Towards Native Intelligence: An Efficient and Flexible AI Services Provision Scheme in Multi-layer Heterogeneous Networks

Jingchao He, *Student Member*, *IEEE*, Nan Cheng, *Senior Member*, *IEEE*, Ruijin Sun, *Member*, *IEEE*, Ruqian Zhang, Conghao Zhou, *Member*, *IEEE*, Wei Quan, *Senior Member*, *IEEE*, Changle Li, *Senior Member*, *IEEE*,

***Abstract*—To fulfill future diverse user requirements, 6G networks are envisioned to provide everyone-centric customized services ubiquitously and precisely. However, the diversity in user requirements and the heterogeneity in network resources challenge conventional network operators in network management and service provision. In this paper, we investigate the artificial intelligence (AI) service provision in the multi-layer heterogeneous network. To provide ubiquitous intelligence to users with different computing requirements, an intelligence-native network architecture is designed. Based on the proposed architecture and the AI model stitching mechanism, we formulate the joint AI provision and access selection problem as a mixed integer nonlinear programming (MINLP) problem to maximize the average user satisfaction value and user satisfaction rate. Then, a heuristic solution based on Dung Beetle algorithm is proposed to optimize the AI model selection, AI service deployment, user access, and stitching coefficient jointly. Extensive simulations are conducted to evaluate the performance of our proposed architecture and algorithm.**

***Index Terms*—User-centric service, artificial intelligence (AI) model, multi-layer network, service deployment.**

I. INTRODUCTION

THE rapid development of mobile communication networks has led to an exponential increase in the number of mobile devices, with projections estimating this number will reach 19.08 billion by 2025 and 29 billion by 2030 [1]. These devices continuously generate unprecedented volume of data, which involves inestimable potential values in future service fulfillment, network optimization, etc. However, the fundamental bottlenecks in processor speed, memory size, battery life, and heat dissipation limit the data processing in mobile devices [2], [3]. Since ChatGPT's launch by OpenAI on November 30, 2022, large language model (LLM) technologies have spurred an artificial intelligence (AI) renaissance. ChatGPT excels across various domains like content creation, image recognition, math computations, and code generation, often outperforming human experts. Industries are not only building their own large models, like Google's AlphaFold for biomedicine [4], IBM's Watson for business [5], and ByteDance's LlamaGen for art and graph design [6], but users are also keen on localizing and customizing these models for enhanced daily productivity.

Deploying LLMs on the user side ensures that their capabilities are accessible anytime and anywhere, even in environments with limited or no network connectivity. However, limited hardware capacity presents a significant challenge for deploying LLMs on mobile devices, while offloading data to cloud data centers can lead to unacceptable delays and high transmission costs [7]. In recent years, researchers have proposed deploying computational resources at the network edge and developing a mobile edge computing (MEC) architecture [8]–[10]. Nevertheless, due to time-varying and unpredictable user distributions, installing suitable computing units on edge servers is prohibitively expensive, especially compared to the overpowered units found in centralized server clusters such as Microsoft Azure or distributed idle terminals. Therefore, an innovative multi-layer heterogeneous network architecture is urgently needed to effectively incorporate dispersed network resources and enable flexible resource utilization for future users [11].

Despite the advancements in the Fifth Generation (5G) mobile system [12], the pre-defined "standard" services like enhanced Mobile Broadband (eMBB), Ultra-Reliable Low Latency Communications (URLLC), and massive Machine Type Communications (mMTC) are not flexible enough to cater to the diverse and dynamic needs of users. The traditional tunnel-based serving mode in 5G ensures good average performance but struggles to guarantee Quality of Experience (QoE) for everyone due to varying user tasks and requirements. Therefore, the Sixth Generation (6G) mobile system aims to provide "everyone-centric" customized services anywhere and anytime, utilizing heterogeneous network resources and pervasive AI [13]. This shift from 5G's "technology-driven" to

This work was supported by the National Key Research and Development Program of China (2020YFB1807700), and the Fundamental Research Funds for the Central Universities and the Innovation Fund of Xidian University under Grant YJSJ24017.

Jingchao He, Nan Cheng (*corresponding author*), Ruijin Sun, and Changle Li are with the State Key Laboratory of ISN and School of Telecommunications Engineering, Xidian University, Xi'an 710071, China (e-mail: jchhe@stu.xidian.edu.cn; dr.nan.cheng@ieee.org; sunruijin@xidian.edu.cn; clli@mail.xidian.edu.cn).

Ruqian Zhang is with the AI and Intelligent Operation Center, China Mobile Research Institute, Xi'an 710071, China (e-mail: zhangruqian@chinamobile.com).

Conghao Zhou is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, N2L 3G1, Canada (e-mail: c89zhou@uwaterloo.ca).

Wei Quan is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: weiquan@bjtu.edu.cn).

6G's "service-oriented" approach represents a paradigm shift in mobile communication. In 6G, our focus is on achieving a fine service granularity, guaranteeing every user's personalized QoE through adaptive end-to-end service provisioning algorithms tailored to different application scenarios and network conditions.

The challenge lies in ensuring tailored and everyone-centric services to diverse user groups with varying needs, both ubiquitously and immersively. Users require integrated services that encompass multi-dimensional resources like storage, data rate, security, reliability, and knowledge, and the supports of complex combinations of virtual network functions or microservices. However, the traditional best-effort service provisioning does not have a specific network performance metrics specification for multi-dimensional services, and struggles to adapt to the diverse and dynamic needs of future user applications [14]. Recent studies have demonstrated the benefit of AI in network management, where AI is embedded into the software-defined networking (SDN) controller to promote the resource utilization [15], [16]. By distributing the AI to the user side, network manager can collect and extract user preference information, and provide proactive service optimization. Besides, AI models—such as convolutional neural networks (CNNs) for image processing, generative adversarial networks (GANs) for image generation, and Transformer models for natural language processing tasks—once trained, encapsulate a vast amount of knowledge and rules within their parameters. These models can be serialized, stored, compressed [17], [18], and later reloaded for inference tasks without requiring retraining. This capability allows them to function as self-contained modules or services, a concept often referred to as AI as a Service (AIaaS) or AI service provision [19]. By integrating AI into network management and AIaaS, it can become an integral part of the network architecture, known as native intelligence or an intelligence-native network [20], [21]. This fusion allows the network to dynamically adapt to user needs in real time, creating a highly responsive and intuitive digital ecosystem. Such networks have the potential to transform traditional network infrastructures, enabling continuous service evolution and delivering more personalized, efficient, and intelligent digital experiences.

However, existing research on AI service provision focuses on deploying services at the network edge [13], [22]–[27], while studies on multi-layer heterogeneous networks primarily concentrate on network management or traffic offloading for Internet of things (IoT) devices [8], [28]–[37]. To enable the intelligence within communication networks and realize everyone-centric service mechanisms, the study on AI service provision in multi-layer heterogeneous networks is urgently needed. For the excessive cost of computing units, how to design the network architecture and distribute the limited resources efficiently is fundamental. Then, since user distribution and their service requirements are time-varying and differentiated, how to schedule the expensive computation resources ably is the second problem to be solved. Further, existing studies on AI service provision consider AI service as an exclusive service block and ignore the intrinsic knowledge in trained models, which incurs huge resource waste in computation and caching. Therefore, the third problem lies in promoting resource utilization in the provision process. Recent works on semantic communication [38], [39] and computer vision [40] have illustrated the potentiality of AI model stitching for knowledge migration and model multiplexing, which utilizes the same portion of AI models for users with different service requirements to reduce the training process and computing resource occupation. Nevertheless, inevitable consideration on stitching decision is attached, for the index of it affects the service performance, inference delay, and resource cost, which exacerbates the complexity of the problem.

In this paper, we explores the efficient and flexible deployment of AI services within intelligence-native networks. Firstly, a novel multi-layer heterogeneous network architecture based on native intelligence is proposed. Considering the limitations of network resources and diverse user requirements, the joint AI provisioning and access selection problem is formulated as a mixed-integer nonlinear programming (MINLP) problem, with the objective of maximizing user satisfaction and service acceptance. To minimize computing resource consumption and enhance service efficiency, model stitching is introduced to enable customization and reuse of AI models for users with varied needs. Subsequently, a Dung Beetle-inspired algorithm is developed to jointly optimize AI model selection, service deployment, user access, and stitching coefficients. Finally, extensive simulations are conducted to evaluate the performance of the proposed algorithm in terms of convergence, user satisfaction value, user satisfaction ratio, and resource consumption. The main contributions of this paper are summarized as follows.

1) We proposed an intelligence-native architecture to perceive and recognize the user requirements intelligently. The proposed architecture leverages and utilizes various system resources such as sensing, computing, communication, to facilitate diverse AI methods and tailored services to meet individual needs.
2) Based on the proposed architecture, the stitching technique is introduced to provide a more flexible and fine-grained AI model provision than existing studies. Considering the limitations in computing resources and user requirements on service type, delay, and service performance, a Dung Beetle-based algorithm is presented to optimize the AI model selection, AI service deployment, user access, and stitching index, jointly.
3) Extensive simulation results are exhibited to evaluate the proposed algorithm in convergence, user satisfaction value, user satisfaction ratio, and resource consumption. Then, we execute our algorithm over a conventional AI service module (without the stitching mechanism) to evaluate the resource consumption of the introduced stitching mechanism. It is demonstrated that the introduced stitching technique achieves a higher user satisfaction value and ratio, while reducing the average resource consumption by up to 20%.

The remainder of this paper is organized as follows. The related work is introduced in Section II. In Section III, the proposed 6G intelligence-native network architecture is

introduced. In Section IV, we introduce the system model. The AI service provision and user access problem in multi-layer network is formulated in Section V. Then, a Dung Beetle-based approach is proposed to solve the formulated problem in VI. We evaluate the performance of the proposed algorithm through extensive simulations in Section VII. Section VIII concludes this paper.

