(p)$ corresponding to the optimization function. The virtual link between each two nodes will select the path with the shortest route hops and required resources in the actual network as the actual route path. Typically, the nodes in the graph lack weights, and most existing algorithms leverage link weights to determine the shortest path. Therefore, we decompose each node into two virtual nodes connected by a virtual link in the middle, with the link weight representing the weight of the node. For example, the node v_n can be decomposed into virtual nodes v_n' and v_n'' and the weight of the virtual link between v_n' and v_n'' can be expressed as the weight of node v_n. Each user performs agent placement, DNN model deployment, and routing path planning through the shortest path algorithm according to the weighted directed graph. We can derive the weight of each user about the node and path. As shown in Fig. 3, the agent placement for AI tasks is opposite to the direction of the DNN model distributed from the cloud. Therefore, the total weight can be taken part into four parts: 1) W_n^a; 2) W_l^a; 3) W_m^S; and 4) W_l^S. The agent placement node is denoted by W_n^a, the weight of the task link is W_l^a, and the weight of the DNN model placement node is W_m^S. The link weight W_l^S is obtained according to the shortest path. Finally, the task will choose a routing path to minimize the objective. $SP1$ can be transformed into $SP1.1$

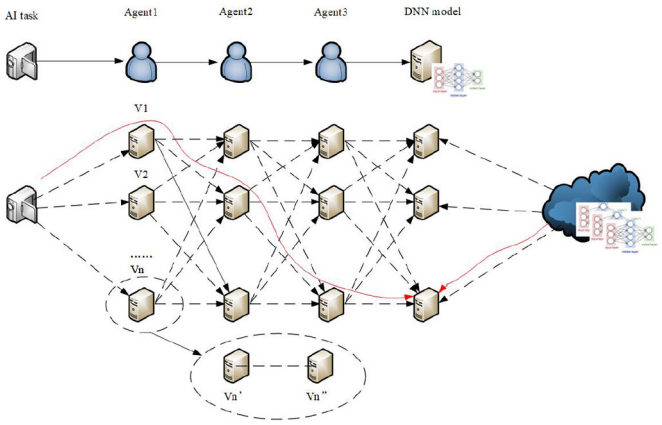


Fig. 3. Establishment of directed weighted graph.

$$SP1.1: \min_{\mathbf{x}(p),\mathbf{y}(p)} \sum_{j\in K_{AI}} \sum_{n\in N_j} W_n^a + W_m^S + \max\left\{W_l^a, W_l^S\right\} \quad \text{s.t.} \quad C_1 - C_{12}. \tag{35}$$

The specific expression of weight value is as follows:

$$W_n^a = \sum_{t=p}^{p+T_0-1} E_v(t) \frac{\beta_{j,n}^{\mathrm{cpu}}}{\varphi_v^{\mathrm{cpu}}(t) C_v^{\mathrm{cpu}}} + \frac{V}{|K_{AI}|} D_{\mathrm{ins}} \delta_v^{jn}(p) \tag{36}$$

$$W_l^a = \frac{V}{|K_{AI}|} l_{uv}^{nn'}(p) \frac{\prod_{i=n_{j,1}}^{n} \alpha_i A_j(p) \tau}{R_j} \tag{37}$$

$$\begin{aligned} W_m^S = & \sum_{t=p}^{p+T_0-1} F^j(t) \left[\beta_{j,m}^{\mathrm{cpu}}(p) - R_j L\right] \\ & + \sum_{t=p}^{p+T_0-1} E_v(t) \left[M^{\mathrm{cpu}} \frac{\beta_{j,m}^{\mathrm{cpu}}(p)}{\varphi_v^{\mathrm{cpu}}(p) C_v^{\mathrm{cpu}}} + M^{\mathrm{mem}} \frac{\beta_{j,m}^{\mathrm{mem}}(p)}{\varphi_v^{\mathrm{mem}}(t) C_v^{\mathrm{mem}}} \right] \\ & + \frac{V \rho_j A_j(p) T_0 L}{|K_{AI}| \beta_{j,m}^{\mathrm{cpu}}(p)} \end{aligned} \tag{38}$$

Algorithm 1 Joint AI Service Agent Placement With DNN Deployment and Dynamic Bandwidth Resource Allocation Algorithm (JAAPD-D)

Require: $G^P, G^a, K_{AI}, W_j(p), T, C_v^{\text{mem}}, C_v^{\text{cpu}}, C_{uv}^{\text{link}}, \varphi_v^{\text{cpu}}(t), \varphi_v^{\text{mem}}(t), \varphi_{uv}^{\text{link}}(t), \beta_{j,n}^{\text{cpu}}, R_j$
Ensure: $\mathbf{x}(p)$, $\mathbf{y}(p)$, $\mathbf{r}(t)$
1: Initialize $p = 1, t = 1$
2: **for** t in T **do**
3: The minimum DNN model set $S(t) = \{S_1(t), \ldots, S_j(t), \ldots, S_{K_{AI}}(t)\}$ can be obtained by the accuracy requirements and arrival data quality of all AI users task.
4: All tasks are deployed and planned from high to low according to the accuracy requirements, $I(t)$ is the new sequence.
5: **if** $mod(t, T_0) == 1$ **then**
6: **for** j in $I(t)$ **do**
7: Delete the nodes and links that do not have sufficient resources for serving user j to construct the network $G_j(p) = \langle V_j, E_j \rangle$ available to user j according to the known network information.
8: Construct a directed weighted graph $G_j{}'(p)$ and find the minimum weight path $P_j(p)$ through Algorithm 2.
9: **if** $P_j(p) = \emptyset$ **then**
10: Reject the user j
11: **else** Map the shortest path to the physical network and update the resources in the network.
12: **end if**
13: **end for**
14: **end if**
15: Allocate bandwidth dynamically according to the sequence $I(t)$ through Algorithm 3
16: Update queue $Q^j(t), F^j(t), E_v(t)$
17: $t = t + 1$
18: **end for**

$$W_l^S = \frac{V}{|K_{AI}|} \sum_{t=p}^{p+T_0-1} \min\left\{ l_{cv}^{jm}(p) \frac{S_j^m(t)}{r_j^m(t)}, \delta_v^{jm}(p) l_{uv}^{jm}(p) \frac{S_j^m(t)}{r_j^m(t)} \right\}. \tag{39}$$

In Algorithm 1, we first prioritize the tasks and select the DNN model with the minimum size for each AI task (lines 3 and 4). Then, we construct the G_j by deleting all the nodes and links that can not meet the needs of task j (line 5). Next, we construct G_j' and find the minimum weight path P_j(line 6). In Algorithm 2, we describe how to construct G_j' in detail. First, we calculate the weight of each node when searching the path in Algorithm 2. We can get the weight of each agent in the task to select different servers, the routing path between adjacent agents, and calculate the weight of the DNN model deployed to feasible destination nodes (lines 2 and 3). Then, according to the shortest Dijkstra algorithm [49], the alternative path set P' (line 4) from the source node to all feasible destination nodes is obtained. At this time, the path only considers the weight of the forward agent and DNN placement not the optimal placement and routing strategy. We need to further filter according to the task optimization goal to find the optimal placement and routing strategy (lines 5–8) and ensure that it meets the resource constraints (lines 9–12).

