Cost-minimized Microservice Migration with Autoencoder-assisted Evolution in Hybrid Cloud and Edge Computing Systems

Jiahui Zhai, *Student Member, IEEE*, Jing Bi, *Senior Member, IEEE*, Haitao Yuan, *Senior Member, IEEE*, Mengyuan Wang, Jia Zhang, *Senior Member, IEEE*, Yebing Wang, *Senior Member, IEEE*, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Hybrid cloud-edge systems combine the advantages of cloud computing and mobile edge computing (MEC) to achieve flexible integration and fluidity of data between the cloud and the edge. To address dynamic and stochastic loads caused by mobile users (MUs) and time-varying tasks, MEC network operators need to continuously migrate installed services among edge servers, significantly increasing network maintenance costs. Existing studies often overlook the service migration cost resulting from MU mobility. Therefore, we present a joint optimization scheme focusing on minimizing the operational cost of hybrid cloud-edge systems while considering the dynamic service migration cost induced by MUs. With the rapid development of 5G/6G technologies, many MUs require connectivity to edge nodes (ENs) or cloud data centers (CDCs) for processing. Minimizing the operational cost of hybrid cloud-edge systems while considering many heterogeneous decision variables is a challenge. To solve this complex high-dimensional mixed-integer nonlinear problem, we develop a novel deep learning-based evolutionary algorithm called Autoencoder-based Multi-swarm Grey wolf optimizer based on Genetic learning (AMGG). Experimental results with real data demonstrate that AMGG achieves lower system cost by 49.69% while strictly meeting task latency requirements of MUs compared with state-of-the-art algorithms.**

***Index Terms*—Mobile edge computing, service migration, autoencoders, high-dimensional optimization algorithms, grey wolf optimizer.**

I. Introduction

With the emergence of mobile services such as instant messaging, navigation, autonomous driving, and streaming media, the demand for delay-sensitive and compute-intensive services among mobile users (MUs) has rapidly increased [1]. A revolutionary paradigm of mobile cloud computing has emerged, allowing latency-sensitive and compute-intensive services of MUs to run remotely in cloud data centers (CDCs) [2]. However, due to the significant geographical distance between MUs and CDCs, the remote physical location of CDCs often leads to unpredictable long-term latency issues, severely affecting the quality of service (QoS) for MUs. Mobile edge computing (MEC) has emerged to address this issue. It provides cloud computing capabilities at edge nodes (ENs) in mobile networks. The basic principle of MEC involves deploying a small base station (SBS) near MUs in each EN, with each SBS connected to an MEC server [3]. This proximity deployment routes tasks from MUs to nearby ENs in the coverage range, thereby accelerating task execution speed and effectively reducing significant communication latency and cost between MUs and CDCs [4]. However, in 5G/6G communication, due to limited computing and communication capabilities in ENs, MEC cannot meet higher demands for MU service mobility and greater data processing capabilities. To address this issue, distributing MU services to a hybrid cloud-edge system to facilitate real-time collaborative management of communication and computing resources can further enhance resource utilization efficiency, effectively meeting MUs' QoS requirements [5].

However, hybrid cloud-edge systems face several major challenges. First, to accommodate the mobility of MUs, MEC network operators implement frequent service migrations among various microservices of MU requests [6]. This architecture allows MUs to move in different EN coverage areas while receiving real-time services. However, this strategy significantly increases network operating cost [7]. For network operators, the significant challenge of microservices in a hybrid cloud-edge system is minimizing the system cost while ensuring the QoS of MUs. Second, with the development of 5G/6G technologies and the widespread deployment of ENs, the density of ENs continues to increase. EN density is expected to reach 50 ENs per square kilometer in the future [8]. MUs may simultaneously be in the overlapping coverage areas of multiple ENs. In this multi-MEC environment, MUs can send tasks to any covered EN for processing [9]. However, given the constraints of limited resources and dense EN deployments, effectively performing appropriate task routing

This work was supported by the National Natural Science Foundation of China under Grants 62173013 and 62073005, the Beijing Natural Science Foundation under Grants L233005 and 4232049, in part by Beihang World TOP University Cooperation Program, and China Scholarship Council. (Corresponding author: Haitao Yuan.)

J. Zhai and J. Bi are with the College of Computer, Beijing University of Technology, Beijing 100124, China. (e-mail: zhaijiahui@emails.bjut.edu.cn; bijing@bjut.edu.cn).

H. Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China. (e-mail: yuan@buaa.edu.cn).

M. Wang is with the School of Energy and Power Engineering, Beihang University, Beijing 100191, China. (e-mail: mengyuanwang@buaa.edu.cn).

J. Zhang is with the Department of Computer Science, Southern Methodist University, Dallas, TX 75206, USA. (e-mail: jiazhang@smu.edu).

Y. Wang is with the Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA. (e-mail: yebinwang@ieee.org).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA. (e-mail: zhou@njit.edu).

and service deployment is a significant challenge [10]. Despite the numerous solutions proposed for service migration in MEC networks [6], [7], [18]–[25], several challenges remain. Specifically, MEC networks' dynamic and stochastic nature, such as MU mobility and temporal variability of tasks, pose challenges for the required request routing and service deployment. Additionally, MEC servers' limited computing, memory, and storage resources greatly impact decision variables of optimization problems, which are overlooked in some related studies [6], [7]. Furthermore, with the rapid increase in MUs in MEC networks, there is a sharp increase in decision variables. Therefore, this problem has become typical high-dimensional complex optimization that must be quickly addressed in a hybrid cloud-edge system. The optimization problems discussed above are low-dimensional, and their methods cannot solve high-dimensional complex optimization problems [11], [12].

This work makes the following three novel contributions to the hybrid cloud-edge domain to address the above-mentioned challenges.

1) This work introduces an improved architecture for hybrid cloud-edge systems comprising multiple MUs, ENs, and a CDC. Additionally, it proposes a large-scale constrained cost minimization problem subject to various constraints, including transmission, computation, and dynamic service migration cost. The problem is a high-dimensional mixed-integer nonlinear program (MINLP).
2) To address it, this work introduces a deep learning-driven hybrid optimization algorithm termed as Autoencoder-based Multi-swarm Grey wolf optimizer based on Genetic learning (AMGG). AMGG synergistically combines an autoencoder [13] and a novel Multi-swarm Grey wolf optimizer based on Genetic learning (MGG) proposed in this work.
3) To balance exploration and exploitation during the optimization process, two sub-populations of AMGG evolve cooperatively in an asynchronous manner: one sub-population evolves in the high-dimensional space following the original MGG. In contrast, the other sub-population evolves in a lower-dimensional space by integrating feature extraction and dimension reduction techniques alongside MGG. Besides, AMGG exchanges information dynamically between two sub-populations.

In addition, this work employs real-world data and evaluates the performance of AMGG. Extensive results with real data demonstrate that AMGG achieves lower system cost by 49.69% than other state-of-the-art peers, including self-adaptive bat algorithm with genetic operations (SBAGO) [14], genetic simulated annealing (SA)-based particle swarm optimization (GSP) [15], hybrid genetic-grey wolf optimization (HGGWO) [16], and SA-based particle swarm optimization (SAPSO) [17]. To clearly show the novelty of our work, we compare this work with state-of-the-art studies in Table I.

Section II provides an overview and a comparison of relevant studies. Section III elucidates the architecture of hybrid cloud-edge systems and formulates the system cost optimization problem. Section IV outlines specific details of AMGG. Section V discusses the performance evaluation results of AMGG. Section VI concludes the work.

II. Related work

A. Cost optimization in hybrid cloud-edge systems

Several studies on MEC network optimization focus on optimizing the cost in hybrid cloud-edge systems [7], [18]–[20]. Ouyang *et al.* [7] primarily focus on optimizing performance in mobile services perception under long-term cost constraints, with a principal emphasis on exploring the trade-off between migration cost and latency in mobile services. However, it fails to consider edge servers' computational capacity, memory, and disk storage limitations. Taleb *et al.* [18] introduce Follow-Me Cloud (FMC), an architecture for codifying cellular mobile networks. FMC optimizes service delivery and connectivity for MUs, employing a Markov decision process and two continuity schemes. Analytical modeling and experiments highlight FMC's user experience and resource utilization advantages. However, it only considers service deployment in cloud servers and ignores user associations between MUs and ENs in a hybrid cloud-edge system. Yuan *et al.* [19] propose a multi-agent deep reinforcement learning algorithm to address service migration cost issues and mobility optimization in vehicular edge computing. However, it overlooks the energy constraints of edge servers in ENs. Additionally, Wang *et al.* [20] investigate a computation cost-based prediction method to forecast the future time cost parameters for optimizing the deployment locations of services, to minimize the average cost in a specified time frame. However, it ignores the service migration cost brought by the mobility of MUs.