II. Related Work

Recently, several works on AI service deployment and multi-layer network architecture have been conducted.

In [13], a preliminary insight of network AI architecture with integrated network resources and AI capacities is introduced. [22] investigates AI service provisioning using network slicing. Then, the AI service deployment and resource-pooling are optimized to balance AI service performance with resource consumption. [23]–[27] study the AI service deployment in IoT. To mitigate the imbalance between each IoT user, the user allocation modules and virtualization configurations are optimized to maximize the flow rate of each service [26]. Similarly, the sub-task deployment and computing resources among multiple mobile devices are scheduled in [24]. To minimize the end-to-end delay, an MINLP problem is formulated, which is solved by an Markov approximation-based approach. In industrial IoT (IIoT) scenarios, the deployment of AI services is crucial for enhancing operational efficiency and decision-making by enabling real-time data processing, predictive maintenance, and optimized resource management across resource-poor industrial devices. To minimize the average service delay of IIoT devices, the requirements on service performance, sampling rate adaptation, inference task offloading, and edge computing resource allocation are jointly considered in [23]. Similarly, a time-slotted system incorporating AI service deployment, computing resource allocation, and wireless channel allocation is presented in [25], and a deep learning-based algorithm is proposed to minimize total processing delay and error inference penalty. Examining a network that mimics human neural and cognitive functions, [27] explores the AI service deployment and resource allocation within a human-like networking architecture, where the communication, computation, and memory resources are optimized to minimize the average end-to-end delay. Current research on AI service provision is in its early stages, and the network architectures considered are limited to single-layer simple networks. As future network structures become increasingly complex, existing methods and architectures will face challenges in resource scheduling and service provision.

The space-air-ground integrated network (SAGIN) is a comprehensive communication framework that seamlessly integrates satellites, aerial platforms, and terrestrial networks to provide ubiquitous global coverage and enhanced connectivity [28], [29]. This multi-layered network architecture supports a wide range of applications, from remote sensing and broadband communications to disaster management and defense operations, ensuring reliable data transmission across diverse and often challenging environments. To cope with the mobility of vehicles and coverage deficiency of terrestrial networks, an SAGIN-assisted vehicular network is proposed in [30] to utilize the flexibility of aerial networks and coverage ability of satellite networks to provide ubiquitous network access, and a federated RL-based traffic offloading mechanism is provided to evaluate the proposed system. Focused on establishing trustworthy and privacy-preserving vehicular networks within a 6G framework, [31] analyzes the requirements and challenges in ensuring data security and user privacy. Then, a trustworthy and privacy-preserving network architecture is proposed to provide rapid decision-making, reputation feedback mechanisms, traceability mechanisms, and Sybil attack resistance ability. In our previous works [32], the SDN and network functions virtualization (NFV) are utilized to support multi-dimensional resource scheduling in a large-scale dynamic network environment. Similarly, based on the SDN, a distributed architecture is proposed in [8] and the multi-controller deployment problem is studied.

Propelled by the widespread connectivity and extensive coverage offered by SAGINs, there has been a significant transformation in resource provision by MEC. This shift has moved from traditional terrestrial edge computing to aerial or orbital edge computing, particularly in remote regions [33]. In [34], a hybrid cloud and MEC scenario based on SAGIN is considered to schedule the data offloading, where user pairing and partial offloading are optimized to minimize the energy consumption and average latency. Similarly problem is studied in [35], where civil aircraft are introduced to enhance the current satellite-terrestrial networks in terms of coverage, capacity, and task scheduling. Aiming to minimize the delay and energy consumption, a generalized benders decomposition method is proposed to optimize the access strategy, transmit power, computing resources, and delay tolerance. Orbital edge computing (OEC) extends the principles of edge computing to satellites and other space-based assets, playing a critical role within SAGINs by processing data directly at the orbit level. In [33] and [36], two types of OEC architecture are presented. In [33], a hierarchical satellite network composed of both GEO and LEO satellite layers is designed to relay the data to a remote data center. On the contrary, a collaborative mechanism is utilized within the LEO satellite layer to distribute the data to nearby satellites and reduce the processing delay [36]. Although these studies considered the network management or traffic offloading for IoT devices, the native-intelligence is not included in their contributions.

III. 6G Network Architecture Based on Native Intelligence

To address diverse user demands, we propose a 6G network architecture centered on native intelligence, named as 6G Network AI. This architecture utilizes high-speed, low-latency transmission channels to integrate distributed communication resources across a multi-layer heterogeneous network for AI model deployment. Users can directly access and utilize these AI models through the network, leading to enhanced service quality, greater intelligence, and more personalized user experiences.

To achieve comprehensive intelligent management and precise on-demand services in future networks, the network ar-

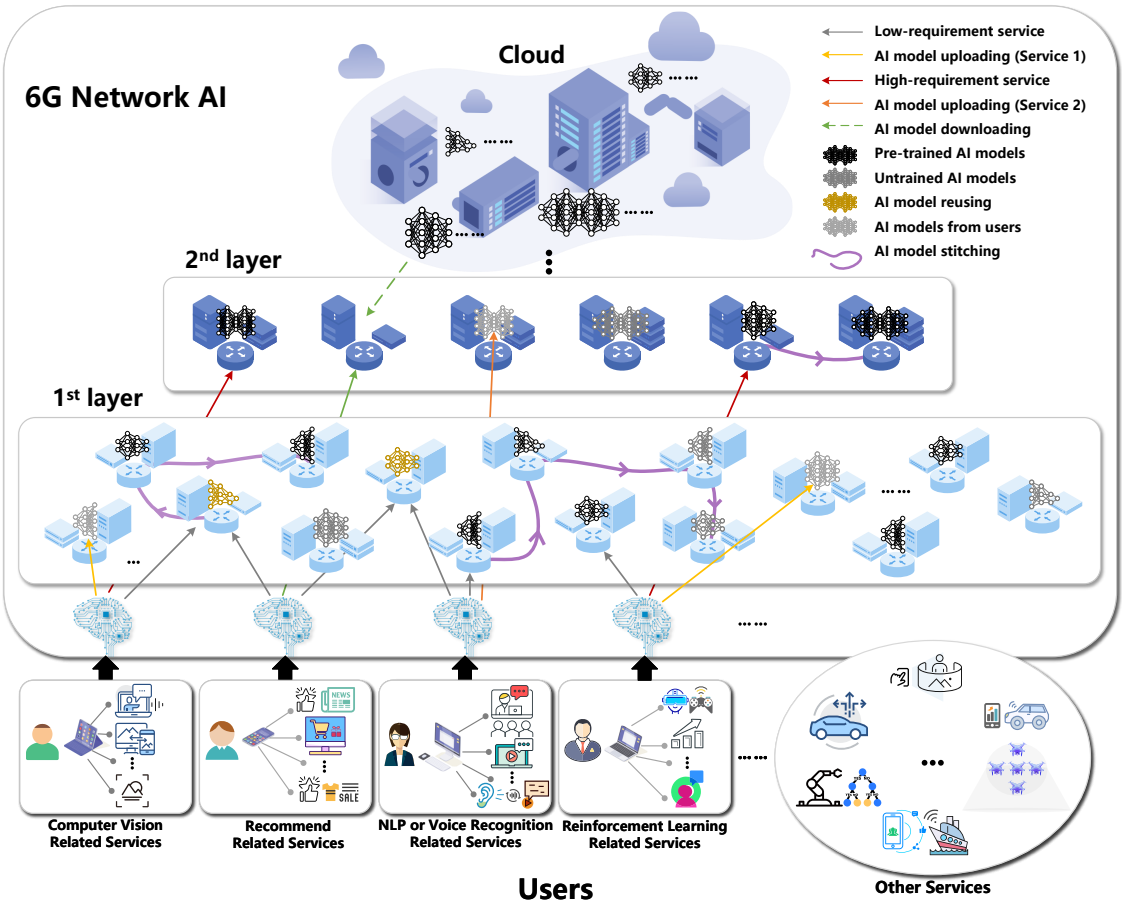


Fig. 1: 6G network architecture based on native intelligence.

chitecture must meet the dual requirements of scenario awareness and service customization. Accordingly, the network architecture proposed comprises three intercooperative layers: the cognitive layer, the decision layer, and the intelligent layer. The primary responsibility of the cognitive layer is to interpret user intent and network capabilities, construct an all-scenario information ontology, and accurately identify the user's context and needs during the service process. The decision layer, which is based on scenario identification results from the cognitive layer, perceives resources and allocates them as needed. The intelligent layer is the core of the entire intelligence-native network architecture. Based on the constructed AI model library, knowledge base, and intent library, it stores and applies knowledge accumulated during the cognitive and decision processes to enhance decision accuracy and execution efficiency. Additionally, it delivers ubiquitous AI services directly to users through network nodes, thereby significantly reducing transmission delay and promoting the utilization of dispersed computing resources.

The 6G Network AI, as shown in Fig. 1, is a multi-layer network AI architecture comprising various 6G network element nodes. By integrating and coordinating cross-domain sensing, storage, communication, computing, control, and AI resources, this architecture processes local and regional user data and executes distributed AI models to provide personalized services for each user.

On the user side, when an AI service request from the user device arrives at the network, the cognitive layer begins to function. It employs a scenario knowledge base and AI technologies, such as natural language processing, to accurately identify and classify the user's needs and perform preliminary processing. The decision layer then receives the processed scenario and requirement information. At this layer, the system designs a detailed on-demand service plan based on the task type. This plan includes formulating service strategies, drawing from the knowledge base and AI model library, and optimizing service deployment according to the current network resource status and network elements' computing capabilities.

The intelligent layer activates once the decision layer completes the user service configuration. This layer is responsible for deploying and invoking AI models to execute specific tasks and feedback the results to the user, thus achieving on-demand serving. The deployment and invocation strategies for AI models are diverse: they can involve the direct invocation of a single AI model, the reuse and combination of multiple AI models, or the uploading and installing of user-customized models. Invoking a single AI model entails selecting the most suitable model from those already deployed for the current task. Reusing AI models improves model utilization, reduces redundant resource consumption, and lowers operational costs. The combination of multiple AI models integrates

the strengths of each model, resulting in more precise and comprehensive service outcomes. Allowing users to upload their AI models further enhances the personalization and flexibility of network services, enabling them to deploy their custom-developed AI models into the network for use in specific scenarios.