Algorithm 2 Construction of Directed Weighted Graph and Routing Path Selection

Require: $G_j(p), S_j(p)$
Ensure: $G_j'(p), P_j(p)$
1: Calculate the weight value of each user about the node and path according to the optimization objective converted from Lyapunov optimization.
2: Calculate the weights of the virtual links and other links formed by nodes, and obtain the node weight W_n^a and link weight W_l^a of each selected routing path respectively.
3: Calculate the node weight W_m^S and link weight W_l^S of the shortest path of the feasible DNN model deployed to each feasible destination node.
4: Calculate the shortest path set P' from the source node to all feasible destination nodes according to Dijkstra algorithm
5: **for** P_j' in P' **do**
6: The actual weight value of each path in P' can be obtained from $W_n^a + W_m^S + \max\{W_l^a, W_l^S\}$
7: Find the routing path $P_j'(p)$ with the lowest weight in P' and its corresponding total weight $W_j(p)$.
8: **end for**
9: **if** $P_j(p)$ can meet agent and DNN placement constraints and resource constraints **then**
10: $P_j(p) = P_j'(p)$
11: **else** $P_j(p) = \emptyset$
12: **end if**

B. Real-Time Bandwidth Allocation Problem

After each task deployment and routing for a long period is completed, the agent deployment and routing path of these tasks remain static in the long time interval. However, the traffic arrival of each task is dynamic and changes over time. To better meet the service needs and optimize objectives, we allocate bandwidth resources dynamically [30]. The bandwidth requirements for each task are adjusted in real-time based on incoming data and queue information, to minimize the end-to-end delay for each task within the constraints of available resources.

Under the premise that the deployment and routing paths of agent and DNN models in the long time interval are known, the optimization problem $P1$ can be transformed into

$$SP2: \min_{\mathbf{r}(t)} \; -Q^j(t) r_j(t) \tau + F^j(t) r_j(t) L + \frac{V}{|K_{AI}| r_j(t)} \max\left\{ \sum_{uv \in E^a} \sum_{n,n' \in N_j} y_{uv}^{jnn'}(p) l_{uv}^{jnn'}(p) \prod_{i=n_{j,1}}^{n} \alpha_i A_j(p) \tau \right.$$
$$\min \left\{ \sum_{c,v \in V^a} \sum_{cv \in E^a} \sum_{m \in M} x_v^{jm}(p) l_{cv}^{jm}(p) S_j^m(p) \right.$$

Algorithm 3 Dynamic Bandwidth Resource Allocation Algorithm

Require: $G^P, G^a, W_j(p), C_{uv}^{link}, \varphi_{uv}^{link}(t), P_j(p), R_j, I(t)$
Ensure: $r_j(t)$

```
Initialize r_j(t) = 0
for j in I(t) do
    Calculate the optimal bandwidth r^{j,opt}(t) of task j according to the path corresponding to the task and the real-time traffic.
    Determine the available bandwidth in the network min φ_uv^link(t) C_vu^link.
    if R^j ≥ r^{j,opt}(t) then
        r_j(t) = R^j
        Update network resources
    else if min φ_uv^link(t) C_vu^link + R^j ≥ r^{j,opt}(t) ≥ R^j then
        r_j(t) = r^{j,opt}(t)
        Update network resources
    else
        r_j(t) = min φ_uv^link(t) C_vu^link + R^j
        Update network resources
    end if
end for
```

$$\sum_{u,v\in V^a}\sum_{uv\in E^a}\sum_{m\in M}\delta_v^{jm}(p)x_v^{jm}(p)l_{uv}^{jm}(p)S_j^m(p)\Big\}\Big\}$$
$$\text{s.t. } C_5. \tag{40}$$

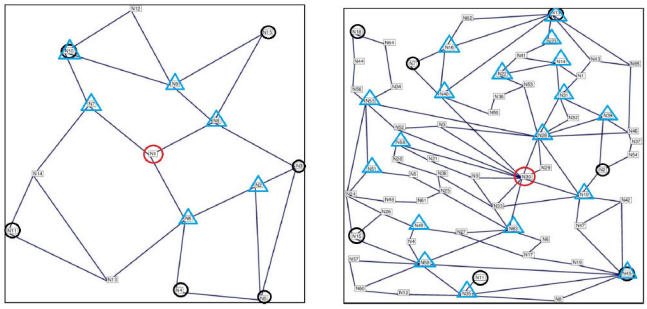
$SP2$ is a convex function with bandwidth as a variable. Therefore, the optimal solution $r^{j,\text{opt}}(t)$ can be obtained by calculating the partial derivative.

$$r_j(t)=\begin{cases} R^j, R^j \geq r^{j,\text{opt}}(t) \\ r^{j,\text{opt}}(t), \min\varphi_{uv}^{\text{link}}(t)C_{vu}^{\text{link}}+R^j \geq r^{j,\text{opt}}(t) \geq R^j \\ \min\varphi_{uv}^{\text{link}}(t)C_{vu}^{\text{link}}+R^j, r^{j,\text{opt}}(t) \\ \geq \min\varphi_{uv}^{\text{link}}(t)C_{vu}^{\text{link}}+R^j. \end{cases} \tag{42}$$

In the scenario where the allocated bandwidth remains constant at each hop along the route, it is essential to acknowledge that, during the reallocation of real-time bandwidth, the scalable bandwidth size for the task's bandwidth should be determined by considering the minimum remaining bandwidth across all paths traversed by the task.

C. Complexity Analysis

First, we select the DNN model and sort the task planning according to the requirements of arriving tasks. The complexity is $\mathcal{O}(|K_{AI}| + |K_{AI}|\log(|K_{AI}|))$. Then, for each task j, we first prune the original network according to the actual network resource topology information to obtain the physical network that can meet the task requirements, with the complexity of $\mathcal{O}(|V^P|+|E^P|)$. Then, create a directed weighted graph of task j, with the complexity of $\mathcal{O}(|N_j|^2|V^a|(|V^P| + |E^P|)\log(|V^P|))$. According to the directed weighted graph and the optimization goal, the routing path is obtained with the complexity of $\mathcal{O}(|V^a|(|V^a| + |E^a|)\log|V^a|)$. Finally, the time complexity of dynamic bandwidth allocation is $\mathcal{O}(|K_{AI}|)$. So the complexity of the entire JAAPD-D algorithm can be calculated as $\mathcal{O}(|K_{AI}|\log(|K_{AI}|) + |K_{AI}|(|N_j|^2|V^a|(|V^P| + |E^P|)\log(|V^P|) + |V^a|(|V^a| + |E^a|)\log|V^a|))$.