Unlike these methods, this work considers the diverse costs associated with migrating MU services across multiple cloud-edge systems. Specifically, the cost of service migration for MUs fluctuates due to their dynamic location changes. Additionally, factors such as service latency, data transmission cost during migration, computational expenses for service execution, and migration cost vary depending on the types of services being migrated and the deployment scenarios. Moreover, our method emphasizes analyzing service migration costs across multiple system layers during the dynamic movement of MUs rather than solely focusing on static service deployment in a single system. Furthermore, it considers the practical constraints of task routing and service deployment in the hybrid cloud-edge system, enhancing its comprehensiveness and applicability.

B. Joint optimization of request routing and service deployment

Some related studies concentrate on the joint optimization of request routing and service deployment. Wang *et al.* [6] propose a reinforcement learning-based algorithm to determine the optimal edge for offloading computation, aiming to optimize the overall migration costs and service latency during user mobility. It jointly considers the decision variables of microservice coordination under the available processing capacity of edge servers, data transmission conditions, and service requests. Unlike it, we focus on jointly optimizing request routing and service deployment to minimize the total

TABLE I
COMPARISON OF THE PROPOSED METHOD AND STATE-OF-THE-ART STUDIES

Novelties		Our work	[7]	[18]	[19]	[20]	[6]	[23]	[24]	[25]	[26]
System architectures	Hybrid cloud-edge systems	✓				✓	✓	✓		✓	✓
	Multi-task offloading	✓					✓	✓	✓	✓	
	Dynamic microservice migration	✓				✓	✓	✓	✓		✓
	Mobility-aware service deployment	✓	✓	✓	✓	✓		✓	✓		✓
Algorithms	Task priority								✓	✓	
	Heuristic algorithms	✓						✓	✓		
	Deep learning-based algorithms	✓			✓						✓
	High-dimensional optimization	✓			✓						
Performance analysis	Cost minimization	✓	✓	✓	✓						
	Latency minimization				✓	✓	✓	✓		✓	✓

cost of the cloud-edge environment. He *et al.* [23] investigate the optimal edge service configuration that satisfies the requirements of both shareable resources (storage) and non-shareable resources (computing and communication). Nevertheless, it does not consider the memory resource requirements of edge servers. Yu *et al.* [24] conduct an effective three-stage method to optimize request routing and service deployment to minimize resource consumption and end-to-end response time. However, it only considers a service provider that provides a cloud infrastructure with a set of microservices in a service environment. Unlike this, we aim to design a request routing and service deployment framework following a collaborative paradigm with edge and cloud. In [25], the joint optimization of request routing and service deployment in MEC networks is studied. This work proposes a solution that considers various MEC constraints in practical scenarios, aiming to minimize the number of requests routed to the cloud center while ensuring the overall system performance without violating MEC resource constraints. However, the solution above does not consider the dynamic movement of MUs over time and the long-term cost of the system. Sahil *et al.* [26] propose a Fog-Cloud centric Internet of Things(IoT)-based collaborative framework that enables machine learning-based situation-aware traffic management. The integrated Fog-Cloud-IoT framework can improve real-time data processing, reduce network congestion, and improve situation awareness.

Different from the above studies, this work initially constructs a microservice-based service migration architecture for a hybrid cloud-edge system, including MUs, ENs, and CDC. Subsequently, we consider the cost components of all constituents in the entire system and formulate them into a constrained long-term total cost optimization problem. We jointly optimize request routing and service deployment, with decision variables including request routing, CPU resource allocation for ENs, service deployment decisions for ENs, service migration cost for ENs, transmission power, and channel bandwidth allocation for MUs. To tackle the high-dimensional optimization problem, we propose a novel algorithm named AMGG, which integrates the evolution of MGG in high-dimensional search spaces and an autoencoder-assisted optimization approach in low-dimensional spaces to extract the most useful information and key features from the population.

In contrast to our preliminary work [27], this work makes three major enhancements. First, this work designs a novel mechanism where two sub-populations evolve in high and low-dimensional spaces asynchronously to reduce the space and time complexity of AMGG significantly. Second, a cost-computing model is enhanced. Specifically, we consider the modeling of CPU, memory, and storage in ENs under realistic service migration scenarios. Third, this work evaluates AMGG more comprehensively, including two benchmark scenarios to demonstrate its superiority in solving the formulated MINLP over its peers.

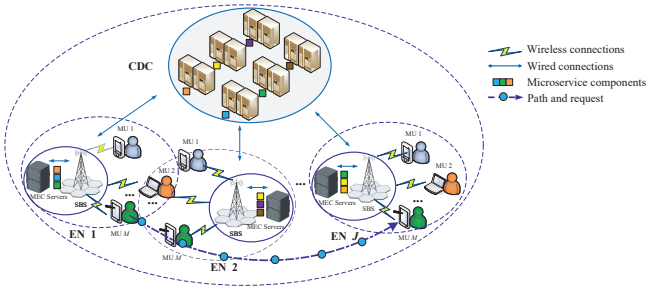


Fig. 1. The proposed service migration architecture for microservices in the hybrid cloud-edge systems.

III. PROBLEM FORMULATION

We emphasize the scalability and adaptability of potential real-world applications of our framework to address pressing challenges such as frequent service migrations required for MU mobility, raising network costs while maintaining QoS and increasing EN density in task routing and service deployment, especially with limited resources. Specifically, our proposed solution is designed to provide real-time, data-driven, and cost-minimized decision-making capabilities, which are critical for managing complex systems in dynamic environments.

To further demonstrate its applicability, this work considers microservice applications similar to [28]. Each MU application is decoupled into multiple functionally independent microservices. Fig. 1 illustrates that MUs can send tasks to SBSs in the coverage area of ENs via wireless connections. These route requests are made to request the migration of microservices to MEC servers connected via wired links to SBSs or to be executed in CDC. A service scheduler deployed in MEC servers covered by ENs gathers information regarding the real-time dynamic movement of MUs, uplink/downlink

channel conditions, completion time requirements of MUs, and available resources in ENs/CDCs. Subsequently, the scheduler determines the optimal service migration and deployment strategy. This section formulates the problem of minimizing the total cost of hybrid cloud-edge systems. We introduce the system model, followed by the system total cost model, the model for CPU, memory, and storage in ENs, and the latency model. Finally, we formulate a constrained cost minimization problem. For clarity, Table II lists the system's main notations and decision variables in this section.

A. System model

Let M denote the number of MUs. It is assumed that an application comprises K microservices, each microservice $s_m^k(1{\leq}m{\leq}M, 1{\leq}k{\leq}K)$ is characterized by a four tuple, and $s_m^k{=}(\varsigma_m^k, w_m^k, q_m^k, \varpi_m^k)$. Here, ς_m^k represents the storage capacity occupied by s_m^k (bits). w_m^k denotes the computational intensity required by s_m^k (cycles/sec.). q_m^k represents the data scale of the service request of s_m^k (bits). ϖ_m^k represents the total computational workload of s_m^k (cycles). We denote each time slot as $t(1{\leq}t{\leq}T)$ as the unit of time. At each time slot t, MU m generates K tasks, denoted as $s_m^k(t)$, $s_m^k(t){=}\left(\varsigma_m^k(t), w_m^k(t), q_m^k(t), \varpi_m^k(t)\right)$. $\lambda_{m,j}(t)$ and $\lambda_{m,0}(t)$ are binary variables that denote the routing status of service request of $s_m^k(t)$ at time slot t. Specifically, $\lambda_{m,j}(t){\in}\{0,1\}$ and $\lambda_{m,0}(t){\in}\{0,1\}$, *i.e.*, when an EN accepts a service request of $s_m^k(t)$ from MU m and forwards it to EN j $(1{\leq}j{\leq}J)$ for computation at time slot t, $\lambda_{m,j}(t){=}1$; otherwise, it's directed to CDC, $\lambda_{m,0}(t){=}1$. Service request of $s_m^k(t)$ is exclusively routed for processing to either EN or CDC. Hence, for each MU m covered by EN j, we express:

$$\sum_{j=0}^{J} \lambda_{m,j}(t){=}1 \tag{1}$$

It's crucial to emphasize that this work is centered on processing decisions at the current time slot. If a task is not completed in the current time slot, its remaining portion is treated as a new task at the subsequent time slot. Based on the routing decision for the request, the system can manage this task in the new time slot. Only the execution environment configured for $s_m^k(t)$ can execute the task on the edge server in EN.