The intelligence-native network architecture designed in this paper centers around the intelligent layer, achieving ubiquitous and personalized services through the close coupling of modular AI models with network nodes. This architecture significantly enhances the serving capabilities and efficiency of the network while improving service adaptability and customization levels through the flexible invocation and combination of different AI models. Moreover, it allows users to upload custom AI models to meet more personalized service needs, providing a scalable, flexible, and efficient network solution. This enables the 6G network to offer more intelligent and personalized service experiences, improving resource utilization efficiency and user satisfaction.

IV. SYSTEM MODEL

Based on the proposed 6G Network AI, this section constructs an on-demand service system with stitchable AI model. The symbols used in this paper are listed in Table I.

TABLE I: NOTATIONS

Notations	Description
$\mathcal{Q}$	Set of user service requests.
$\mathcal{M}$	Collection of AI models.
N	Number of AI models in the model library.
$\boldsymbol{\beta}$	Collection of stitching index for all users.
$\boldsymbol{L}$	Deployment location matrix of AI models.
$\boldsymbol{X}$	Position matrix of dung beetle population.
$\boldsymbol{Y}$	Collection the service reception index for all users.
R_i	The transmission rate of i-th layer network.
C_i	The computing capacity of i-th layer network.
$\boldsymbol{\beta}_q^p$	Stitching ratio vector of preceding model for user request q.
$\boldsymbol{\beta}_q^s$	Stitching ratio vector of succeeding model for user request q.
$\boldsymbol{\beta}_q^{ns}$	Index vector of model without stitching for user request q.
Z_q	Data volume of user request q.
T_q	Delay requirement of user request q.
A_q	Required model performance of user request q.
L_q^p	Deployment location vector of the preceding model of user request q.
L_q^s	Deployment location vector of the succeeding model of user request q.
L_q^{ns}	Deployment location vector of model without stitching of user request q.
t_q	Actual end-to-end delay of user request q.
c_q^{tr}	Total transmission resource consumption of user request q.
y_q	The reception index of user request q.

A. Stitching Service Model

Consider a network with I layers, and each layer of the network contains network nodes deployed with computing units, where the number of nodes, transmission rate, and computing capacity of layer i are denoted by N_i, R_i, and C_i, respectively. The top network layer, i.e., the I-th layer, can be likened to the cloud layer with abundant computing resources. As the layer descends, the nodes become more numerous, and their computing resources are more dispersed. The lowest level, i.e., the first layer, can be regarded as the user layer, where the service requests initiate. Let $\mathcal{Q}$ be the set of user service requests, defined as $\mathcal{Q} = \{q|q = 1, 2, \ldots, |\mathcal{Q}|\}$, where $|\mathcal{Q}|$ is the number of services. Each service q is associated with a triple (Z_q, T_q, A_q), where Z_q denotes the desired data volume to be processed, T_q denotes the maximum tolerable delay, and A_q is the desired model performance. Without loss of generality, we consider one type of application in each decision making process, and the type of service q is denoted by M_q. AI model library is denoted by $\mathcal{M} = \{M_1, M_2, \ldots, M_N\}$, where N is the number of models.

Generally, the AI model in the model base is designed and trained in advance by statistical requirements on inference delay, inference accuracy, etc. When the service requests from users arrive, the network operator is able to choose and install the AI model with sufficient performance to fulfill the requirements. However, as user demands become more strict and complex, the limited number of models and traditional reliance on single-metric model selection criteria will increasingly prove inadequate. Nonetheless, blindly increasing the number of models with diverse models is not cost-effective.

In this paper, we utilize the model stitching technique to stitch available AI models in model base for users with non-default requirements and reduce the performance redundancy by replacing unnecessary models with high performance. Based on this, multiple users who require the same part of the model can reuse the installed model, thereby reducing resource consumption. Without losing generality, we consider the AI model is able to be dismantled and stitched by two separative parts of models in the same library [40], [41]. Intuitively, from the inference sequence, the front part of the model is defined as the preceding model, and the remaining part is defined as the succeeding model.

The selected models for service q are defined as m_p^s and m_r^s, where m_p^s represents the front model and m_r^s denotes the rear model in stitching. For service q, a one-dimension vector $\boldsymbol{\beta}_q^p = [\beta_{1,q}^p, \beta_{2,q}^p, \cdots, \beta_{N,q}^p]^T$ is introduced to represent the stitching ratio of the preceding model, where $\beta_{n,q}^p \in [0, 1)$ denotes the ratio of model n served as the preceding model. If $0 < \beta_{n,q}^p < 1$, it denotes that model n is utilized as the preceding model, and the stitching ratio is $\beta_{n,q}^p$; and if $\beta_{n,q}^p = 0$, it denotes that model n is not utilized for service q. At the same time, only one element in $\boldsymbol{\beta}_q^p$ is positive, and the others equal 0. The collection of it is denoted by matrix $\boldsymbol{\beta}^p = [\boldsymbol{\beta}_1^p, \boldsymbol{\beta}_2^p, ..., \boldsymbol{\beta}_{|\mathcal{Q}|}^p]$. Similarly, $\boldsymbol{\beta}_q^s = [\beta_{1,q}^s, \beta_{2,q}^s, \cdots, \beta_{N,q}^s]^T$ denotes the succeeding model, and the collection of it is denoted by matrix $\boldsymbol{\beta}^s = [\boldsymbol{\beta}_1^s, \boldsymbol{\beta}_2^s, ..., \boldsymbol{\beta}_{|\mathcal{Q}|}^s]$. Besides, an one-hot vector $\boldsymbol{\beta}_q^{ns} = [\beta_{1,q}^{ns}, \beta_{2,q}^{ns}, \cdots, \beta_{N,q}^{ns}]^T$ is introduced to denote the situation without stitching, where $\beta_{n,q}^p = 1$, it denotes the model n is utilized directly for user q without stitching. The collection of it is denoted by matrix $\boldsymbol{\beta}^{ns} = [\boldsymbol{\beta}_1^{ns}, \boldsymbol{\beta}_2^{ns}, ..., \boldsymbol{\beta}_{|\mathcal{Q}|}^{ns}]$. Then, the inference performance of q is defined as $a_q = f(\boldsymbol{\beta}_q^p, \boldsymbol{\beta}_q^s, \boldsymbol{\beta}_q^{ns})$. The collection of the stitching index is denoted by $\boldsymbol{\beta} = \{\boldsymbol{\beta}^p, \boldsymbol{\beta}^s, \boldsymbol{\beta}^{ns}\}$.

B. Service Deployment Model

In stitchable neural network technology, we typically use a small-scale model as the preceding part and a large-scale model as the succeeding part, known as the fast-to-slow stitching direction. Each part of the model can be deployed on different locations or the same location. Considering the network resource distribution and user requirements, user can select appropriate network layer to access and offload the data to be processed. If the preceding and succeeding models are deployed on different layers, intermediate output will be transmitted to the layer with succeeding model. The final result transmission after processing can often be neglected due to its minimal computational load and smaller data size.

For user request q, a one-hot vector $\mathbf{L}_q^p = [l_{1,q}^p, l_{2,q}^p, \cdots, l_{I,|\mathcal{Q}|}^p]^T$ represents the deployment location of the preceding part of the model, where the binary variable $l_{i,q}^p = 1$ indicates the preceding model is installed on the i-th layer of network, and $l_{i,q}^p = 0$ otherwise. $\mathbf{L}_q^s = [l_{1,q}^s, l_{2,q}^s, \cdots, l_{I,q}^s]^T$ represents the deployment location of the succeeding part, where $l_{1,q}^s = 1$ indicates the succeeding model is deployed on the i-th layer of the network, and $l_{i,q}^s = 0$ otherwise. $\mathbf{L}_q^{ns} = [l_{1,q}^{ns}, l_{2,q}^{ns}, \cdots, l_{I,q}^{ns}]^T$ refers to the deployment location of the AI model without stitching, where $l_{1,q}^{ns} = 1$ indicates the stand-alone model for service q is installed on the i-th layer of network, and $l_{i,q}^{ns} = 0$ otherwise. If a stand-alone model for service q is deployed, it will not need the preceding model and succeeding model, which means that all the elements in $\mathbf{L}_q^p$, $\mathbf{L}_q^s$, $\boldsymbol{\beta}_q^p$, and $\boldsymbol{\beta}_q^s$ equal 0.

The location matrix collection $\mathbf{L}$ is defined as

$$\mathbf{L} = \{\mathbf{L}^p, \mathbf{L}^s, \mathbf{L}^{ns}\}, \tag{1}$$

where $\mathbf{L}^p = [\mathbf{L}_1^p, \mathbf{L}_2^p, \cdots, \mathbf{L}_{|\mathcal{Q}|}^p]$ represents the deployment position matrix of the preceding stitching model, $\mathbf{L}^s = [\mathbf{L}_1^s, \mathbf{L}_2^s, \cdots, \mathbf{L}_{|\mathcal{Q}|}^s]$ represents the deployment position matrix of the succeeding stitching model, and $\mathbf{L}^{ns} = [\mathbf{L}_1^{ns}, \mathbf{L}_2^{ns}, \cdots, \mathbf{L}_{|\mathcal{Q}|}^{ns}]$ represents the deployment position matrix of a single AI model. After stitching the model, these position matrices jointly determine the deployment location. Denote binary variable $y_q = 1$ to signify the reception of service q, and $y_q = 0$ otherwise.