(a) (b)

Fig. 4. Network topology. (a) Atlanta. (b) ta2.

VI. Performance Evaluation

In this section, we verify the performance of the proposed algorithm.

A. Experimental Environment Settings

We use two real-world network topologies of different scales of the SNDlib website to conduct experimental simulation [50]. As shown in Fig. 4, we use a network topology called Atlanta, which is composed of 15 nodes and 22 links, and a network topology called ta2, which is composed of 65 nodes and 108 links. In both networks, the red-circled nodes are assumed to be the clouds, the black-circled nodes are the source of the AI services, and the blue-triangled nodes are assumed to be the autonomous nodes. We consider four types of agents: 1) FW; 2) data preprocessing; 3) data analysis; and 4) video rendering whose demanding computation resources are 400 MHz, 800 MHz, 1.6 GHz, and 2.4 GHz. The data change rate through each agent is 0.95, 0.8, 1.1, and 1.2, respectively. Each AI task has 1–3 agents with a fixed order. Table II lists the main simulation parameters [28], [51], [52], [53].

We refer to [12] and [14] and the actual data to set the performance metrics of the pretrained DNN model. The relationship between the DNN model and the accuracy of the task is shown in Table III. $g(x)$ is set as a piecewise function. If the data is noise-free (i.e., the data quality is good), $\omega_j = 1$, otherwise, $\omega_j = 0$

$$g(\omega_j)=\begin{cases}1, & \omega_j=1\\ 0.95, & \omega_j=0.\end{cases} \tag{43}$$

$$r^{j,\text{opt}}(t)=\sqrt{\frac{V\max\Big\{\sum_{uv\in E^a}\sum_{n,n'\in N_j}y_{uv}^{jnn'}(t)l_{uv}^{jnn'}(t)\prod_{i=n_{j,1}}^{n}\alpha_i A_j(p)\tau\,,\min\{\sum_{c,v\in V^a}\sum_{cv\in E^a}\sum_{m\in M}x_v^{jm}(p)l_{cv}^{jm}(p)S_j^m(p),\sum_{u,v\in V^a}\sum_{uv\in E^a}\sum_{m\in M}\delta_v^{jm}(p)x_v^{jm}(p)l_{uv}^{jm}(p)S_j^m(p)\}\Big\}}{|K_{AI}|\big(-Q^j(t)\tau+F^j(t)L\big)}} \tag{41}$$

TABLE II
SIMULATION PARAMETERS OF NETWORK AND SERVICES

parameter	value
link capacity C_{vu}^{link} (edge, net, cloud)	5Gbps,10Gbps,100Gbps
computation capacity C_v^{cpu} (autonomous node, cloud)	[8,16]GHz,80GHz
memory capacity C_v^{mem} (autonomous node, cloud)	50GB,100GB
The cost of computation resourceM^{cpu}	[5*10^5, 10^6]
The cost of memory resourceM^{mem}	[5*10^5, 10^6]
Total time length T	200s
Long time interval length T_0	10s
Short time interval length τ	1s
The requirements of accuracy η_j	[0.7, 0.9]
The requirements of bandwidth R_j	[10, 30]Mbps
User mean arrival rate $A_j(t)$	[500, 2000]Kbps
Number of users U	40
control vector V	5*10^{13}
CPU density L	297.62 cycles/bit

TABLE III
SIMULATION PARAMETERS OF DNN MODEL

	Data size $S_j^m(t)$	**Accuracy factor** ε_j^m
Uncompressed DNN	[180, 200]MB	[0.95, 0.99]
Compressed DNN	[20, 30]MB	[0.7, 0.9]

B. Benchmark

Due to the NP-hard nature of the problem, obtaining an optimal solution is infeasible. However, our algorithm can provide an optimized solution. In comparison to the following contrastive algorithm, it achieves improvements in terms of delay, accepted rate, throughput, and cost.

1) *JAAPD-S:* This algorithm is part of the proposed JAAPD-D algorithm that does not consider real-time bandwidth allocation.
2) *JAAPD-C:* The JAAPD-C algorithm places the DNN model on the cloud to perform task inference directly without dynamically deploying, i.e., the optimization objective does not consider the dynamic placement of the DNN model.
3) *DVPRA [53]:* DVPRA constructs a general heuristic optimization structure called VNF split multilevel edge weight graph, and uses the Lagrange multiplier method to consider the optimization objective. To make DVPRA comparable, we only consider the delay requirement and add it to *P*0.
4) *DGASP [54]:* The algorithm uses an aerial node to assist in transmission by connecting with the nodes that have computing capabilities. It employs VNF embedding based on greedy-matching and Dijkstra-based routing. However, it does not optimize the embedded cost.
5) *SPO [40]:* Introduces a heuristic called one-step ahead to efficiently solve the multiple-period model SPO. The objective is to deploy DNN models on both edge servers and the cloud, aiming to minimize costs while meeting delay constraints.

C. Performance Comparison

Fig. 5 depicts the variation in the backlog of communication and computation queues over time. The Lyapunov optimization method proposed in this article is designed to ensure a smooth accumulation of queues within the network, thereby safeguarding the overall stability of the entire system and mitigating the risk of persistent service backlogs. The graph demonstrates that the proposed JAAPD algorithm maintains queue stability over extended time periods. As time progresses, queue congestion does not exhibit continuous and infinite growth, ensuring system stability over prolonged intervals and averting the potential for network congestion.

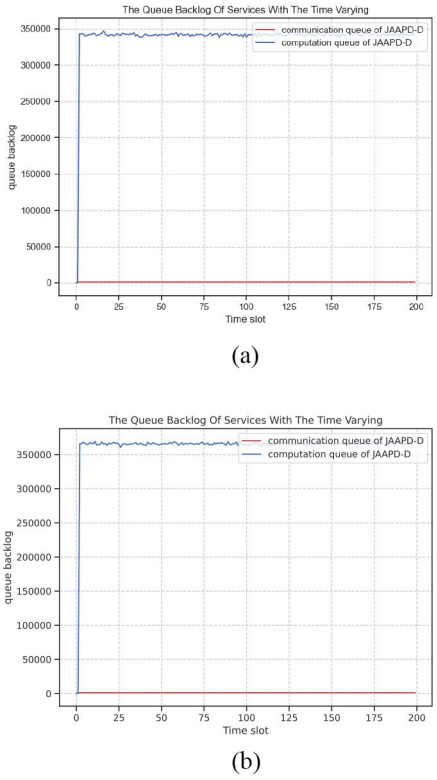
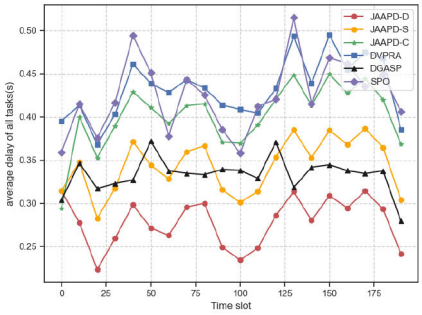
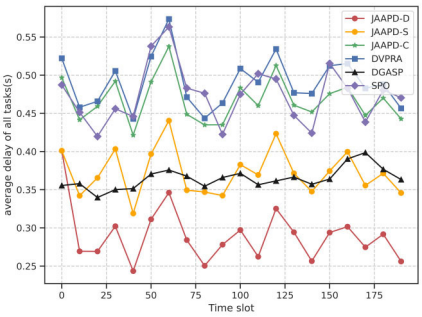


Fig. 5. Communication and computation queue backlog versus the length of time slots. (a) Atlanta. (b) ta2.