This work utilizes a multi-access communication model with orthogonal frequency division for each MU and its respective EN/CDC [31]. Following the communication model proposed by [32], $(d_{m,j})^{-v}$ denotes the trajectory loss linking MU m and its corresponding EN j. $d_{m,j}$ represents the distance from MU m to EN j. v denotes the path loss exponent. Let $R_{m,j}^U(t)$ denote the uplink data rate between MU m and EN j at time slot t, *i.e.*,

$$R_{m,j}^U(t){=}\mu_{m,j}(t)W_j^U \log_2(1{+}\frac{P_m(t)(d_{m,j})^{-v}\left|h_1\right|^2}{G_0}) \tag{2}$$

where $\mu_{m,j}(t)(0{\leq}\mu_{m,j}(t){\leq}1)$ represents the uplink channel bandwidth utilized by MU m in EN j at time slot t. W_j^U denotes the uplink channel bandwidth of EN j. $P_m(t)$ signifies the transmission power of MU m at time slot t. h_1 denotes the uplink fading coefficient, and G_0 represents the power of the additive white Gaussian noise. We assume all MUs connected to EN j share its channel bandwidth. Therefore, for each EN

TABLE II
MAIN NOTATIONS IN SECTION III

Notations	***Definition***
M	Number of MUs
K	Number of services
J	Number of ENs
ς_m^k	Service s_m^k size of occupying storage capacity (bits)
q_m^k	Data scale of the service request of s_m^k (bits)
ϖ_m^k	Total computational workload of s_m^k (cycles)
$d_{m,j}$	Distance from MU m to EN j
v	Path loss exponent
h_1	Uplink fading coefficient
G_0	Power of the additive white Gaussian noise
W_j^U	Uplink channel bandwidth of EN j
P_m^0	Power consumption of MU m in an idle state
ρ_m	Power amplifier utilized to transmit data from MU m
β_1	Data transmission cost per unit in each uplink channel from each MU to each EN
$\epsilon(\nu)$	Coefficients that measure the cost of communication via the uplink (downlink) channels
$\hat{P}_m$	Maximum limit of transmission power of MU m
σ_j	Fixed value determined by the chip structure of EN j
P_δ^1	Power consumption when each EN transmits service request of s_m^k to CDC via the wired line
β_2	Data transmission cost per unit in each uplink channel from each EN to CDC
e_0	Energy consumed by each CPU cycle in the CDC
r_0	Transmission rate (bits/sec.) of the backhaul connection between ENs and CDC
$\hat{E}_j$	Maximum available energy in EN j
$\hat{F}_j$	Maximum CPU speed of EN j
τ	Weighting factor for the cost of energy consumption
$\hat{z}$	Maximum cost of migrating service $s_m^k(t)$ to EN j
$\tilde{C}$	Cost budget of the T time slots
Ψ	Total cost of the hybrid cloud-edge system
$\hat{A}_j$	Maximum value of CPU cycles in each EN j
$\hat{G}_j$	Maximum limit of memory in EN j
ϑ_m^k	Amount of memory required for each data bit of service request of s_m^k
$\hat{S}_j$	Maximum limit of storage in EN j
w_0	Computing speed of CDC (cycles/sec.)
T_m	Total completion time required for transmitting and processing K services in ENs and CDC
$\hat{T}_m$	Latency constraint of T_m
$\vec{\upsilon}$	Vector of decision variables
$\widetilde{\Delta}$	Transformed new objective function utilized to compute the fitness value of each solution in AMGG
$\mathcal{N}$	Significantly big positive constant
Θ	Total penalty for all constraints
$\mathbb{N}^{=}(\mathcal{N}^{\neq})$	Number of equality (inequality) constraints
Variable	***Definition***
w_m^k	Computational intensity required by s_m^k (cycles/sec.)
$\lambda_{m,j}$	Binary variable that denotes the routing status of service request of s_m^k between MU m and EN j
$\lambda_{m,0}$	Binary variable that denotes the routing status of service request of s_m^k between MU m and CDC
$\mu_{m,j}$	Uplink channel bandwidth utilized by MU m in EN j
P_m	Transmission power of MU m
y_j^k	Binary variable that represents the decision to deploy service s_m^k in EN j
z_j^k	Cost of migrating service k to EN j

j, we have:

$$\sum_{m=1}^{M} \lambda_{m,j}(t)\mu_{m,j}(t)=1 \tag{3}$$

For most traditional delay-sensitive services, the downlink has significantly lower energy consumption and latency than the uplink, so we simplify by overlooking the downlink's energy consumption and latency [33].

B. Total cost model

For a given service request of $s_m^k(t)$, associated costs are incurred in transmitting and processing it. These costs comprise three aspects: a) the cost of transmitting the data related to service request of $s_m^k(t)$; b) the computational cost incurred by ENs/CDC in executing service $s_m^k(t)$; and c) the cost of migrating service $s_m^k(t)$ in the EN.

1) Cost model of data transmission

P_m^0 represents the power consumption of MU m in idle state. ρ_m represents the coefficient of the power amplifier utilized to transmit data from MU m. Let $\phi_{m,j}^1$ and $C_{m,j}^1$ denote the energy and cost of uploading data between MU m and EN j, respectively. $\phi_{m,j}^1$ and $C_{m,j}^1$ are computed as:

$$\begin{aligned}\phi_{m,j}^1(t)=\sum_{k=1}^{K}\{&(P_m^0+\rho_m\mu_{m,j}(t)P_m(t))\\ &\frac{\beta_1(\lambda_{m,j}(t)+\lambda_{m,0}(t))q_m^k(t)}{R_{m,j}^U(t)}\}\end{aligned} \tag{4}$$

$$C_{m,j}^1(t)=\sum_{k=1}^{K}\{(\lambda_{m,j}(t)+\lambda_{m,0}(t))q_m^k(t)(\epsilon+\nu)\} \tag{5}$$

where $\beta_1\,(\beta_1>0)$ represents the data transmission cost per unit in each uplink channel from each MU to each EN [34]. ϵ and ν denote the coefficients that measure communication cost via uplink and downlink channels, respectively. Additionally, $\hat{P}_m$ denotes represents the upper boundary of $P_m(t)$. Then,

$$0\le P_m(t)\le\hat{P}_m \tag{6}$$

2) Cost model of ENs/CDC

Let $E_{m,j}(t)$ represent the energy consumption incurred when MU m's service request of $s_m^k(t)$ is routed to EN j or CDC at time slot t. Naturally, $E_{m,j}(t)$ comprises two components, denoted as $\phi_{m,j}(t)$ and $\phi_{m,0}(t)$, which represents the energy consumption when MU m's service request of $s_m^k(t)$ is routed to EN and CDC respectively. $E_{m,j}(t)$ is expressed as:

$$E_{m,j}(t)=\phi_{m,j}(t)+\phi_{m,0}(t) \tag{7}$$

$\phi_{m,j}(t)$ also consists of two components including $\phi_{m,j}^1(t)$ and $\phi_{m,j}^2(t)$. $\phi_{m,j}^1(t)$ denotes the energy consumption for uploading service request data of $s_m^k(t)$ and $\phi_{m,j}^2(t)$ denotes the energy consumption for executing the service $s_m^k(t)$ from MU m in EN j. Therefore, we can derive:

$$\phi_{m,j}(t)=\phi_{m,j}^1(t)+\phi_{m,j}^2(t) \tag{8}$$

$$\phi_{m,j}^2(t)=\sum_{k=1}^{K}\{\sigma_j\lambda_{m,j}(t)\varpi_m^k(t)(w_{m,j}^k(t))^2\} \tag{9}$$

where σ_j represents a fixed value determined by the chip structure of EN j [15].