C. Delay Model

Data transmission and model execution dominate the delay for service q, which contains a certain amount of data to transmit and process. We express the total delay of service request q, which spans from task initiation to completion, as

$$t_q = t_q^{tr} + t_q^{ex}, \tag{2}$$

where t_q^{tr} is the delay of data transmission from users to the installed AI model, and t_q^{ex} is the inference delay.

The transmission delay is expressed as

$$t_q^{tr} = \sum_{i=1}^{I} \frac{Z_q}{R_i} l_{i,q}^{ns} + \sum_{i=1}^{I} \frac{Z_q}{R_i} l_{i,q}^{p} + \sum_{i=i_p}^{I} \frac{Z_q^{out}}{R_i} l_{i,q}^{s}, \tag{3}$$

where Z_q^{out} is the data volume of service q output from the preceding model, and i_p denotes the layer index of the preceding model. The amount of output data of the task data after being processed by the pre-model is influenced by both the stitching coefficients and AI model. Denote function $\varphi\left(\beta_{n,q}^p\right)$ to represent the relation between stitching coefficient and volume of output data for service q. The output data amount Z_q^{pout} of task q is expressed as

$$Z_q^{out} = \varphi\left(\beta_{n,q}^p\right) \cdot Z_q. \tag{4}$$

D. Resource Consumption Model

In certain scenarios, multiple user requests may arrive simultaneously, necessitating the execution of stitching operations on the same layer of the same model to accomplish various tasks. Alternatively, a single AI model may need to handle similar tasks for different users. Therefore, this paper considers the reuse of partially stitched models to reduce resource consumption and improve service efficiency. The indicator function $\gamma(\cdot)$ is introduced to denote the deployment of AI model, and $M(\boldsymbol{\beta}_n^p)$ to denote the deployment of model n with stitching coefficient $\beta_{n,q}^p$. For example, $\gamma(i, M(\boldsymbol{\beta}_n^p))$ denotes whether the preceding part of model n with stitching coefficient $\boldsymbol{\beta}_n^p$ is deployed on layer i, where $\gamma(i, M(\boldsymbol{\beta}_n^p)) = 1$ indicates that as one model is deployed, and $\gamma(i, M(\boldsymbol{\beta}_n^p)) = 0$ otherwise.

Deploying AI models to network nodes requires significant computational resources of CPUs or GPUs, which is influenced by the model type and stitching ratio. As the computation capacity is limited, it is crucial to consider the resource consumption in service deployment. The computation resource consumption of the preceding part of deployed model is expressed as

$$c^p(i, \boldsymbol{\beta}_n^p) = \gamma(i, M(\boldsymbol{\beta}_n^p))\, \sigma(\boldsymbol{\beta}_n^p)\, C_n, \tag{5}$$

where C_n is the computing resource consumption of the module n. $\sigma(\cdot)$ denotes the ratio of parameter amount of the preceding part to that of the original model. Similarly, the resource consumption of the succeeding part is expressed as

$$c^s(i, \boldsymbol{\beta}_n^s) = \gamma(i, M(\boldsymbol{\beta}_n^s))\, \varsigma(\boldsymbol{\beta}_n^s)\, C_n, \tag{6}$$

where $\varsigma(\cdot)$ denotes the occupied computing resources of the succeeding part. The expression for the module's resource consumption without stitching is

$$c^{ns}(i, \boldsymbol{\beta}_n^{ns}) = \gamma(i, M(\boldsymbol{\beta}_n^{ns}))\, C_n. \tag{7}$$

Since the model may deploy far from user side, the transmission cost is indispensable, which is expressed as

$$c_q^{tr} = \sum_{i=1}^{I} \frac{Z_q}{R_i} l_{i,q}^{ns} k_i + \sum_{i=1}^{I} \frac{Z_q}{R_i} l_{i,q}^{p} k_i + \sum_{i=i_p}^{I} \frac{Z_q^{out}}{R_i} l_{i,q}^{s} k_i, \tag{8}$$

where k_i denotes the average transmission power of layer i.

V. PROBLEM FORMULATION

In this section, we investigate the stitchable AI service deployment in the multi-layer heterogeneous network. An MINLP problem is formulated to maximize the total system profit under the constraints on service delay, inference performance, and computing resources.

A. Service Provision Constraints

The system must meet the delay requirements for each user request, ensuring timely responses to maintain user satisfaction. Additionally, the system must fulfill the requirements on model performance, ensuring the AI model delivers accurate and reliable results as expected by the user. The constraints on service provision are expressed as

$$C_1 : t_q \leq T_q, \forall q \in \mathcal{Q}; \tag{9}$$

$$C_2 : a_q \leq A_q, \forall q \in \mathcal{Q}. \tag{10}$$

B. Capacity Constraints

For each layer of the network, the computation resource is limited, and the occupied resources of each module cannot exceed the capacity, which is expressed as

$$C_3 : \sum_{n\in[1,N]} c^p(i,\boldsymbol{\beta}_n^p) + \sum_{n\in[1,N]} c^s(i,\boldsymbol{\beta}_n^s) + \sum_{n\in[1,N]} c^{ns}(i,\boldsymbol{\beta}_n^p) \leq C_i, \forall i \in \{1,2,...,I\}. \tag{11}$$

C. Service Deployment Constraints

For each service request q, the deployment layer of the front model must be earlier than or equal to the post model, which ensures the logical order of AI model and the correctness of data flow, which is expressed as

$$C_4 : \sum_{i=[1,i_1]} l_{i,q}^s \leq \sum_{i=[1,i_1]} l_{i,q}^p, \forall q \in \mathcal{Q}, \forall i_1 \in \{1,2,...,I\}. \tag{12}$$

Constraints C_5 to C_6 must be satisfied in order to guarantee that the AI model of receipted service must be installed, which is express as

$$C_5 : y_q \leq \sum_{i=[1,I]} l_{i,q}^p + \sum_{i=[1,I]} l_{i,q}^{ns}, \forall q \in \mathcal{Q}; \tag{13}$$

$$C_6 : y_q \leq \sum_{i=[1,I]} l_{i,q}^s + \sum_{i=[1,I]} l_{i,q}^{ns}, \forall q \in \mathcal{Q}, \tag{14}$$

where binary variable $y_q = 1$ signifies the reception of service request q, otherwise $y_q = 0$. The collection service reception index is represented as $\mathbf{Y}$.

For each service request q, the provided model must be complete to guarantee correct inference process. This means that the sum of the stitching coefficients must equal 1, or the deployed model for q is not stitched, which is expressed as

$$C_7 : \|\boldsymbol{\beta}_q^p\| + \|\boldsymbol{\beta}_q^s\| + \|\boldsymbol{\beta}_q^{ns}\| = 1 \ , \forall q \in \mathcal{Q}. \tag{15}$$

D. MINLP Problem

In the paper, the stitching coefficient and model deployment are optimized to maximize the total system profit P. Combined with constraints $C_1 - C_7$, the joint AI provision and user access selection problem is formulated as

$$\begin{aligned} P1 : \max_{\mathbf{L},\mathbf{Y},\boldsymbol{\beta}} \quad & P = R - C \\ s.t. \quad & C_1 - C_7, \\ & C_8 : \mathbf{L} \in \{0,1,...,I\}, \\ & C_9 : \boldsymbol{\beta} \in [0,1]. \end{aligned} \tag{16}$$

where R denotes the user satisfaction value of all service requests and C represents the sum of cost in resource consumption. The satisfaction value is expressed as

$$R = \sum_{q\in\mathcal{Q}} y_q(1 + \rho_q^t + \rho_q^a)R_q + \frac{\sum_{q\in\mathcal{Q}} y_q}{|\mathcal{Q}|}R', \tag{17}$$

where R_q is the inherent revenue when service q is fulfilled, and R' is revenue for service reception. ρ_q^t and ρ_q^a measure the fulfillment of the requirements on delay and inference performance [42], i.e.,

$$\rho_q^t = \begin{cases} \frac{T_q - t_q}{T_q}, & \text{if (9) is fulfilled,} \\ 0, & \text{otherwise;} \end{cases} \tag{18}$$

$$\rho_q^a = \begin{cases} \min(\frac{a_q - A_q}{A_q}, 2), & \text{if (10) is fulfilled,} \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

We formulate this mathematical problem to optimize the provision of AI models in a multi-layer heterogeneous network to maximize total system profit while meeting service requirements and performance constraints. This paper considers stitching AI models from the library to meet specific user service requests, reducing redundancy and enhancing resource efficiency. By incorporating stitching coefficients and deployment location matrices, this model satisfies the constraints on delay, performance, and capacity, ultimately balancing the utilization of computational resources and achieving high user satisfaction with minimized costs. For the binary and continuous variables and non-linear constraints, the proposed problem can be categorized into an MINLP problem.

VI. DBSSO Algorithm

The problem formulated in this paper involves optimizing multiple integer and continuous variables under constraints, posing significant challenges for conventional approaches like deep reinforcement learning. The Dung Beetle algorithm, by effectively balancing exploration and exploitation within the search space, excels in identifying optimal solutions to complex optimization problems. In this paper, we propose the Dung-Beetle-based Stitching Coefficient and Service Access Optimization (DBSSO) algorithm [43]. In this algorithm, the stitching coefficients and deployment positions are encoded as genes of the individual dung beetles, forming the population matrix. The objective function is transformed into a fitness function, allowing the dung beetle population to evolve based on their adaptability. Our goal is to find the individual with the maximum fitness value in the dung beetle population, representing the optimal service strategy. The optimization matrix of the proposed algorithm in the t-th iteration is expressed as

$$\mathbf{X}^t = \begin{bmatrix} \mathbf{x}_1^t \\ \mathbf{x}_2^t \\ \vdots \\ \mathbf{x}_V^t \end{bmatrix} = \begin{bmatrix} x_{1,1}^t & x_{1,2}^t & \cdots & x_{1,D}^t \\ x_{2,1}^t & x_{2,2}^t & \cdots & x_{2,D}^t \\ \vdots & \vdots & \ddots & \vdots \\ x_{V,1}^t & x_{V,2}^t & \cdots & x_{V,D}^t \end{bmatrix}, \tag{20}$$

where D is the gene dimension of the individual dung beetle, and V represents the number of dung beetles in the population.