Fig. 6 shows the change of user average end-to-end delay over time in different network topologies. As can be seen from Fig. 6, the delay of each algorithm is relatively stable over time. This conclusion holds for larger network topologies as well. The average end-to-end delay of JAAPD-D is significantly reduced by 52.6% and 26.7% compared with DVPRA and DGASP in Fig. 6(a). The JAAPD-C algorithm, DVPRA algorithm, and SPO algorithm exhibit similar delay, primarily because all three algorithms concurrently consider delay and cost balancing when selecting agent placement. Additionally, they all tend to place DNN models in the cloud, resulting in minimal distinctions in the paths for agent placement in smaller network topologies, hence their closely aligned delay. In contrast, the DGASP algorithm takes into account user delay constraints and utilizes auxiliary air nodes to reduce the number of hops in service path routing, resulting in comparatively lower delay among the compared algorithms. Its delay performance is on par with that of the JAAPD-S algorithm. In TA2, the average delay of each algorithm increases, mainly due to the growing network size, leading to an increase in the number of hops in service paths and subsequent delay

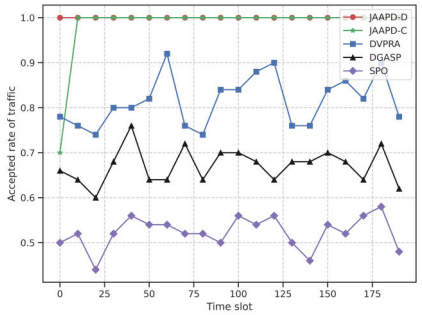


(a)

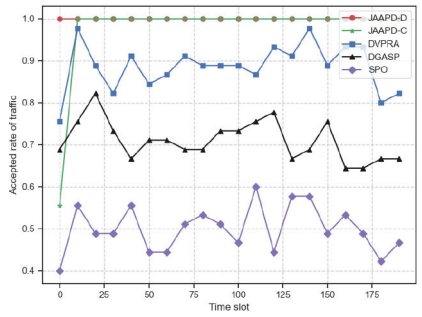


(b)

Fig. 6. Average delay versus the length of time slots. (a) Atlanta. (b) ta2.



(a)



(b)

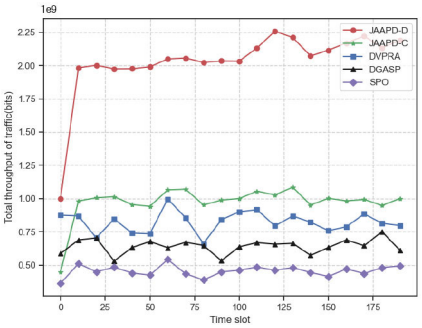
Fig. 7. Accepted rate versus the length of time slots. (a) Atlanta. (b) ta2.

changes. As the network topology expands and becomes more complex, our proposed algorithm gains an advantage due to its ability to dynamically adjust bandwidth based on network and service requirements, thereby minimizing delay and ensuring that service delay consistently remains at a lower level.

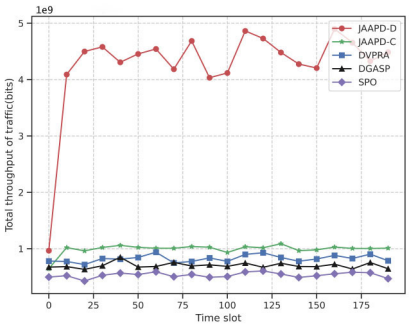
Fig. 7 illustrates the comparison of traffic accepted rates over time. The traffic accepted rate is the proportion of the users who are accepted compared to the total number of users. To rigorously control variables and investigate the relationship between traffic accepted rate and network resources allocation method, we ensure that all services can reliably select models meeting their accuracy requirements. However, the actual accessibility of services to the network and their successful completion hinged upon the orchestration of network resources. Since the proposed JAAPD-D algorithm can dynamically select DNN models and place agents, it can ensure the accepted rate of traffic. This substantiates that effective agent deployment and resource allocation take place for incoming service after entering the network. Consequently, the network can concurrently handle a greater number of services. In contrast, the JAAPD-C algorithm exhibits a lower accepted rate at the beginning of the time slot in the graph. This is attributed to the absence of cost queue backlog, and the service tends to find the shortest delay path. Therefore, when the DNN model is deployed in the cloud, agents also tend to be deployed in the cloud to reduce transmission delay. If the cloud resources are exhausted, the number of accepted users will decrease. The JAAPD algorithm considers cost variations based on the changes in the cost queue, avoiding the placement of all agents in a single autonomous node or the cloud. This results in a high accepted rate in subsequent time periods. Additionally, as DVPRA can dynamically place agents within autonomous nodes, its accepted rate surpasses that of DGASP. DGASP needs to consider user delay deadlines, so if the time limit is exceeded, it will reject traffic acceptance and tend to place all agents in the same location. SPO, on the other hand, leans toward placing agents at edge nodes or in the cloud, hence its accepted rate is limited.

Fig. 8 demonstrates the comparison of traffic throughput in the network. Due to the high user acceptance rate of the JAAPD-D algorithm, it exhibits the highest throughput. It can be observed that the throughput of the JAAPD-C algorithm is only slightly higher than the other algorithms, while the system throughput of the proposed algorithm is at least twice that of DVPRA. In comparison to the JAAPD-C algorithm, the dynamic placement of DNN models and dynamic bandwidth resource allocation in JAAPD-D contribute to its higher throughput.