In addition, ENs and CDC are connected via low-latency wired optical fiber transmission [35]. P_δ^1 denotes the power consumption when each EN transmits service request of $s_m^k(t)$ to CDC via the wired line. $\beta_2\,(\beta_2>0)$ represents the data transmission cost per unit in each uplink channel from each EN to CDC. e_0 represents the energy consumed by each CPU cycle in the CDC. Therefore, we obtain $\phi_{m,0}(t)$ as:

$$\phi_{m,0}(t)=\sum_{k=1}^{K}(\frac{P_\delta^1\beta_2\lambda_{m,0}(t)q_m^k(t)}{r_0}+\varpi_m^k(t)\lambda_{m,0}(t)e_0) \tag{10}$$

where r_0 represents the transmission rate (bits/sec.) of the backhaul connection between ENs and CDC.

$\hat{E}_j$ deotes the maximum available energy in EN j. The total energy consumed by the tasks of MUs to EN j cannot exceed $\hat{E}_j$. Therefore,

$$\sum_{m=1}^{M}\lambda_{m,j}(t)\phi_{m,j}(t)\le\hat{E}_j \tag{11}$$

$\tilde{F}_j$ denotes the maximum CPU speed of EN j. Therefore, the total execution speed for servicing K services from M MUs cannot exceed $\tilde{F}_j$, *i.e.*,

$$\sum_{m=1}^{M}\sum_{k=1}^{K}\lambda_{m,j}(t)w_{m,j}^k(t)\le\tilde{F}_j, w_{m,j}^k(t)\in N^+ \tag{12}$$

Let $C_{m,j}^2(t)$ denote the energy cost for transmitting K tasks and performing K services for MU m covered by EN j. Therefore, $C_{m,j}^2$ is given as:

$$C_{m,j}^2(t)=\tau E_{m,j}(t) \tag{13}$$

where τ represents a weighting factor for the cost of energy consumption [36].

3) Cost model of service migration

A binary $y_j^k(t)\in\{0,1\}$ is defined where if service $s_m^k(t)$ is deployed in EN j at time slot t, $y_j^k(t)=1$; otherwise, $y_j^k(t)=0$. Given the uncertain nature of MU mobility, it is essential to quickly deploy microservices to ENs near MUs to ensure uninterrupted real-time services. Nevertheless, services' dynamic deployment and relocation result in increased operational expenses. Let $C_j^3(t)$ denotes the cost of migrating a service to EN j at time slot t, *i.e.*,

$$C_j^3(t)=\sum_{k=1}^{K}\{z_j^k(t)\Phi(y_j^k(t)>y_j^k(t-1))\} \tag{14}$$

where $z_j^k(t)$ represents a decision variable for the cost of migrating service k to EN j at time slot t, and $\hat{z}$ is the maximum of $z_j^k(t)$. $\Phi(\cdot)$ is the 0-1 indicator function that indicates the necessity of migration. If $y_j^k(t)=1$ and $y_j^k(t-1)=0$, *i.e.*, $\Phi(y_j^k(t)>y_j^k(t-1))=1$, service k is migrated from CDC to EN j. To route the service request of $s_m^k(t)$ from MU m to EN

j, it is necessary to deploy the corresponding service k in EN j, *i.e.*,

$$\lambda_{m,j}(t)\leqslant y_j^k(t), \forall j\in J \tag{15}$$

$\tilde{C}$ denotes the cost budget of the T time slots. Therefore, we have:

$$\sum_{t=0}^{T-1}\sum_{j=1}^{J} C_j^3(t)\leq \tilde{C} \tag{16}$$

4) Total cost

According to (5), (13), and (14), Ψ denotes the total system cost, which is calculated as:

$$\Psi=\sum_{t=0}^{T-1}\sum_{j=1}^{J}(C_j^3+\sum_{m=1}^{M} C_{m,j}^1+C_{m,j}^2) \tag{17}$$

C. CPU, memory, and storage models in ENs

$\hat{A}_j$ denotes the maximum value of CPU cycles in each EN j. Then, the total number of CPU cycles required in EN j for computing K services of M MUs must not exceed the corresponding limit, *i.e.*,

$$\sum_{m=1}^{M}\sum_{k=1}^{K}(\lambda_{m,j}(t)\varpi_m^k(t))\leq \hat{A}_j \tag{18}$$

$\hat{G}_j$ denotes the upper limit of memory in EN j. Then, the total amount of memory consumed by K services of M MUs in EN j must not exceed the corresponding limit, *i.e.*,

$$\sum_{m=1}^{M}\sum_{k=1}^{K}(\lambda_{m,j}(t)q_m^k(t)\vartheta_m^k)\leq \hat{G}_j \tag{19}$$

where ϑ_m^k represents the memory required for each data bit of service $s_m^k(t)$.

$\hat{S}_j$ denotes the upper limit of storage in EN j. Consequently, the total size of K services of M MUs stored in EN j must not exceed its storage capacity, *i.e.*,

$$\sum_{m=1}^{M}\sum_{k=1}^{K}\left(y_j^k(t)\varsigma_m^k(t)\right)\leq \hat{S}_j \tag{20}$$

D. Latency model

The latency is determined by the delay in data uploading and processing $s_m^k(t)$ in ENs/CDC. The service request of $s_m^k(t)$ is transmitted to a specific EN or further transmitted from EN to CDC. Let $T_{m,j}^k(t)$ denote the time that MU m spends transmitting the service request of $s_m^k(t)$ and executing $s_m^k(t)$ in EN j and CDC at time slot t. Specifically, $\ddot{T}_{m,j}^k(t)$ denotes the total time required for uploading and processing $s_m^k(t)$ in EN j, and $\tilde{T}_{m,j}^k(t)$ denotes the corresponding time in the CDC, *i.e*,

$$T_{m,j}^k(t)=\ddot{T}_{m,j}^k(t)+\tilde{T}_{m,j}^k(t) \tag{21}$$

Referring to (2), $\beta_1\left(\lambda_{m,j}(t)+\lambda_{m,0}(t)\right)q_m^k(t)/R_{m,j}^U(t)$ denotes the transmission data time for service request of $s_m^k(t)$ uploaded from MU m to EN j. $\lambda_{m,j}(t)\varpi_m^k(t)/w_{m,j}^k(t)$ deontes the calculation time of the service $s_m^k(t)$ in EN j. Thus, $\ddot{T}_{m,j}^k(t)$ is calculated as:

$$\ddot{T}_{m,j}^k(t)=\frac{\beta_1\left(\lambda_{m,j}(t)+\lambda_{m,0}(t)\right)q_m^k(t)}{R_{m,j}^U(t)}+\frac{\lambda_{m,j}(t)\varpi_m^k(t)}{w_{m,j}^k(t)} \tag{22}$$

Similar to (22), $\tilde{T}_{m,j}^k(t)$ is calculated as:

$$\tilde{T}_{m,j}^k(t)=\frac{\beta_2\lambda_{m,0}(t)q_m^k(t)}{r_0}+\frac{\lambda_{m,0}(t)\varpi_m^k(t)}{w_0} \tag{23}$$

where w_0 denotes the computing speed of CDC (cycles/sec.).