$\mathbf{x}_v^t$ denotes the position of v-th dung beetle in the t-th iteration, which is a four-dimension vector that combines the stitching coefficients, deployment positions, and reception index.

To enhance the convergence while satifying the constraints, we introduce the fitness function $f(\cdot)$ to evaluate the position of each dung beetle. The fitness function for this algorithm, based on the objective function in the previous section, introduces the penalty for constraint violation, which is defined as

$$f(\mathbf{x}_v^t) = \mathcal{P} - \sum_{q \in \mathcal{Q}} y_q(\rho_q^\beta \Psi_1 + \rho_q^l \Psi_2), \tag{21}$$

where Ψ_1 and Ψ_2 are the penalty values for the violation of stitching index and model deployment, respectively. ρ_q^β and ρ_q^l are the coefficients to measure the violation degree, which is expressed as

$$\rho_q^\beta = [1 - \min(1, (\|\boldsymbol{\beta}_q^p\| + \|\boldsymbol{\beta}_q^s\| + \|\boldsymbol{\beta}_q^{ns}\|))]; \tag{22}$$

$$\rho_q^l = \begin{cases} 1, & \text{if (12) is fulfilled,} \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

There are four types of dung beetles in the population: male dung beetles with a number of V_1, breeding dung beetles with a number of V_2, foraging dung beetles with a number of V_3, and thief dung beetles with a number of V_4. Although each different type of dung beetle performs distinct behaviors, they work together as a collaborative whole to intelligently search for solutions in complex spaces. In the following, we will introduce these types of dung beetles individually.

1) Male dung beetle: Male dung beetles engage in two behaviors, rolling balls and dancing, to conduct searches in local areas. In the search space, each male dung beetle moves in a specific direction to roll balls. During the rolling process, the dung beetle updates its position to account for the impact of light source intensity on its path as

$$\mathbf{x}_i^{t+1} = \mathbf{x}_v^t + \vartheta \varepsilon \mathbf{x}_v^{t-1} + b\Delta\mathbf{x}, \tag{24}$$

where ϑ is the natural coefficient assigned -1 or 1, $\varepsilon \in (0, 0.2]$ represents the deflection coefficient, and $b \in (0, 1)$ is a constant to denote the influence of light fluctuation. When the value of the natural coefficient is set to 1, it means that it does not deviate from the original direction; otherwise, the value is set to -1. To simulate the uncertainty of the real environment, this paper uses the probability selection method of λ_1 to select the natural coefficients. To expand the search range of the proposed algorithm, $\Delta\mathbf{x}$ is introduced to represent the change of simulated light intensity, which is expressed as

$$\Delta\mathbf{x} = \left|\mathbf{x}_v^t - \mathbf{x}^{\text{worst}}\right|, \tag{25}$$

where $\mathbf{x}^{worst}$ is the position of the dung beetle with the global worst fitness by (21).

When dung beetles encounter obstacles that prevent them from rolling in a straight line, they dance and reposition themselves in order to find a new rolling path. We use the tangent function to update the position of the dung beetle. The definition of the dancing behavior's position update is

$$\mathbf{x}_v^{t+1} = \mathbf{x}_v^t + \tan(\varpi)(\mathbf{x}_v^t - \mathbf{x}_v^{t-1}), \tag{26}$$

where ϖ is the deflection angle uniformly selected from $[0, \pi]$, and when ϖ equals 0, $\frac{\pi}{2}$ or π, the position will not be updated. After determining its new direction, the dung beetle resumes its rolling ball behavior, with its current location and historical location information closely influencing its location update.

2) Breeding dung beetle: To ensure the safe growth of their offspring, the breeding dung beetles, also known as female dung beetles, roll the dung balls to a safe place and burrow them. Choosing a suitable egg-laying site is a key step in the dung beetle spawning process. We propose a boundary selection strategy to mimic the oviposition area of female dung beetles. The mathematical expression of this boundary selection strategy is defined as

$$Lb^* = \max\left((1 - R)\mathbf{x}^*, Lb\right); \tag{27}$$

$$Ub^* = \min\left((1 + R)\mathbf{x}^*, Ub\right), \tag{28}$$

where $\mathbf{x}^*$ denotes the current local optimal position, Lb^* and Ub^* represent the lower and upper bounds of the spawning area, respectively. $R = 1 - \frac{t}{T_{\max}}$ denotes adjustment factor that changes dynamically as the number of iterations increases, and $T_{\max}$ is the threshold of iterations. Lb and Ub represent the lower and upper bounds of the search space, respectively.

After determining the spawning area, the female dung beetle will choose a location to lay eggs, and each female dung beetle produces one egg ball per iteration. The boundaries of the spawning area are dynamic and mainly determined by R. Accordingly, the position update of the brood ball is also dynamic, which is defined as

$$\mathbf{x}_v^{t+1} = \mathbf{x}^* + b_1(\mathbf{x}_v^t - Lb^*) + b_2(\mathbf{x}_v^t - Ub^*), \tag{29}$$

where b_1 and b_2 are two independent random vectors of size $1 \times D$, and D represents the dimension of the optimization problem. The position update algorithm of the brood ball is presented in Algorithm 1.

Algorithm 1: The updating strategy of breeding dung beetles

Input: The maximum number of iterations $T_{\max}$, the number of brood balls V_3, and current number of iteration t

Output: The position $\mathbf{x}_v^{t+1}$ of the v-th breeding dung beetle

```
R = 1 − t/T_max;
for i ← 1 to V_3 do
    Update the position of the breeding dung beetle ball according (29);
    for d ← 1 to D do
        if x_{v,d} > Ub* then
            x_{v,d} ← Ub*;
        if x_{v,d} < Lb* then
            x_{v,d} ← Lb*;
```

3) Foraging dung beetle: Foraging beetles exhibit the behavior of searching for optimal resources by exploring the search space extensively. They utilize adaptive boundaries to guide their movement, allowing them to efficiently locate high-quality solutions. The boundaries are defined as

$$Lb^b = \max\left((1-R)\mathbf{x}^b, Lb\right); \tag{30}$$

$$Ub^b = \min\left((1+R)\mathbf{x}^b, Ub\right), \tag{31}$$

where X^b represents the global optimal position, and Lb^b and Ub^b represent the lower and upper bounds of the optimal foraging area respectively. Accordingly, the position update formula of the foraging dung beetle is

$$\mathbf{x}_v^{t+1} = \mathbf{x}_v^t + B_1(\mathbf{x}_v^t - Lb^b) + B_2(\mathbf{x}_v^t - Ub^b), \tag{32}$$

where B_1 represents a random number that follows the normal distribution, and B_2 represents a random vector that falls within the range of $(0, 1)$.

4) Thief dung beetle: There is a special type of individual called *thieves* who obtain food by stealing the dung balls of other individuals. X^b is introduced to point to the position of the optimal food source, i.e., the potential position of the optimal solution. The location update formula of the thief dung beetle is defined as

$$\mathbf{x}_v^{t+1} = X^b + \nu g\left(|\mathbf{x}_v^t - \mathbf{x}^*| + |\mathbf{x}_v^t - X^b|\right), \tag{33}$$

where g is a random vector of size $1 \times D$ that obeys a normal distribution, and ν is a constant used to adjust the step size. They dynamically adjust their positions based on their proximity to the best solutions discovered, improving the algorithm's ability to escape local optima and find the global optimum.

The pseudocodes of the proposed algorithm are shown in Algorithm 2. A time-slotted manner is utilized in this system, and the set of service requests $\{(Z_q, T_q, A_q), \forall q \in \mathcal{Q}\}$, maximum number of iterations $T_{\max}$, and the population of dung beetle population V are input first. In lines 1-2, the position of each dung beetle is randomly initialized to ensure that dung beetles with different behavioral modes can be distinguished, including male dung beetles, breeding dung beetles, foraging dung beetles, and thief dung beetles. Then, the positions of these dung beetles are updated successively. For male dung beetles, the changes in light intensity are computed by (25), firstly. Then, the position is updated by (24) in lines 7-13 with consideration of the previous position and simulated light. For breeding dung beetles, the local optimal lower and upper bounds, i.e., the spawning area, are obtained by (27) and (28). Correspondingly, each breeding dung beetle chooses a location to produce one egg ball by (29) in lines 15-16. Foraging dung beetles prefer to explore the area of optimal global positioning. Based on the global optimal position of (30) and (31), the position of them is updated by (32) in lines 18-19. For thief dung beetles, they are influenced by both the global and local boundaries, which is updated by (33) in line 21. After each dung beetle's position is updated, the position with a higher fitness value replaces the previous one. Finally, the procedure stops if the iteration exceeds the maximum value T_{max} or $\mathbf{X}^b$ remains unchanged in succession.