Fig. 9 depicts the cost queue backlog comparison over time. The cost of the comparison algorithm DGASP and DVPRA increases over time. In contrast, the proposed JAAPD algorithm and SPO maintain a stable cost queue over extended time periods, ensuring that costs are controlled and do not grow infinitely. Since SPO considers cost issues and minimizes costs as an objective, its costs are relatively low. Although the DVPRA algorithm considers cost factors when dynamically placing agents, it does not account for long-term considerations, resulting in consistently high costs. The DGASP algorithm, due to its lower acceptance rate, may not incur

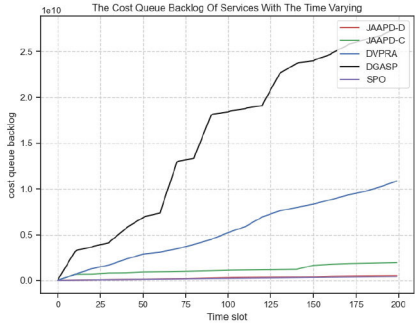


(a)

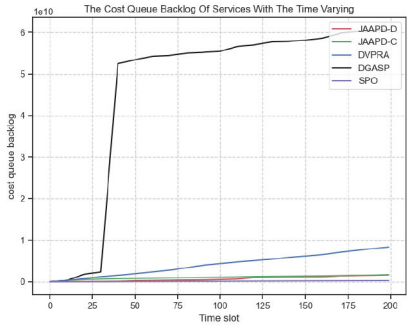


(b)

Fig. 8. Total throughput versus the length of time slots. (a) Atlanta. (b) ta2.



(a)



(b)

Fig. 9. Cost queue backlog versus the length of time slots. (a) Atlanta. (b) ta2.

high costs in the short term. However, as time progresses, its preference for placing agents in a single node may lead to rapid cost escalation.

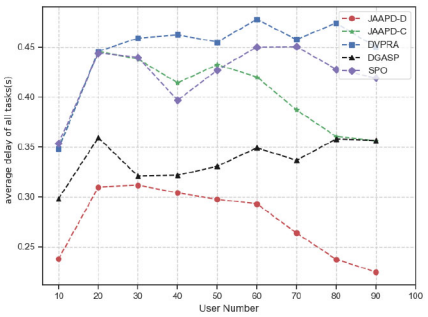


Fig. 10. Average delay versus user number.

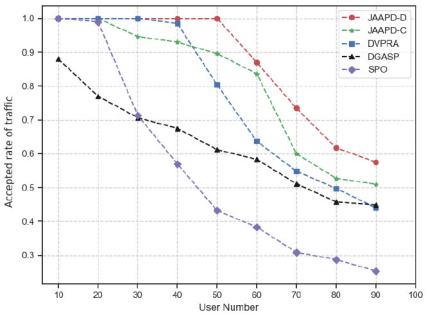


Fig. 11. Accepted rate versus user number.

We explore the relationship between the number of users and network performance in the Atlanta network, where $V = 5 \times 10^{13}$, and $T = 50$. Fig. 10 illustrates the variation of end-to-end delay with the number of users. JAAPD-D consistently maintains lower service delay. When the number of users is low, the overall service in the network is limited, resulting in relatively small delay for all algorithms. However, as the number of users increases, the average delay of comparative algorithms shows no significant change, whereas the trend in the proposed JAAPD algorithm is to reduce delay with an increasing number of users. With the growth in the number of users, the delay of DVPRA, DGASP, and SPO remain almost constant due to a decrease in traffic accepted rates, keeping the actual number of users processed in the network constant. JAAPD-D can effectively manage resources as the number of users increases and exhibits better delay performance with an increasing user count.

Fig. 11 validates that the change of traffic accepted rate is controlled by the number of users. When the number of users increases, the accepted rate of all algorithms decreases due to the limited communication, computation, and memory resources in the network. JAAPD-D has always maintained a high traffic accepted rate. Similarly, as shown in Fig. 12, the throughput of JAAPD-D has always maintained a high level. When the number of users is small, the throughput in the network increases with the increase of the number of users. When the number of users increases to the point where the network cannot load, the system throughput will tend to be stable or even less.

The relationship between V and network performance in the Atlanta network with the number of users $U = 40$ and $T = 50$ is considered in Fig. 13 which shows

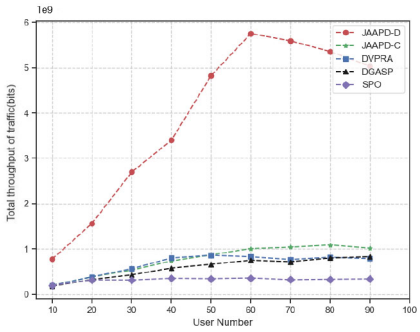


Fig. 12. Total throughput versus user number.

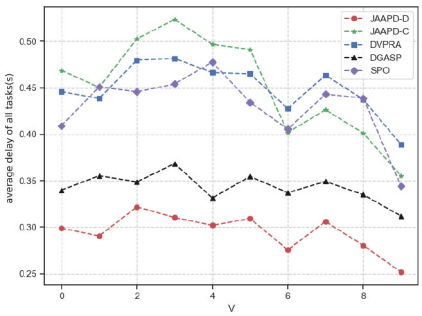


Fig. 13. Average delay versus V.

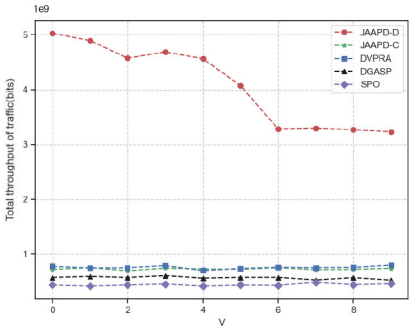


Fig. 14. Total throughput versus V.

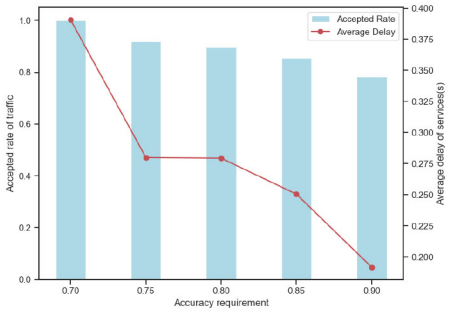


Fig. 15. Average delay and accepted rate versus accuracy requirements.

the average delay with different control factors $V = [10^2, 10^4, 10^6, 10^8, 10^{10}, 10^{12}, 10^{14}, 10^{16}, 10^{18}, 10^{20}]$. Fig. 14 shows the system throughput with different control factors. Combining the information from two figures, significant variations in delay and throughput are observed when V falls within the interval $[10^{10}, 10^{14}]$. During this range, V is actively balancing the overall delay and system throughput. When V is relatively small or large, it plays a decisive role for a specific variable, resulting in less significant overall changes. The delay of our proposed algorithm consistently remains low. This is attributed to the dynamic bandwidth algorithm, which, considering the control factors V and task requirements, minimizes delay as much as possible. However, through the ablation experiment of the JAAPD-C algorithm, the influence of V becomes evident. The JAAPD-C algorithm is significantly more affected by V; when V is small, its average service delay is higher than that of DVPRA. As V increases, delay decreases because service considerations involve fewer cost factors and tend to favor placement and routing solutions with lower delay. On the other hand, when V changes, throughput is only affected by JAAPD-D because V determines real-time bandwidth allocation. When bandwidth allocation is affected, significant changes in system throughput occur.