T_m denotes the total completion time required for transmitting and processing K services in ENs/CDC, which is calculated as:

$$T_m=\sum_{t=0}^{T-1}\sum_{k=1}^{K}\sum_{j=1}^{J} T_{m,j}^k(t) \tag{24}$$

$\hat{T}_m$ represents the latency constraint of T_m, *i.e.*,

$$T_m\leq \hat{T}_m \tag{25}$$

E. Cost minimization problem

$\boldsymbol{\upsilon}$ is a vector of decision variables, containing $\lambda_{m,j}(t)$, $\lambda_{m,0}(t)$, $\mu_{m,j}(t)$, $P_m(t)$, $w_{m,j}^k(t)$, $y_j^k(t)$, and $z_j^k(t)$. The cost minimization problem is formulated as:

$$\arg\mathop{\mathbf{Min}}_{\boldsymbol{\upsilon}}\{\Psi=\sum_{t=0}^{T-1}\sum_{j=1}^{J}(C_j^3+\sum_{m=1}^{M} C_{m,j}^1+C_{m,j}^2)\} \tag{26}$$

subject to (1), (3), (6), (11), (12), (15), (16), (18-20).

$$\lambda_{m,j}(t)\in\{0,1\} \tag{27}$$

$$\lambda_{m,0}(t)\in\{0,1\} \tag{28}$$

$$0\leq\mu_{m,j}(t)\leq 1 \tag{29}$$

$$0\leq z_j^k(t)\leq \hat{z} \tag{30}$$

$$y_j^k(t)\in\{0,1\} \tag{31}$$

Here, $\lambda_{m,j}(t)$, $\lambda_{m,0}(t)$, and $y_j^k(t)$ are discrete integer variables, while $\mu_{m,j}(t)$, $P_m(t)$, $w_{m,j}^k(t)$, and $z_j^k(t)$ are continuous variables. It is evident that (3) and (12) are nonlinear about $\lambda_{m,j}(t)$, $\mu_{m,j}(t)$, and $w_{m,j}^k(t)$. Moreover, (2) is nonlinear with respect to $P_m(t)$, and therefore (25) is also nonlinear concerning $P_m(t)$. In sum, (26) is a constrained MINLP problem.

Theorem 1: The solution complexity of this problem is NP-hard.

Proof: In generalized edge computing, we consider the computation offloading problem in a given un-directed complete graph $G=(M_p, J_p)$, where M_p and J_p are the positions of each MU and EN. The problem offloads the computing tasks into ENs for calculation, which has been proven NP-hard [37]. This work focuses on the computation offloading problem in edge computing by determining task routing, service deployment, and cloud-edge collaborative computing. We construct a collaborative cloud-edge computing network $G'=(M_p, Q)$ from network $G=(M_p, J_p)$, where Q denotes the location

decision of task offloading, *i.e.*, $Q{=}M_p{\cup}J_p{\cup}C_p$. C_p denotes the position of the CDC. Therefore, the optimal solution to the offloading problem is also that of G. Since the task offloading problem in G is NP-hard, our proposed offloading problem is also NP-hard. Thus, no polynomial-time approaches exist for our problem [38]. To address these limitations, we employ a penalty function approach to transform each constraint into a positive penalty [39]. Afterwards, the limited problem discussed earlier is converted into an unrestricted problem, which is expressed as:

$$\mathbf{Min}_{\mathfrak{V}}\{\widetilde{\Delta}{=}\mathcal{N}\Theta{+}\Psi\} \tag{32}$$

$$\Theta{=}\sum_{p=1}^{\mathcal{N}^{\neq}}\left(\max\left\{0,-g_p(\mathfrak{V})\right\}\right)^2+\sum_{q=1}^{\mathbb{N}^{=}}\left|h_q(\mathfrak{V})\right|^2 \tag{33}$$

$$g_p(\mathfrak{V})\geq 0 \tag{34}$$

$$h_q(\mathfrak{V}){=}0 \tag{35}$$

where $\widetilde{\Delta}$ is a transformed new objective function utilized to compute the fitness value of each solution in AMGG. $\mathcal{N}$ represents a significantly big positive constant. Θ represents the total penalty for all constraints, $\mathbb{N}^{=}$ refers to the numbers of equality and $\mathcal{N}^{\neq}$ are inequality constraints respectively. To address the unconstrained problem, this work devises an innovative autoencoder-assisted evolutionary algorithm, *i.e.*, AMGG, to obtain solutions close to the optimum. The following section provides a detailed description of AMGG.

IV. PROPOSED AMGG

For the MINLP with a small number of decision variables, traditional evolutionary algorithms (EAs) such as genetic algorithms (GA) [29], particle swarm optimization (PSO) [15], grey wolf optimizer (GWO) [30], and simulated annealing (SA) [17] have shown effective solving capabilities. However, as the dimensionality and quantity of decision variables in real-world problems increase, such as the number of MUs (M), services (K), and ENs (J) in this work, the dimensionality of the search space and the complexity of solving real-world problems with EAs grow exponentially, resulting in what is commonly referred to as the curse of dimensionality [40].

To tackle the high-dimensional expensive problems (HEPs), this work introduces a two-stage high-dimensional optimization framework based on deep learning named AMGG. As shown in Fig. 2, AMGG includes two stages: autoencoder training and population coevolution process between two sub-populations.

1) In stage 1, the population is first initialized. Leveraging the superior ability of MGG to procure high-quality solution sets in high-dimensional complex problems, high-quality population data (Ω) are provided for training the autoencoder. After a predefined number of generations, the trained autoencoder ($\urcorner$) and the final population ($\mathbb{P}$) are obtained through the training of the selected individuals.

2) In stage 2, $\mathbb{P}$ is divided into two sub-populations based on fitness value, *i.e.*, $\mathbb{P}_1$ and $\mathbb{P}_2$. $\mathbb{P}_1$ directly evolves to $\mathbb{P}_1^{'}$ with MGG in the high-dimensional space. The major component of MGG is made up of Grey wolf optimizer based on Genetic learning (GG) and three different proposed strategies for achieving an appropriate balance between exploration and exploitation capabilities. Specifically, MGG is based on a multi-swarm framework, which combines the high-quality search capability of GG with three innovative strategies, namely the dynamic-subgroup number strategy (DNS), sub-population recombination strategy (SRS), and purposeful detection strategy (PDS). Meanwhile, the high-dimensional population $\mathbb{P}_2$ is further encoded into a low-dimensional population $\check{\mathbb{P}}_2$ by a well-trained autoencoder. The evolution of $\check{\mathbb{P}}_2$ through MGG yields $\check{\mathbb{P}}_2^{'}$. The low-dimensional population $\check{\mathbb{P}}_2^{'}$ is decoded into $\mathbb{P}_2^{'}$, enabling fitness evaluation in the high-dimensional space. Elite individuals from sub-populations $\mathbb{P}_1$ and $\mathbb{P}_2$ are chosen based on their fitness values and combined to form a new population $\mathbb{P}$ for the following generation's evolution.

If the termination condition is not met, $\mathbb{P}$ will undergo further division, facilitating the dynamic exchange of information between the two sub-populations. Thus, AMGG comprises two main components: high-dimensional optimization and autoencoder-assisted optimization.

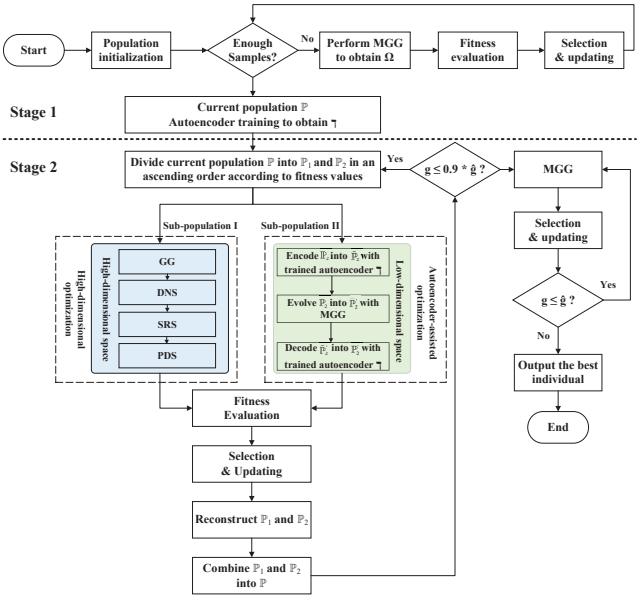


Fig. 2. Process of AMGG.

A. High-dimensional optimization

MGG adopts a typical meta-heuristic algorithm that combines GG and the multi-swarm mechanism of DNS, SRS, and PDS to solve the HEPs. The specific principles of MGG are as follows.