Algorithm 2: DBSSO algorithm

Input: The set of user requirements $\{(Z_q, T_q, A_q), \forall q \in \mathcal{Q}\}$, the maximum number of iterations $T_{\max}$, the population of dung beetle population V

Output: Stitch coefficient matrix β, model deployment matrix collection $\mathbf{L}$, reception index matrix collection $\mathbf{Y}$

1 Initialize current iteration index $t \leftarrow 1$;
2 Initialize dung beetle population $\mathbf{X}^t$;
3 **while** $t \leq T_{\max}$ **do**
4 **for** $v \leftarrow 1$ **to** V **do**
5 Set $\vartheta \leftarrow 1$;
6 **if** $v \leq V_1$ **then**
7 **if** *rand*$(1) < \lambda_0$ **then**
8 **if** ***rand***$(1) \leq \lambda_1$ **then**
9 Set $\vartheta \leftarrow -1$;
10 Update the position of male dung beetle by (24) ;
11 **else**
12 Randomly select ϖ from [0, π] ;
13 Update the position of male dung beetle by (26) ;
14 **if** $V_1 < v \leq V_1 + V_2$ **then**
15 Update the spawning area by (27) and (27) ;
16 Update the position of breeding dung beetle by (29) ;
17 **if** $V_1 + V_2 < v \leq V - V_4$ **then**
18 Update the foraging area by (30) and (31);
19 Update the position of foraging dung beetle by (32) ;
20 **if** $V - V_4 < v \leq V$ **then**
21 Update the position of thief dung beetle by (33) ;
22 **if** *the newly formed dung beetle is better than before* **then**
23 Update the location of v-th dung beetle;
24 $t = t + 1$;

VII. PERFORMANCE EVALUATION

This section demonstrates the simulations to evaluate the performance of the proposed algorithm in terms of convergence, user satisfaction ratio, total user satisfaction value, and average resource consumption. Drawing from the configuration in [40], [13], and [44], the simulations are conducted in a four-layer network, where realistic inference data of DeiT-Ti, DeiT-S and DeiTB based on ImageNet-22K pre-training is utilized. Considering the characteristics of the AI-based 6G network architecture proposed in this paper, we have established the corresponding network architecture simulation parameters, as detailed in Table II. The simulation parameter

TABLE II: SCENARIO PARAMETER SETTINGS

Parameters	Value
Transmission rate to the cloud layer R_4	$125 \times 10^9 (bytes/s)$
Number of network nodes in cloud layer N_4	1000
Computing capacity of each node in cloud layer C_4	$200\,(Flops\,(G))$
Transmission rate from the second layer to the third layer R_3	$12.5 \times 10^9 (bytes/s)$
Number of network nodes in the third layer N_3	20
Computing capacity of each node in the third layer C_3	$80\,(Flops\,(G))$
Number of network nodes in the second layer N_2	60
Transmission rate from the first layer to the second layer R_2	$6.25 \times 10^9 (bytes/s)$
Computing capacity of eahc node in the second layer C_2	$20\,(Flops\,(G))$
Number of network nodes in the first layer N_1	100
Transmission rate from users to the first layer R_1	$1.25 \times 10^9 (bytes/s)$
Computing capacity of eahc node in the first layer C_1	$5\,(Flops\,(G))$
Maximum tolerable delay T_q	$[1, 200]\ (ms)$
Desired model performance A_q	$[60\%, 82\%]$
Desired data volume Z_q	$[1 \times 10^5, 5.2 \times 10^7]\ (bytes)$

TABLE III: ALGORITHM PARAMETER SETTINGS

Algorithm parameters	Value
Maximum number of iterations $T_{\max}$	100
Total dung beetle population V	300
Number of male dung beetles V_1	60
Number of breeding dung beetles V_2	60
Number of foraging dung beetles V_3	70
Number of thief dung beetles V_4	110
Deflection coefficient of dung beetle rolling ball behavior ε	0.1
Dung beetle rolling ball behavior step coefficient b	0.3
Dung beetle stealing behavior step coefficient ν	0.5

settings take into account the trend of a decreasing number of network nodes with increasing network layers, while the computational capacity of individual network nodes correspondingly increases. Additionally, this paper assumes that the network nodes located in the cloud layer have sufficient computational capacity but suffer transmission delay. For the proposed DBSSO algorithm, the deflection coefficient ε of the dung beetle rolling ball behavior, the step coefficient b of the dung beetle rolling ball behavior, and the step coefficient ν of the dung beetle stealing behavior were determined based on the empirical values. The specific parameters of the algorithm are shown in Table III. To examine the performance of the proposed algorithm, three benchmarks are utilized: genetic algorithm (GA) [45], [46], delay optimal algorithm (DOA) [44], and random benchmark (RB) [24]. The simulation is carried out on a computer with 2.10 GHz Intel Core i7-12700F, 16GB RAM, and NVIDIA 2060Ti.

We mainly examined two scenarios: intelligence-native cloud network architecture (Cloud AI) and intelligence-native MEC architecture (MEC AI), to evaluate the performance of the proposed 6G Network AI. Among them, the parameter settings of Cloud AI are consistent with the cloud layer parameters in Table II, while the parameters of MEC AI combine the

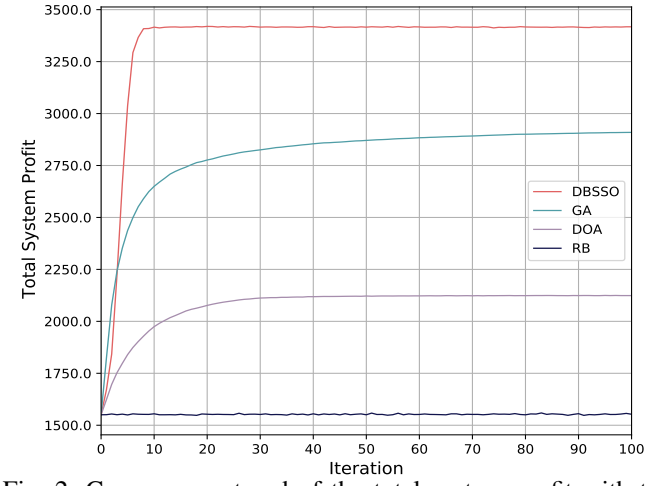
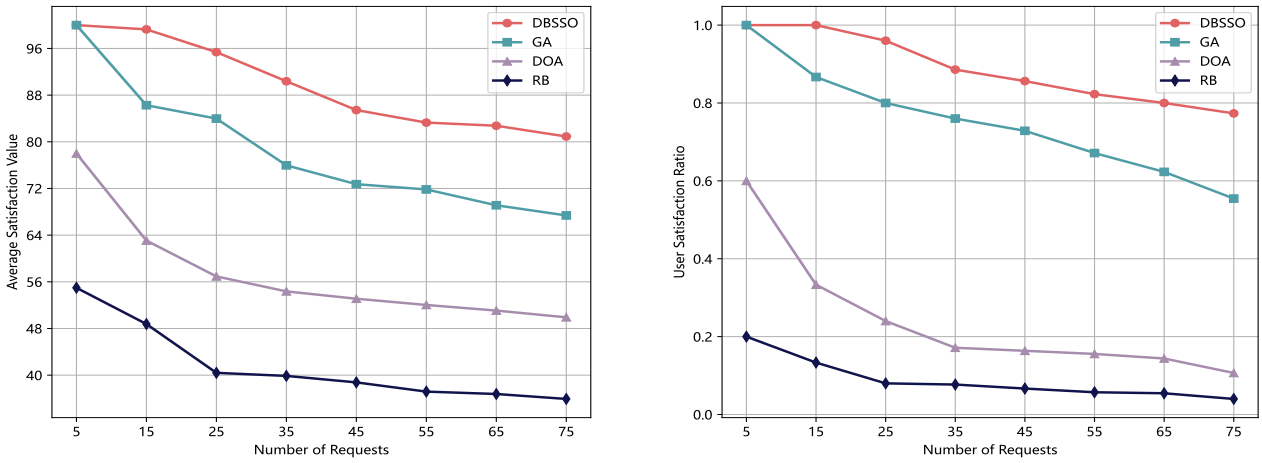


Fig. 2: Convergence trend of the total system profit with the training iteration.

parameter settings of the first-layer domain network elements and cloud layer in Table II.

Fig. 2 evaluates the convergence of each algorithm with 45 user requests. The proposed approach outperforms benchmarks in both convergence iterations and satisfaction value. Initially, all algorithms start at a satisfaction value of 38.8 due to random initialization. The proposed algorithm excels by maintaining population diversity, balancing local and global searches, and avoiding local optima. The satisfaction value of that reaches nearly 85 at 10-th iteration and converges, while the satisfaction values of GA and DOA continue to converge until nearly 72 on the 50-th and nearly 53 on the 30-th iterations, separately. In contrast, RB maintains the lowest total system satisfaction value, at nearly 38.

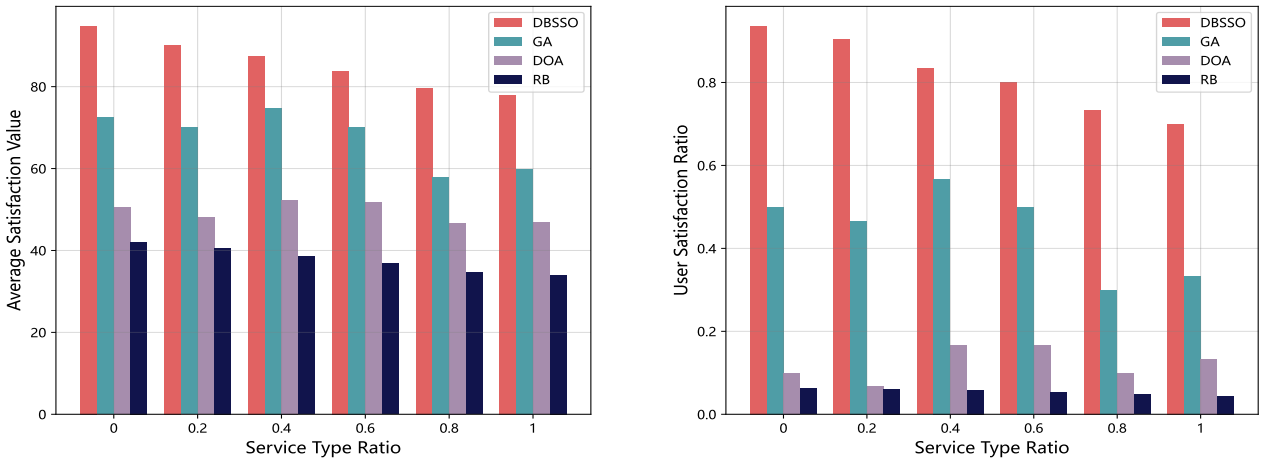
The plot in Fig. 3(a) depicts the comparison of the satisfaction value with the varied number of requests. The DBSSO algorithm demonstrates a consistently decreasing trend in satisfaction value as the number of requests increases, outperforming the benchmarks across most request levels. Specifically, at 5 requests, DBSSO and GA achieve a satisfaction value of approximately 100, compared to DOA at 77 and RB at 55. We witness a deeper decline in the satisfaction value values of GA, DOA, and RB compared to DBSSO as the number of requests increases to 15. As the number of requests reaches 75, DBSSO's satisfaction value achieves about 81, while GA, DOA, and RB achieve around 68, 50, and 36, respectively. Similarly, Fig. 3(b) plots the user satisfaction ratio against the number of requests. DBSSO consistently maintains the highest user satisfaction ratio across all request levels, which outperforms the best benchmark by nearly 20%. The clear separations in satisfaction value and user satisfaction ratio indicate that DBSSO is more effective in generating higher satisfaction values and fulfilling users with an increasing number of requests. Intuitively, satisfaction value should seemingly increase with the number of services. However, in this paper, we also measure user service satisfaction and cost within the definition of average satisfaction value. Therefore, the satisfaction value in this simulation decreases as the number of services increases. Although the trends of both appear



(a) Total system satisfaction value versus the number of requests.