To investigate the impact of different service accuracy requirements on network performance, we also explore the variations in traffic accepted rate and average service delay with changing accuracy requirements in the Atlanta network, with an user number (U) of 60 and 50 time slots (T). Fig. 15 illustrates the traffic accepted rate and average service delay for accuracy requirements in the range [0.7, 0.75, 0.8, 0.85, 0.9]. We consider services with identical accuracy requirements, yet due to variations in the data quality of different services, model selection differs to meet accuracy demands. As observed from Fig. 11, when the user number is 40 or lower, the proposed algorithm can essentially guarantee access to all services. Therefore, to highlight disparities, we set the user number to 60. In this scenario, when service accuracy requirements are randomly distributed between [0.7, 0.9], network resources cannot support access to all services. However, from Fig. 15, if service accuracy requirements are all 0.7, all services can be accepted. As accuracy requirements increase, more services necessitate uncompressed models to ensure accuracy, leading to larger models consuming more network resources, consequently reducing the number of services the network can accommodate. Furthermore, when services adopt uncompressed DNN models, they have more neurons and layers and consume more computation resources, reducing service computation delay, while transmission delay remains relatively stable with constant traffic in the network. Therefore, from a holistic perspective, as service accuracy requirements increase, the average service delay decreases. At accuracy levels of 0.75 and 0.8, where data quality impacts service selection, the probability of selecting two types of DNN models is relatively balanced, resulting in similar average service delays. This observation further validates the influence of AI service accuracy requirements on its transmission performance.

VII. Conclusion

In this article, an AI service management and network resource scheduling architecture based on HLN is proposed. The allocation of communication, computation and memory resources in the network and the QoS requirements of AI

services are fully considered, aiming to ensure network stability while minimizing end-to-end service delay. Lyapunov optimization is used to transform the original optimization problem into a long-term and short-term resource allocation scheme, we propose a JAAPD and dynamic bandwidth resource allocation algorithm, adjust agent placement and DNN deployment and service routing path planning in the long term, and dynamically allocate bandwidth resource to ensure service performance. Simulation results show that the proposed algorithm outperforms the comparison algorithm in terms of delay, traffic accepted rate, network system throughput and cost.

APPENDIX
PROOF OF THEOREM 1

The inequality

$$\{\max[Q-R,0]+A\}^2 \le Q^2+R^2+A^2-2Q(R-A) \quad (44)$$

always hold.

For $Q^j(t)$, we have

$$Q^j(p+T_0-1)^2 \le Q^j(p)^2+\sum_{t=p}^{p+T_0-1}\left[r_j(t)\tau\right]^2 + \sum_{t=p}^{p+T_0-1}\left[\rho_j A_j(p)\tau\right]^2 - 2\sum_{t=p}^{p+T_0-1} Q^j(t)\Big[r_j(t)\tau-\rho_j A_j(p)\tau\Big]. \quad (45)$$

For $F^j(t)$, similarly

$$F^j(p+T_0-1)^2 \le \sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p)^2 F^j(p)^2+\beta_{j,m}^{\mathrm{cpu}}(p)^2 + \sum_{t=p}^{p+T_0-1}\left[Lr_j(t)\right]^2 - 2\sum_{t=p}^{p+T_0-1} F^j(t)\Big[\beta_{j,m}^{\mathrm{cpu}}(p)-Lr_j(t)\Big]. \quad (46)$$

For $E_v(t)$

$$E_v(p+T_0-1)^2 \le E_v(p)^2+\sum_{t=p}^{p+T_0-1}(Z^r)^2+\sum_{t=p}^{p+T_0-1} Z_v(t)^2 - 2\sum_{t=p}^{p+T_0-1} E_v(t)\big[Z^r-Z_v(t)\big]. \quad (47)$$

According to the inequality (45), (46), and (47), $\Delta(L(\boldsymbol{\Theta}(p)))$ can be calculated as

$$\begin{aligned} &L(\boldsymbol{\Theta}(p+T_0-1))-L(\boldsymbol{\Theta}(p)) \\ &\le \frac{1}{2}\sum_{t=p}^{p+T_0-1}\left\{\left[\sum_{j\in K_{AI}} r_j(t)\tau\right]^2+\left[\sum_{j\in K_{AI}}\rho_j A_j(p)\tau\right]^2 \right. \\ &\quad +\left[\sum_{j\in K_{AI}}\sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p)\beta_{j,m}^{\mathrm{cpu}}(p)\right]^2 \\ &\quad +\left[\sum_{j\in K_{AI}}\sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p) r_j(t) L\right]^2 \\ &\quad \left.+\left[\sum_{v\in V^a} Z^r\right]^2+\left[\sum_{v\in V^a} Z_v(t)\right]^2\right\} \\ &\quad -\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}} Q^j(t)\Big[r_j(t)\tau-\rho_j A_j(p)\tau\Big] \\ &\quad -\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}}\sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p)^2\left\{F^j(t)\Big[\beta_{j,m}^{\mathrm{cpu}}(p)-Lr_j(t)\Big]\right\} \\ &\quad -\sum_{t=p}^{p+T_0-1}\sum_{v\in V^a} E_v(t)\big[Z^r-Z_v(t)\big]. \end{aligned} \quad (48)$$

The upper bound of the Lyapunov drift-plus-penalty function can be expressed as

$$\begin{aligned} &\Delta(L(\boldsymbol{\Theta}(p)))+V\mathbb{E}\big\{D^E(p)|\boldsymbol{\Theta}(p)\big\} \\ &\le C-\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}} Q^j(t)\mathbb{E}\Big[r_j(t)\tau-\rho_j A_j(p)\tau|\boldsymbol{\Theta}(p)\Big] \\ &\quad -\sum_{t=p}^{p+T_0-1}\sum_{j\in K_{AI}}\sum_{v\in V^a}\sum_{m\in M} F^j(t)\mathbb{E}\Big\{x_v^{jm}(p)^2\Big[\beta_{j,m}^{\mathrm{cpu}}(p)-Lr_j(t)\Big]|\boldsymbol{\Theta}(p)\Big\} \\ &\quad -\sum_{t=p}^{p+T_0-1}\sum_{v\in V^a} E_v(t)\mathbb{E}\big[Z^r-Z_v(t)|\boldsymbol{\Theta}(p)\big] \\ &\quad +\frac{V}{|K_{AI}|}\mathbb{E}\left[\sum_{j\in K_{AI}} D_j^E(p)|\boldsymbol{\Theta}(p)\right] \end{aligned} \quad (49)$$

where

$$\begin{aligned} C \ge \frac{1}{2}\sum_{t=p}^{p+T_0-1}&\left\{\left[\sum_{j\in K_{AI}} r_j(p)\tau\right]^2+\left[\sum_{j\in K_{AI}}\rho_j A_j(p)\tau\right]^2\right. \\ &+\left[\sum_{j\in K_{AI}}\sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p)\beta_{j,m}^{\mathrm{cpu}}(p)\right]^2 \\ &+\left[\sum_{j\in K_{AI}}\sum_{v\in V^a}\sum_{m\in M} x_v^{jm}(p) r_j(t) L\right]^2 \\ &\left.+\left[\sum_{v\in V^a} Z^r\right]^2+\left[\sum_{v\in V^a} Z_v(t)\right]^2\right\} \end{aligned} \quad (50)$$

This completes the proof of Theorem 1.