(1) Grey wolf optimizer based on Genetic learning (GG)

The core of GWO is its population updating mechanism, which mimics the hierarchical structure and hunting behavior of grey wolves for iterative optimization. This gives GWO advantages in convergence speed, efficiency, and precision over traditional EAs like PSO and differential evolution [30]. However, GWO can suffer from premature convergence and

local optima in complex high-dimensional problems. To address this, we integrate genetic operations from GA and the Metropolis acceptance rule from SA into GWO to enhance population diversity and search space coverage. Our approach introduces two main innovations: incorporating elite individuals to initialize a high-quality population, and optimizing the population updating strategy by using elite individuals as reference points for the three best individuals.

Each population has $|\mathbb{P}_1|$ individuals in GWO. $\boldsymbol{x}_\alpha$, $\boldsymbol{x}_\beta$, and $\boldsymbol{x}_\delta$ denote the first three best positions of $\mathbb{P}_1$. D denotes the total number of elements in each position. Here, we design a superior exemplar e_i for each individual i ($1 \le i \le |\mathbb{P}_1|$), and its d ($1 \le d \le D$) entry is $e_{i,d}$, *i.e.*,

$$e_{i,d}=\frac{c_1 \cdot r_1 \cdot x_{\alpha,d}+c_2 \cdot r_2 \cdot x_{\beta,d}+c_3 \cdot r_3 \cdot x_{\delta,d}}{c_1 \cdot r_1+c_2 \cdot r_2+c_3 \cdot r_3} \tag{36}$$

where c_1, c_2, and c_3 denote the constant of social acceleration for the effect of $\boldsymbol{x}_\alpha$, $\boldsymbol{x}_\beta$, and $\boldsymbol{x}_\delta$, and r_1, r_2, and r_3 is a random number uniformly generated in (0,1).

The four core operations developed in GG are crossover, mutation, selection, and position update for each individual.

1) Crossover operation: In traditional GA, the crossover of two selected individuals involves different segments of their chromosomes. However, in this case, the crossover operation utilizes the globally optimal individual and the second and third-best individuals in the current population. This approach aims to improve the search performance and enhance the quality of individuals. Let $\widetilde{\Delta}(\boldsymbol{\mho}_i)$ denote the value of $\widetilde{\Delta}$ for each individual i. A crossover operation is performed for each entry d of individual i. Firstly, a random individual κ ($\kappa \in \{1, 2, \ldots, |\mathbb{P}_1|\}$) is selected. Then, a crossover is applied to $\boldsymbol{x}_\alpha$, $\boldsymbol{x}_\beta$, and $\boldsymbol{x}_\delta$ to generate a new offspring $\boldsymbol{o}_i=(o_{i,1}, o_{i,2}, \ldots, o_{i,D})$, *i.e.*,

$$o_{i,d}=\begin{cases} r_d^1 \cdot x_{\alpha,d}+r_d^2 \cdot x_{\beta,d}+(1-r_d^1-r_d^2) \cdot x_{\delta,d} & \widetilde{\Delta}(\boldsymbol{x}_i)<\widetilde{\Delta}(\boldsymbol{x}_\kappa) \\ x_{\kappa,d} & \text{otherwise} \end{cases} \tag{37}$$

where r_d is a uniformly generated random number in the range (0,1). Specifically, if $\widetilde{\Delta}(\boldsymbol{x}_i)<\widetilde{\Delta}(\boldsymbol{x}_\kappa)$, $o_{i,d}$ is generated through a linear combination of $x_{\alpha,d}$, $x_{\beta,d}$, and $x_{\delta,d}$; otherwise, $o_{i,d}$ is assigned from $x_{\kappa,d}$.

2) Mutation operation: To enhance the diversity of population individuals and overcome local optimal solution constraints, thereby increasing the quality of each exemplary instance, we perform the following mutation operation on individual $\boldsymbol{o}_i$:

$$o_{i,d}=\textbf{rand}(\check{b}_d, \hat{b}_d), \text{ if } r_d<\zeta \tag{38}$$

where $\textbf{rand}(\cdot)$ denotes a rand function, $\hat{b}_d(\check{b}_d)$ is an upper (lower) limit of element d of $\boldsymbol{\mho}_i$, ζ denotes a specified mutation probability.

3) Selection operation: After obtaining the mutated individual $\boldsymbol{o}_i$, it is necessary to filter the exemplary instances in the iteration to increase the likelihood of receiving a high-quality solution set subsequently. We employ the Metropolis acceptance rule from the SA algorithm as the filtering criterion. Υ_0 denotes the starting temperature. Φ denotes the cooling rate. The specific formula is as follows:

$$\boldsymbol{e}_i=\begin{cases} \boldsymbol{o}_i, & \widetilde{\Delta}\boldsymbol{o}_i)<\widetilde{\Delta}(\boldsymbol{e}_i), \\ \boldsymbol{o}_i, & \widetilde{\Delta}(\boldsymbol{o}_i) \ge \widetilde{\Delta}(\boldsymbol{e}_i) \text{ and } \exp^{(-\frac{\boldsymbol{o}_i-\boldsymbol{e}_i}{\Upsilon_g})}>\aleph, \\ \boldsymbol{e}_i, & \widetilde{\Delta}(\boldsymbol{o}_i) \ge \widetilde{\Delta}(\boldsymbol{e}_i) \text{ and } \exp^{(-\frac{\boldsymbol{o}_i-\boldsymbol{e}_i}{\Upsilon_g})} \le \aleph. \end{cases} \tag{39}$$

where Υ_g denotes the current temperature value of each iteration g, and $\aleph$ is a random number between 0 and 1.

4) Position update of each individual: In the original GWO, the parameter A is crucial for controlling the hunting behavior of the wolf population. The convergence factor a, which linearly decreases from 2 to 0, directly influences A. However, for complex problems, this linear strategy often results in insufficient search space exploration. We aim to maintain a larger a value in the early stages for thorough global exploration, and a smaller a value in the later stages to promote local exploitation and faster convergence. Thus, we propose a non-linear convergence factor a, defined as follows:

$$a=1+\cos(\frac{\pi \cdot g}{\hat{g}_2}) \tag{40}$$

where $\hat{g}_2$ represents the maximum number of iterations of AMGG in stage 2.

Let $\boldsymbol{v}_\alpha$, $\boldsymbol{v}_\beta$, and $\boldsymbol{v}_\delta$ denote the distances between $\boldsymbol{x}_\alpha$, $\boldsymbol{x}_\beta$, and $\boldsymbol{x}_\delta$ and other individuals, respectively. Then, $\boldsymbol{x}$ indicates the position vector of an individual, and it is updated as:

$$\boldsymbol{v}_\alpha=|\boldsymbol{h}_1 \cdot \boldsymbol{x}_\alpha-\boldsymbol{e}|, \boldsymbol{v}_\beta=|\boldsymbol{h}_2 \cdot \boldsymbol{x}_\beta-\boldsymbol{e}|, \boldsymbol{v}_\delta=|\boldsymbol{h}_3 \cdot \boldsymbol{x}_\delta-\boldsymbol{e}| \tag{41}$$

$$\boldsymbol{x}_1=\boldsymbol{x}_\alpha-\boldsymbol{a}_1 \cdot \boldsymbol{v}_\alpha, \boldsymbol{x}_2=\boldsymbol{x}_\beta-\boldsymbol{a}_2 \cdot \boldsymbol{v}_\beta, \boldsymbol{x}_3=\boldsymbol{x}_\delta-\boldsymbol{a}_3 \cdot \boldsymbol{v}_\delta \tag{42}$$

$$\boldsymbol{x}(g+1)=\frac{\iota(\boldsymbol{x}_1+\boldsymbol{x}_2+\boldsymbol{x}_3)}{3} \tag{43}$$

where $\iota=(\hat{\iota}-\check{\iota}) \cdot \frac{g}{\hat{g}_2}+\check{\iota}$, $\boldsymbol{h}$ is coefficient vector in [0,2], $\boldsymbol{x}(g+1)$ represents the updated position of the next generation's search individual. ι ensures individuals' strong social learning ability in the early stages and exploration ability towards the global optimum position, enhancing search space coverage. In later stages, it tends to search near the α individual to accelerate convergence. GG is described as Algorithm 1.