(b) User satisfaction ratio versus the number of requests.

Fig. 3: Algorithm performance under different number of requests.



(a) Total system satisfaction value versus the service type ratio.

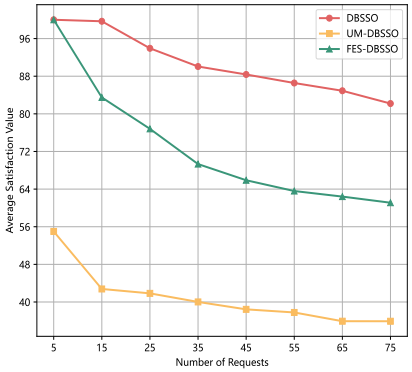
(b) User satisfaction ratio versus the service type ratio.

Fig. 4: Algorithm performance under different ratios of service type.

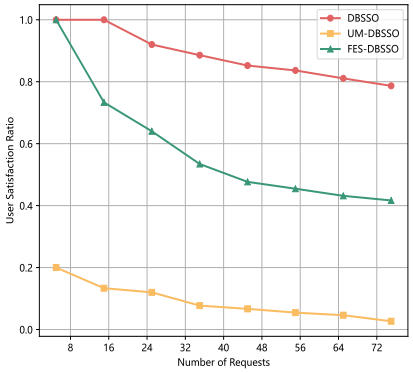
similar, it does not imply that satisfaction value iDespite the similarities between the trends, the user service acceptance rate does not dominate the satisfaction value. nce rate remains at 100%, yet the average satisfaction value decreases. This is because the increase in the number of users leads to resource contention, thereby reducing service performance.

The simulation results shown in Fig. 4 detail the performance of different algorithms under two service types: high-latency low-inference-performance tasks and low-latency high-inference-performance tasks. The horizontal axis represents the proportion of low-latency and high-inference-performance tasks. The figures demonstrate that the proposed approach outperforms each benchmark under varied service type ratios. As the ratio of low-latency high-inference-performance tasks increases, both the satisfaction value and the user satisfaction ratio decrease. This trend highlights the challenges in maintaining user satisfaction when computation resources on the edge are insufficient to meet the increased demand for low-latency services. The average satisfaction value, reflecting the overall service quality and resource cost balance, and the user satisfaction ratio, indicating the percentage of users whose expectations are met, both show a decline due to the limited computational capabilities available.

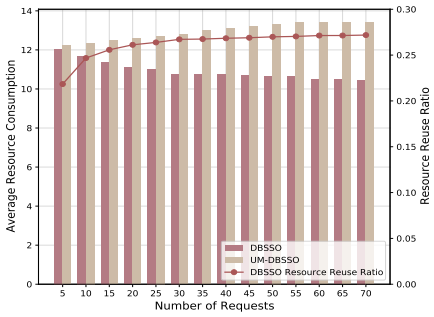
Figs. 5(a) and 5(b) present a comparison of the satisfaction value and user satisfaction ratio under two different stitching mechanisms: the unable-reusing mechanism-based DBSSO (UM-DBSSO) and the fair equal scheduling-based DBSSO (FES-DBSSO). Observations reveal a gradual decline in the performance of the three service strategies as the number of users increases. However, the stitching mechanism used has the slowest performance degradation rate. Although the FES-DBSSO performs similarly to DBSSO when the number of requests equals 5, its performance sharply declines, accounting for three quarters and half of DBSSO's satisfaction value and user satisfaction ratio. UM-DBSSO's performance is the lowest in both satisfaction value and user satisfaction ratio. Fig. 5(c) compares DBSSO and UM-DBSSO's performance in terms of average resource consumption and reuse. It can be seen that the resource consumption per completed service is significantly reduced by 15% to 20% as the number of services



(a) Total system satisfaction value versus the number of requests.

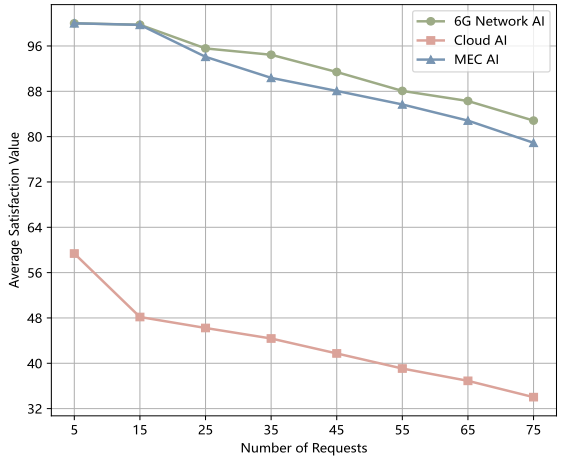


(b) User satisfaction ratio versus the number of requests.

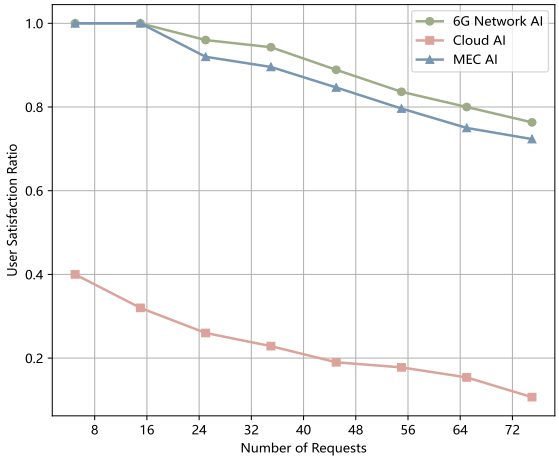


(c) Comparison between with model reusing and without reusing. The bars correspond to the resource consumption per request while the lines correspond to the resource reusing ratio.

Fig. 5: Performance of DBSSO algorithm under different stitching mechanism.



(a) Total system satisfaction value versus the service type ratio.



(b) User satisfaction ratio versus the service type ratio.

Fig. 6: Performance of DBSSO algorithm under different network architecture.

increases from 30 to 70. Although the resource reusing is close to 0% under 5 services, the reusing ratio rises rapidly as the number approaches 30. As the number of services increases, the resource reusing ratio continues to rise. The combination of stitching mechanisms with model reutilization allows the system to make full use of deployed AI models to reuse to serve users with different requirements. Therefore, the utilization of stitching mechanisms and model reusing promotes resource utilization and reduces resource costs.

Fig. 6 compares three architectures—6G Network AI, Cloud AI, and MEC AI—to demonstrate their impact on performance. Figs. 6(a) and 6(b) evaluate the satisfaction value and user satisfaction ratio across these network architectures as the number of users increases. All three architectures show a downward trend in these metrics as the number of users grows, albeit to varying degrees. When the number of requests is less than 10, the satisfaction values of 6G Network AI and MEC AI are equivalent and significantly higher than that of Cloud AI. As the number of requests continues to increase, a noticeable gap emerges, with 6G Network AI outperforming the others. This advantage stems from the combination of diverse network resources and the ability to handle varying user demands effectively. Cloud AI exhibits the lowest performance due to the high latency and transmission costs associated with centralized processing. In contrast, 6G Network AI employs a multi-layered approach, combining cloud, edge, and device-level resources, allowing for dynamic resource allocation and optimal service deployment. This integrated design leads to lower latency, enhanced adaptability, and overall better efficiency, thus enabling 6G Network AI to deliver superior performance compared to the other architectures.

VIII. CONCLUSION

In this paper, we introduced an AI service provision scheme for multi-layer heterogeneous network. Firstly, we introduce a network architecture based on native intelligence, leveraging the stitching technique to enhance AI model usability. Considering the service requirements of different users and the resource limitations, an MINLP problem is proposed to maximize user satisfaction value. Then this paper proposes a heuristic algorithm, DBSSO to jointly optimize the model stitching coefficient and service deployment. We designed a series of simulation experiments to verify the effectiveness of the proposed algorithm and architecture in maximizing the

satisfaction value and the satisfaction ratio. The results of the simulation experiment clearly show the advantages of this solution in improving service performance, and confirm its huge potential in 6G network on-demand service application scenarios. Overall, the on-demand service solution proposed in this article provides a general, innovative idea for realizing flexible and personalized services in 6G networks in the future. In future research, we will introduce user portrait technology and study the protocol interaction with existing protocol in core network. Then, the theoretical performance of proposed architecture is analyzed.

REFERENCES

[1] Statista. (2023) Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2023, with Forecasts from 2022 to 2030.

[2] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing Resource-Poor Mobile Devices with Powerful Clouds: Architectures, Challenges, and Applications," *IEEE Wireless communications*, vol. 20, no. 3, pp. 14–22, 2013.

[3] C. Zhou, J. Gao, M. Li, N. Cheng, X. Shen, and W. Zhuang, "Digital Twin-based 3D Map Management for Edge-assisted Device Pose Tracking in Mobile AR," *IEEE Internet of Things Journal*, vol. 11, no. 10, pp. 17 812–17 826, 2024.