REFERENCES

[1] S. Mhatre, F. Adelantado, K. Ramantas, and C. Verikoukis, "AIaaS for ORAN-based 6G Networks: Multi-time scale slice resource management with DRL," 2023, *arXiv:2311.11668.*

[2] S. Lins, K. D. Pandl, H. Teigeler, S. Thiebes, C. Bayer, and A. Sunyaev, "Artificial intelligence as a service: Classification and research directions," *Bus. Inf. Syst. Eng.*, vol. 63, pp. 441–456, Aug. 2021.

[3] X. Zhou, Y. Zhang, Z. Li, X. Wang, J. Zhao, and Z. Zhang, "Large-scale cellular traffic prediction based on graph convolutional networks with transfer learning," *Neural Comput. Appl.*, vol. 34, pp. 5549–5559, Apr. 2022.

[4] H. Bai, Y. Zhang, Z. Zhang, and S. Yuan, "Latency equalization policy of end-to-end network slicing based on reinforcement learning," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 1, pp. 88–103, Mar. 2023.

[5] S. Wang, C. Yuen, W. Ni, Y. L. Guan, and T. Lv, "Multiagent deep reinforcement learning for cost-and delay-sensitive virtual network function placement and routing," *IEEE Trans. Commun.*, vol. 70, no. 8, pp. 5208–5224, Aug. 2022.

[6] G. Zhu et al., "Pushing AI to wireless network edge: An overview on integrated sensing, communication, and computation towards 6G," *Sci. China Inf. Sci.*, vol. 66, no. 3, 2023, Art. no. 130301.

[7] J. Wu, S. Guo, J. Li, and D. Zeng, "Big data meet green challenges: Big data toward green applications," *IEEE Syst. J.*, vol. 10, no. 3, pp. 888–900, Sep. 2016.

[8] J. Pan, L. Cai, S. Yan, and X. S. Shen, "Network for AI and AI for network: Challenges and opportunities for learning-oriented networks," *IEEE Netw.*, vol. 35, no. 6, pp. 270–277, Nov./Dec. 2021.

[9] M. Li, J. Gao, C. Zhou, X. S. Shen, and W. Zhuang, "Slicing-based artificial intelligence service provisioning on the network edge: Balancing AI service performance and resource consumption of data management," *IEEE Veh. Technol. Mag.*, vol. 16, no. 4, pp. 16–26, Dec. 2021.

[10] W. Zhang, D. Yang, and H. Wang, "Data-driven methods for predictive maintenance of industrial equipment: A survey," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2213–2227, Sep. 2019.

[11] W. Zhang, D. Yang, Y. Xu, X. Huang, J. Zhang, and M. Gidlund, "Deep-Health: A self-attention based method for instant intelligent predictive maintenance in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5461–5473, Aug. 2021.

[12] W. Zhang et al., "Deep reinforcement learning based resource management for DNN inference in industrial IoT," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 7605–7618, Aug. 2021.

[13] M. Yao, L. Chen, Y. Wu, and J. Wu, "Loading cost-aware model caching and request routing in edge-enabled wireless sensor networks," *Comput. J.*, vol. 66, no. 10, pp. 2409–2425, 2023.

[14] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen, "Accuracy-guaranteed collaborative DNN inference in industrial IoT via deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4988–4998, Jul. 2021.

[15] W. Fan et al., "DNN deployment, task offloading, and resource allocation for joint task inference in IIoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1634–1646, Feb. 2023.

[16] B. Yang, X. Cao, X. Li, Q. Zhang, and L. Qian, "Mobile-edge-computing-based hierarchical machine learning tasks distribution for IIoT," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2169–2180, Mar. 2020.

[17] N. Hudson, H. Khamfroush, and D. E. Lucani, "QoS-aware placement of deep learning services on the edge with multiple service implementations," in *Proc. Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2021, pp. 1–8.

[18] W. Q. Ren et al., "A survey on collaborative DNN inference for edge intelligence," *Mach. Intell. Res.*, vol. 20, no. 3, pp. 370–395, 2023.

[19] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of Yolo algorithm developments," *Procedia Comput. Sci.*, vol. 199, pp. 1066–1073, Jan. 2022.

[20] J. Jeong, H. Park, and N. Kwak, "Enhancement of SSD by concatenating feature maps for object detection," 2017, *arXiv:1705.09587*.

[21] Z. Xu, Z. Zhang, W. Liang, Q. Xia, O. Rana, and G. Wu, "QoS-aware VNF placement and service chaining for IoT applications in multi-tier mobile edge networks," *ACM Trans. Sens. Netw.*, vol. 16, no. 3, pp. 1–27, 2020.

[22] H. Song, J. Bai, Y. Yi, J. Wu, and L. Liu, "Artificial intelligence enabled Internet of Things: Network architecture and spectrum access," *IEEE Comput. Intell. Mag.*, vol. 15, no. 1, pp. 44–51, Feb. 2020.

[23] W. Wu et al., "AI-native network slicing for 6G networks," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 96–103, Feb. 2022.

[24] M. Tariq, F. Naeem, and H. V. Poor, "Toward experience-driven traffic management and orchestration in digital-twin-enabled 6G networks," 2022, *arXiv:2201.04259*.

[25] *Framework of Human-Like Networking*, ITU-Rec. T Y. 3680, Int. Telecommun. Union, Geneva, Switzerland, Accessed: Feb. 13, 2022. [Online]. Available: https://handle.itu.int/11.1002/1000/14863

[26] E. T. M. Beltrán et al., "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 4, pp. 2983–3013, 4th Quart., 2023.

[27] J. Pei, P. Hong, K. Xue, D. Li, D. S. L. Wei, and F. Wu, "Two-phase virtual network function selection and chaining algorithm based on deep learning in SDN/NFV-enabled networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1102–1117, Jun. 2020.

[28] S. Yang, F. Li, Z. Zhou, X. Chen, Y. Wang, and X. Fu, "Online control of service function chainings across geo-distributed datacenters," *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3558–3571, Jun. 2023.

[29] H. Chen, S. Wang, G. Li, L. Nie, X. Wang, and Z. Ning, "Distributed orchestration of service function chains for edge intelligence in the industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6244–6254, Sep. 2022.