(2) Multi-swarm mechanism of DNS

The purpose of DNS is to stabilize the sub-population adjustment process. It determines and adjusts the number of sub-populations in ascending order during evolution. This ensures efficient information transmission and selection among populations, meeting exploitation requirements. DNS faces two key issues: determining the number of sub-populations within each population, and deciding when to adjust these numbers. Regarding the first issue, we assume the total population size to be U, and $\mathbb{U}$ represents a decreasing ordered sequence, *i.e.*, $\mathbb{U}=\{u_1, u_2, \cdots, u_v\}$, where $u_1>u_2>\cdots>u_v$. u_v in $\mathbb{U}$ represent the number of sub-populations. In this work, the individual counts in each sub-population are identical. For instance, if $U=60$, $\mathbb{U}=\{30, 20, 15, 12, 10, 6, 5, 4, 3, 2, 1\}$, $\varrho^\diamond$ denotes the individual count in each sub-population, which is 2 when $u_1=30$ at the start of the iterations. At the end of the iterations, all sub-populations merge into the entire population. Concerning the second issue, we adjust the number of sub-populations after every χ fitness evaluation and $\chi=\frac{\hat{g}_2}{|\mathbb{U}|}$, $|\mathbb{U}|$

Algorithm 1: GG

Input: population for high-dimensional optimization ($\mathbb{P}_1$), objective function ($\widetilde{\Delta}$), maximum number of iterations of AMGG in stage 2 (g_2)
Output: final population ($\mathbb{P}_1$)

Obtain the fitness value of each individual i with (32);
Update a global optimum of current population ($\boldsymbol{x}_\alpha$);
Initialize ζ of GA, Υ_0 and Φ of SA, $\boldsymbol{a}$ and $\boldsymbol{h}$ of GWO, parameters of GG including c_1, c_2, c_3, and ι;
Initialize superior exemplars ($\boldsymbol{e}$) with (36);
$g \leftarrow 1$;
while $g \leq \hat{g}_2$ **do**
  Perform the crossover operation of GA with (37) to obtain $\boldsymbol{o}_i$;
  Perform the mutation operation of GA with (38) on $\boldsymbol{o}_i$;
  $\Upsilon_g \leftarrow \Upsilon_{g-1}\Phi$;
  Perform the selection operation of SA with (39) to update $\boldsymbol{e}_i$;
  Update $\boldsymbol{a}$ with (40);
  $\iota = (\hat{\iota} - \check{\iota}) \cdot \frac{g}{\hat{g}_2} + \check{\iota}$;
  Update $\boldsymbol{x}(g+1)$ of GWO with (41), (42), and (43);
  Calculate the fitness value of each individual i with (32);
  Update $\boldsymbol{x}_\alpha$;
  $g \leftarrow g+1$;
end

represents the number of elements in $\mathbb{U}$, which in this example is $\mathbb{U}=11$. DNS is described in Algorithm 2.

Algorithm 2: DNS

Input: frequency of adjustment of sub-populations (χ), total population size (U), decreasing ordered sequence ($\mathbb{U}$), $v=1$, and population for high-dimensional optimization ($\mathbb{P}_1$)
Output: u_v sub-populations of size $\varrho^\diamond$

if $\mathrm{mod}(g,\chi)==0$ *and* $v \leq |\mathbb{U}|$ **then**
  $u_v \in \mathbb{U}$;
  $\varrho^\diamond = \frac{U}{u_v}$;
  Randomly divide the whole population $\mathbb{P}_1$ into u_v sub-populations;
  $v = v+1$;
end

(3) Multi-swarm mechanism of SRS

SRS aims to facilitate sharing valuable information among sub-populations, ensuring timely recombination of high-quality data across multiple populations and enhancing development during the search process. Since the neighbor topology of each sub-population in SRS is modeled as a ring, multiple iterations are required to thoroughly extract useful information from other individuals. To address this issue, we introduce $\hbar$ to represent the number of consecutive stagnation generations at the global optimum ($\boldsymbol{x}_\alpha$). We utilize it as the criterion for sub-population recombination in the population and promptly recombine the entire population into u_v sub-populations. Specifically, we set $\hbar = \left\lfloor \frac{\varrho^\diamond}{2} \right\rfloor$ as the recombination threshold. For instance, if $\varrho^\diamond=10$, and $\hbar$ exceeds 5, the population is reorganized. The details of SRS can be represented as Algorithm 3 based on the information above.

Algorithm 3: SRS

Input: u_v sub-populations of size $\varrho^\diamond$, the number of consecutive stagnation generations at the global optimum ($\hbar$)
Output: u_v sub-populations of size $\varrho^\diamond$, the number of consecutive stagnation generations at the global optimum ($\hbar$)

if $\hbar \geq \frac{\varrho^\diamond}{2}$ **then**
  Randomly regroup the whole population into u_v sub-populations;
  $\hbar = 0$;
end

(4) Multi-swarm mechanism of PDS

PDS aims to enhance global exploration, helping populations escape local optima. It uses historical information about individuals to guide targeted detection operators for the global optimal individual ($\boldsymbol{x}_\alpha$), aiding in overcoming local optima. PDS addresses three issues: selecting relevant information, determining when to execute detection operations, and performing these operations. Each search space dimension is partitioned into S segments for the first issue. Here, ξ_s^d represents the segment s in the dimension d of the search space. We employ $\mathcal{M}_s^d$ ($1 \leq s \leq S$, $1 \leq d \leq D$) to denote the frequency of elite individuals fall into segment s of dimension d, serving as an evaluation criterion for the segment. In other words, the more frequently elite individuals fall into a segment, the more useful information ξ_s^d possesses. x_*^d denotes the dimension d of elite individuals ($\boldsymbol{x}_\alpha$, $\boldsymbol{x}_\beta$, and $\boldsymbol{x}_\delta$). Therefore, $\mathcal{M}_s^d$ can be obtained as follows:

$$\mathcal{M}_s^d = \mathcal{M}_s^d + 1, \text{if } x_*^d \text{ lies in } \xi_s^d \tag{44}$$

Regarding the second issue, we conduct periodic detection operations due to the varying characteristics of populations across different evolutionary stages. Specifically, based on $\mathcal{M}_s^d$, which is utilized to aid elite individuals in locating the most promising solutions in the population, early evolutionary stages with fewer individuals promote more frequent execution of detection operations, facilitating enhanced exploration. However, detection operations almost ceased in later evolutionary stages when the population size is larger, favoring exploitation. Regarding the third issue, we propose a simple detection operation, which involves calculating the $\mathcal{M}_s^d$ of x_*^d. We randomly select positions of individuals that do not frequently detect segments and only replace the position of the individual being detected by x_*^d when the performance of the generated new individual position improves over the original x_*^d. To avoid detecting the same segment s for x_*^d at different periods, thereby avoiding interference with exploration, we introduce a flag denoted by $\mathcal{F}_s^d$. Specifically, if segment s has been detected by x_*^d, $\mathcal{F}_s^d$ is set to 1; otherwise, it is set to 0.

x_*^d cannot detect segments set to 1 until they are reset to 0. When all flags are set to 1, they are reset to 0. In summary, PDS is described as follows:

Algorithm 4: PDS

Input: global optimum of current population ($\boldsymbol{x}_\alpha$), frequency of elite individual falling into segment s of dimension d ($\mathcal{M}_s^d$), and flag bit ($\mathcal{F}_s^d$), objective function ($\widetilde{\Delta}$)
Output: global optimum of current population ($\boldsymbol{x}_\alpha$), flag bit ($\mathcal{F}_s^d$)

for $d \leftarrow 1$ **to** D **do**
  if $x_*^d \in \{\xi_s^d \mid \mathcal{M}_s^d$ *is larger than* $\mathcal{M}_k^d (1 \leq k \leq S,\ k \neq s)\}$ **then**
    x_*^d is replaced by a random value in a less visited segment ξ_k^d where $\mathcal{F}_k^d == 0$;
    if $\widetilde{\Delta}(\boldsymbol{x}_\alpha) > \widetilde{\Delta}(\boldsymbol{x}_*)$ **then**
      $\boldsymbol{x}_\alpha \leftarrow \boldsymbol{x}_*$;
    end
    $\mathcal{F}_k^d \leftarrow 1$;
    if $\forall\ \mathcal{F}_k^d == 1$ **then**
      $\mathcal{F}_k^d \leftarrow 0$;
    end
  end
end

(5) Framework of MGG

By merging the Algorithm 1, 2, 3, and 4, the detail of MGG is described in Algorithm 5. Moreover, the multi-swarm mechanism offers an inclusive cooperation mechanism for all sub-populations, allowing MGG to maximize the benefits of each sub-population.