[4] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly Accurate Protein Structure Prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[5] R. High, "The Era of Cognitive Systems: An Inside Look at IBM Watson and How It Works," *IBM Corporation, Redbooks*, vol. 1, p. 16, 2012.

[6] P. Sun, Y. Jiang, S. Chen, S. Zhang, B. Peng, P. Luo, and Z. Yuan, "Autoregressive Model Beats Diffusion: Llama for Scalable Image Generation," *arXiv preprint arXiv:2406.06525*, 2024.

[7] H. Zhang, A. Ning, R. B. Prabhakar, and D. Wentzlaff, "LLMCompass: Enabling Efficient Hardware Design for Large Language Model Inference," in *2024 ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 1080–1096.

[8] C. Chen, Z. Liao, Y. Ju, C. He, K. Yu, and S. Wan, "Hierarchical Domain-Based Multicontroller Deployment Strategy in SDN-Enabled Space-Air-Ground Integrated Network," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 6, pp. 4864–4879, 2022.

[9] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[10] L. Wang, X. Wu, Y. Zhang, X. Zhang, L. Xu, Z. Wu, and A. Fei, "DeepAdaIn-Net: Deep Adaptive Device-Edge Collaborative Inference for Augmented Reality," *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 5, pp. 1052–1063, 2023.

[11] H. Peng, Z. Zhang, Y. Liu, Z. Su, T. H. Luan, and N. Cheng, "Semantic Communication in Non-Terrestrial Networks: A Future-Ready Paradigm," *IEEE Network*, vol. 38, no. 4, pp. 119–127, 2024.

[12] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A Tutorial Overview of Standards, Trials, Challenges, Deployment, and Practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, 2017.

[13] Y. Yang, M. Ma, H. Wu, Q. Yu, X. You, J. Wu, C. Peng, T. S. P. Yum, A. H. Aghvami, G. Y. Li, J. Wang, G. Liu, P. Gao *et al.*, "6G Network AI Architecture for Everyone-Centric Customized Services," *IEEE Network*, vol. 37, no. 5, pp. 71–80, 2023.

[14] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "How Should I Slice My Network? A Multi-Service Empirical Evaluation of Resource Sharing Efficiency," in *2018 ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2018, p. 191–206.

[15] C. Zhou, J. Gao, M. Li, X. Shen, W. Zhuang, X. Li, and W. Shi, "AI-Assisted Slicing-Based Resource Management for Two-Tier Radio Access Networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 9, no. 6, pp. 1691–1706, 2023.

[16] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic Network Virtualization and Pervasive Network Intelligence for 6G," *IEEE Communications Surveys and Tutorials*, vol. 24, no. 1, pp. 1–30, 2022.

[17] M. Li, D. Yin, H. Qiu, and B. Bai, "A Systematic Review of AI Technology-based Service Encounters: Implications for Hospitality and Tourism Operations," *International Journal of Hospitality Management*, vol. 95, p. 102930, 2021.

[18] Y. Lin, C. Wu, J. Wu, L. Zhong, X. Chen, and Y. Ji, "Meta-Networking: Beyond the Shannon Limit with Multi-Faceted Information," *IEEE Network*, vol. 37, no. 4, pp. 256–264, 2023.

[19] M.-H. Huang and R. T. Rust, "Artificial Intelligence in Service," *Journal of Service Research*, vol. 21, no. 2, pp. 155–172, 2018.

[20] W. Wu, C. Zhou, M. Li, H. Wu, H. Zhou, N. Zhang, X. Shen, and W. Zhuang, "AI-Native Network Slicing for 6G Networks," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 96–103, 2022.

[21] L. Wang, L. Li, L. Xu, X. Peng, and A. Fei, "Failure-Resilient Distributed Inference With Model Compression Over Heterogeneous Edge Devices," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 12 680–12 692, 2024.

[22] M. Li, J. Gao, C. Zhou, X. 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 Vehicular Technology Magazine*, vol. 16, no. 4, pp. 16–26, 2021.

[23] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen, "Accuracy-Guaranteed Collaborative DNN Inference in Industrial IoT via Deep Reinforcement Learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4988–4998, 2021.

[24] W. He, S. Guo, S. Guo, X. Qiu, and F. Qi, "Joint DNN Partition Deployment and Resource Allocation for Delay-Sensitive Deep Learning Inference in IoT," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9241–9254, 2020.

[25] W. Fan, Z. Chen, Z. Hao, Y. Su, F. Wu, B. Tang, and Y. Liu, "DNN Deployment, Task Offloading, and Resource Allocation for Joint Task Inference in IIoT," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1634–1646, 2023.

[26] G. Manogaran, T. Baabdullah, D. B. Rawat, and P. M. Shakeel, "AI-Assisted Service Virtualization and Flow Management Framework for 6G-Enabled Cloud-Software-Defined Network-Based IoT," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14 644–14 654, 2022.

[27] Y. Hu, Q. Li, Y. Chai, D. Wu, L. Lu, N. Shi, Y. Teng, and Y. Zhang, "AI Service Deployment and Resource Allocation Optimization Based on Human-Like Networking Architecture," *IEEE Internet of Things Journal*, vol. 14, no. 8, 2024.

[28] N. Cheng, W. Xu, W. Shi, Y. Zhou, N. Lu, H. Zhou, and X. Shen, "Air-Ground Integrated Mobile Edge Networks: Architecture, Challenges, and Opportunities," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 26–32, 2018.

[29] Z. Yin, N. Cheng, Y. Hui, W. Wang, L. Zhao, K. Aldubaikhy, and A. Alqasir, "Multi-Domain Resource Multiplexing Based Secure Transmission for Satellite-Assisted IoT: AO-SCA Approach," *IEEE Transactions on Wireless Communications*, 2023, early access, DOI: 10.1109/TWC.2023.3250227.

[30] F. Tang, C. Wen, L. Luo, M. Zhao, and N. Kato, "Blockchain-Based Trusted Traffic Offloading in Space-Air-Ground Integrated Networks (SAGIN): A Federated Reinforcement Learning Approach," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3501–3516, 2022.

[31] Z. Liu, L. Wan, Y. Ma, J. Guo, S. Guo, J. Ma, and J. Ma, "Establishing Trustworthy and Privacy-Preserving SAGIVNs in 6G: Architectures, Requirements, and Solutions," *IEEE Network*, vol. 38, no. April, pp. 141–147, 2023.

[32] J. He, N. Cheng, Z. Yin, H. Zhou, C. Zhou, K. Aldubaikhy, A. Alqasir, and X. Shen, "Load-Aware Network Resource Orchestration in LEO Satellite Network: A GAT-Based Approach," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 15 969 – 15 984, 2024.

[33] S. Yu, X. Gong, Q. Shi, X. Wang, and X. Chen, "EC-SAGINs: Edge-Computing-Enhanced Space-Air-Ground-Integrated Networks for Internet of Vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5742–5754, 2022.

[34] C. Huang, G. Chen, S. Member, P. Xiao, and S. Member, "Joint Offloading and Resource Allocation for Hybrid Cloud and Edge Computing in SAGINs : A Decision Assisted Hybrid Action Space Deep Reinforcement Learning Approach," *IEEE Journal on Selected Areas in Communications*, vol. PP, p. 1, 2024.

[35] Q. Chen, W. Meng, T. Q. Quek, and S. Chen, "Multi-Tier Hybrid Offloading for Computation-Aware IoT Applications in Civil Aircraft-Augmented SAGIN," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 399–417, 2023.

[36] H. Zhang, R. Liu, A. Kaushik, and X. Gao, "Satellite Edge Computing with Collaborative Computation Offloading: An Intelligent Deep Deterministic Policy Gradient Approach," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 9092–9107, 2023.

[37] C. Zhou, J. Gao, M. Li, X. Shen, and W. Zhuang, "Digital Twin-empowered Network Planning for Multi-tier Computing," *Journal of Communications and Information Networks*, vol. 7, no. 3, pp. 221–238, 2022.

[38] H. Zhang, S. Shao, M. Tao, X. Bi, and K. B. Letaief, "Deep Learning-Enabled Semantic Communication Systems With Task-Unaware Transmitter and Dynamic Data," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 1, pp. 170–185, 2023.

[39] Y. Bo, Y. Duan, S. Shao, and M. Tao, "Joint Coding-Modulation for Digital Semantic Communications via Variational Autoencoder," *IEEE Transactions on Communications*, vol. PP, p. 1, 2024.

[40] Z. Pan, J. Cai, and B. Zhuang, "Stitchable Neural Networks," in *2023 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 16 102–16 112.

[41] W. Wu, M. Li, K. Qu, C. Zhou, X. Shen, W. Zhuang, X. Li, and W. Shi, "Split Learning Over Wireless Networks: Parallel Design and Resource Management," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 1051–1066, 2023.

[42] J. Li, W. Shi, H. Wu, S. Zhang, and X. Shen, "Cost-Aware Dynamic SFC Mapping and Scheduling in SDN/NFV-Enabled Space-Air-Ground Integrated Networks for Internet of Vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5824–5838, 2021.

[43] J. Xue and B. Shen, "Dung Beetle Optimizer: A New Meta-heuristic Algorithm for Global Optimization," *The Journal of Supercomputing*, vol. 79, no. 7, pp. 7305–7336, 2023.

[44] G. Wang, S. Zhou, S. Zhang, Z. Niu, and X. Shen, "SFC-Based Service Provisioning for Reconfigurable Space-Air-Ground Integrated Networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1478–1489, 2020.

[45] M. Lopuhaä-Zwakenberg, C. E. Budde, and M. Stoelinga, "Efficient and Generic Algorithms for Quantitative Attack Tree Analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp. 4169–4187, 2023.

[46] N. Howgrave-Graham and A. Joux, "New Generic Algorithms for Hard Knapsacks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 235–256.