[30] Z. Zhou et al., "Collaborative learning-based network resource scheduling and route management for multi-mode green IoT," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 928–939, Jun. 2023.

[31] Y. Sun, X. Chi, B. Yu, S. Zhao, S. Li, and Q. Meng, "Hop-by-hop bandwidth allocation and deployment for SFC with end-to-end delay QoS guarantees," *Comput. Commun.*, vol. 192, pp. 256–267, Aug. 2022.

[32] Y. Yang et al., "Task-oriented 6G native-AI network architecture," *IEEE Netw.*, vol. 38, no. 1, pp. 219–227, Jan. 2024.

[33] S. Mao, J. Wu, L. Liu, D. Lan, and A. Taherkordi, "Energy-efficient cooperative communication and computation for wireless powered mobile-edge computing," *IEEE Syst. J.*, vol. 16, no. 1, pp. 287–298, Mar. 2022.

[34] Y. Zhang, Y. Hu, and T. Ma, "Stability-oriented RAN slicing based on joint communication and computation offloading," *IET Commun.*, vol. 16, no. 7, pp. 772–785, 2022.

[35] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.

[36] W. Y. B. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[37] L. Fu, H. Zhang, G. Gao, M. Zhang, and X. Liu, "Client selection in federated learning: Principles, challenges, and opportunities," *IEEE Internet Things J.*, vol. 10, no. 24, pp. 21811–21819, Dec. 2023.

[38] J. Ren, G. Yu, and G. Ding, "Accelerating DNN training in wireless federated edge learning systems," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 219–232, Jan. 2021.

[39] Z. Wang, H. Xu, J. Liu, Y. Xu, H. Huang, and Y. Zhao, "Accelerating federated learning with cluster construction and hierarchical aggregation," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3805–3822, Jul. 2023.

[40] G. Premsankar and B. Ghaddar, "Energy-efficient service placement for latency-sensitive applications in edge computing," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17926–17937, Sep. 2022.

[41] Y. Qiu, J. Liang, V. C. M. Leung, and M. Chen, "Online security-aware and reliability-guaranteed AI service chains provisioning in edge intelligence cloud," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5933–5948, May 2024.

[42] C. Liu et al., "A SDN-based active measurement method to traffic QoS sensing for smart network access," *Wireless Netw.*, vol. 27, pp. 3677–3688, Jul. 2021.

[43] D. Zhou, Z. Yan, G. Liu, and M. Atiquzzaman, "An adaptive network data collection system in SDN," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 562–574, Jun. 2020.

[44] N. Hu, F. Luan, X. Tian, and C. Wu, "A novel SDN-based application-awareness mechanism by using deep learning," *IEEE Access*, vol. 8, pp. 160921–160930, 2020.

[45] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2016, *arXiv:1510.00149*.

[46] J. Feng, Q. Pei, F. R. Yu, X. Chu, J. Du, and L. Zhu, "Dynamic network slicing and resource allocation in mobile edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7863–7878, Jul. 2020.

[47] R. M. Karp, "Reducibility among combinatorial problems," in *50 Years of Integer Programming 1958-2008*, M. Jänger et al., Berlin, Germany: Springer, 2010.

[48] B. Chachuat, "Mixed-integer linear programming (MILP): Model formulation," *McMaster Univ. Dep. Chem. Eng.*, vol. 17, pp. 1–26, Jul. 2019.

[49] "Opendaylight." Accessed: Mar. 25, 2020. [Online]. Available: https://www.opendaylight.org/

[50] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0-survivable network design library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010.

[51] X. Gao, X. Huang, S. Bian, Z. Shao, and Y. Yang, "PORA: Predictive offloading and resource allocation in dynamic fog computing systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 72–87, Jan. 2020.

[52] J. Li, W. Liang, M. Huang, and X. Jia, "Reliability-aware network service provisioning in mobile edge-cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1545–1558, Jul. 2020.

[53] L. Liu, S. Guo, G. Liu, and Y. Yang, "Joint dynamical VNF placement and SFC routing in NFV-enabled SDNs," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4263–4276, Dec. 2021.
[54] G. Wang, S. Zhou, S. Zhang, Z. Niu, and X. Shen, "SFC-based service provisioning for reconfigurable space-air-ground integrated networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 7, pp. 1478–1489, Jul. 2020.

Yuqi Hu received the B.E. degree from Beijing University of Posts and Telecommunications, Beijing, China, where she is currently pursuing the M.S. degree with the School of Electronic Engineering.

Her research interests include 6G networks, resource allocation, and Lyapunov optimization.

Qin Li received the master's degree from Chongqing University, Chongqing, China, in 2002.

She is a Senior Researcher with China Mobile Research Institute, Beijing, China, and has engaged in technical and application research on core network and content network for more than 20 years. Her research interests cover 5G/6G architecture, network intelligence, and digital twin network.

Yuhao Chai is currently pursuing the Ph.D. degree with Beijing University of Posts and Telecommunications, Beijing, China.

His research interests include game theory, 6G networks, and resource allocation.

Di Wu is currently pursuing the master's degree with Beijing University of Posts and Telecommunications, Beijing, China.

Her research interests include machine learning, computing power networks, and deterministic networks.

Lu Lu (Member, IEEE) received the master's degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2005.

She is the Deputy Director of the Department of Basic Network Technology, China Mobile Research Institute, Beijing, the Leader of Core Network Group of CCSA TC5, and the Vice-Chairman of ITU-T SG13. Her main research direction covers 5G/6G architecture and computing power network.

Nanxiang Shi received the master's degree from Beihang University, Beijing, China, in 2013.

He is a Principal Researcher and a Technical Manager of China Mobile Research Institute, Beijing. He is actively participating in communications standardization activities and currently serves as a Rapporteur of ITU-T SG13 Q23 (Networks beyond IMT-2020: Fixed, Mobile and Satellite Convergence) and the Vice Chair of CCSA TC12 WG3 (Collaborative Networking Technologies). His research area includes 6G network, 5G-advanced network, and satellite and terrestrial converged network.

Yinglei Teng (Senior Member, IEEE) received the B.S. degree from Shandong University, Jinan, China, in 2005, and the Ph.D. degree in electrical engineering from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2011.

She is currently an Associate Professor with the School of Electronic Engineering, BUPT. Her current research interests include UDNs and massive MIMO, IoTs, and blockchains.

Yong Zhang received the Ph.D. degree from Beijing University of Posts and Telecommunications (BUPT), Beijing, China in 2007.

He is a Professor with the School of Electronic Engineering, BUPT. He is currently the Director of Fab.X Artificial Intelligence Research Center, BUPT. He is the Deputy Head of the Mobile Internet Service and Platform Working Group, China Communications Standards Association. He has authored or co-authored more than 80 papers and holds 30 granted China patents. His research interests include artificial intelligence, wireless communication, and Internet of Things.