Algorithm 5: MGG

Input: χ, U, $\mathbb{U}$, v=1, $\hbar$, $\mathcal{M}_s^d$, $\mathcal{F}_s^d$, $\widetilde{\Delta}$, $\mathbb{P}_1$
Output: final population ($\mathbb{P}_1$)

1 Partition the search space into S equally sized sub-regions;
2 $\mathcal{F}_s^d \leftarrow 0$;
3 $\mathcal{M}_s^d \leftarrow 0$;
4 **repeat**
5 $\mathbb{P}_1$=**GG**($\mathbb{P}_1$);
6 Perform **DNS**;
7 Update $\hbar$;
8 Perform **SRS**;
9 Update $\mathcal{M}_s^d$ with (44);
10 Perform **PDS**;
11 Update $\mathcal{F}_s^d$, $\boldsymbol{x}_\alpha$;
12 **until** *Combine all sub-populations into* $\mathbb{P}_1$;

B. Autoencoder-assisted optimization

Fig. 3 illustrates the autoencoder-assisted optimization process. First, the high-dimensional population is compressed into a low-dimensional space *via* an encoder, which includes the optimal individual and the search space bounds for each individual. In the low-dimensional space, GG's operations, such as crossover, mutation, and update, along with population strategies, including DNS, SRS, and PDS, are performed to generate high-quality solutions based on previously learned population information. A crucial aspect of this process is evaluating the fitness of low-dimensional individuals. This requires reconstructing them into the high-dimensional space *via* a decoder for meaningful computation. Finally, high-quality solutions generated in the low-dimensional space are reconstructed and combined into a high-dimensional population.

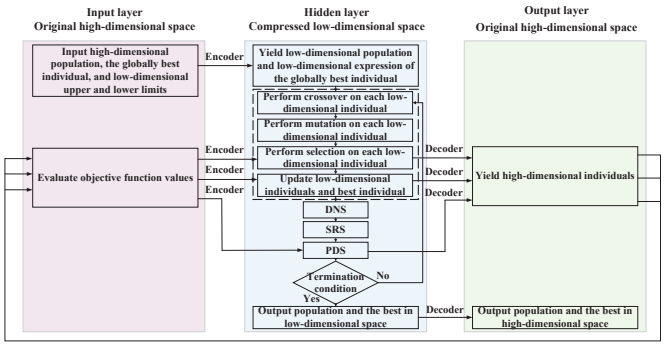


Fig. 3. Process of Autoencoder-assisted Optimization.

C. AMGG

(1) Stage 1: Autoencoder training

The details of AMGG are described in Algorithm 6. Training the autoencoder is to explore the search space of high-dimensional complex problems and capture the relationships between decision variables. In this stage, the initialized data samples undergo several generations of evolution through MGG. As MGG searches, population individuals progress towards better solutions, and a well-trained autoencoder is increasingly likely to learn compressed representations of regions closer to the optimum. Line 1 initializes the parameters of MGG. Line 2 randomly initializes the population positions of MGG to obtain the population $\mathbb{P}$. Line 3 executes MGG $\hat{g}_1$ counts to obtain a new population $\mathbb{P}$ and utilizes it as initialized data samples Ω. Line 4 selects and updates the latest population $\mathbb{P}$ and data samples Ω by evaluation of population individuals $\boldsymbol{x}_i$ utilizing $\widetilde{\Delta}(\boldsymbol{x}_i)$. Line 5 trains the autoencoder utilizing the data samples Ω and obtains the autoencoder $\urcorner$. The **for** loop terminates if $g \leq 0.9 \times \hat{g}_2$ holds in Line 6.

(2) Stage 2: Population coevolution

1) Population splitting: To enhance the solving capability for high-dimensional complex problems and avoid falling into local optima, the current population of AMGG is divided into two sub-populations, which evolve in a distributed parallel manner. Line 7 of AMGG performs the population split to divide $\mathbb{P}$ into $\mathbb{P}_1$ and $\mathbb{P}_2$. Individuals in $\mathbb{P}$ are arranged in ascending order based on their fitness values. The better fitness $|\mathbb{P}_1|$ individuals are put into $\mathbb{P}_1$, and the worse fitness $|\mathbb{P}_2|$ individuals are put into $\mathbb{P}_2$.

Algorithm 6: AMGG

Input: maximum iteration count of AMGG in stage 1 ($\hat{g}_1$) and stage 2 ($\hat{g}_2$), database to train autoencoder (Ω), objective function ($\widetilde{\Delta}$)

Output: global optimum of current population ($\boldsymbol{x}_\alpha$)

1 Initialize the parameters of MGG;
2 Initialize the positions of individuals to obtain $\mathbb{P}$ randomly;
3 Perform MGG for $\hat{g}_1$ counts to obtain Ω;
4 Evaluate $\widetilde{\Delta}(\boldsymbol{x}_i)$ of each $\boldsymbol{x}_i$ to select and update $\mathbb{P}$;
5 Train an autoencoder with a training set Ω to obtain the autoencoder $\urcorner$;
6 **for** $g \leftarrow 1$ **to** $0.9\times\hat{g}_2$ **do**
7 Perform population split to $\mathbb{P}_1$ and $\mathbb{P}_2$;
8 $\mathbb{P}_1^{'} \leftarrow$ **MGG**$(\mathbb{P}_1)$;
9 Update $\widetilde{\Delta}(\boldsymbol{x}_i)$ of each $\boldsymbol{x}_i$ in $\mathbb{P}_1^{'}$;
10 Select the best $|\mathbb{P}_1|$ individuals from $\mathbb{P}_1$ and $\mathbb{P}_1^{'}$ for constructing $\mathbb{P}_1$;
11 $\check{\mathbb{P}}_2 \leftarrow$ **encode**$(\urcorner, \mathbb{P}_2)$;
12 $\check{\mathbb{P}}_2^{'} \leftarrow$ **MGG**$(\check{\mathbb{P}}_2)$;
13 $\mathbb{P}_2^{'} \leftarrow$ **decode**$(\urcorner, \check{\mathbb{P}}_2^{'})$;
14 Update $\widetilde{\Delta}(\boldsymbol{x}_i)$ of each $\boldsymbol{x}_i$ in $\mathbb{P}_2^{'}$;
15 Select the best $|\mathbb{P}_2|$ individuals from $\mathbb{P}_2$ and $\mathbb{P}_2^{'}$ for constructing $\mathbb{P}_2$;
16 $\mathbb{P} \leftarrow \mathbb{P}_1 \cup \mathbb{P}_2$;
17 **end**
18 **for** $g \leftarrow 0.9\times\hat{g}_2$ **to** $\hat{g}_2$ **do**
19 $\mathbb{P} \leftarrow$ **MGG**$(\mathbb{P})$;
20 **end**
21 Select $\boldsymbol{x}_\alpha$ from $\mathbb{P}$, and update its fitness value $\widetilde{\Delta}_\alpha$;

2) High-dimensional population coevolution: $\mathbb{P}_1$ conducts the evolution process of the high-dimensional population, leveraging MGG's powerful search capability to identify optimal solutions in the vicinity of the present solutions. $\mathbb{P}_1$ evolves into $\mathbb{P}_1^{'}$ with MGG in Line 8. This evolution process in the second stage continues the same procedure as in the first. Line 9 updates $\widetilde{\Delta}(\boldsymbol{x}_i)$ of each $\boldsymbol{x}_i$ in $\mathbb{P}_1^{'}$. Line 10 selects the best $|\mathbb{P}_1|$ individuals from $\mathbb{P}_1$ and $\mathbb{P}_1^{'}$ for constructing $\mathbb{P}_1$.

3) Autoencoder-assisted population coevolution: $\mathbb{P}_2$ executes the autoencoder-assisted population evolution process to capture the distribution characteristics of its search space. It compresses individuals from low-quality populations into a low-dimensional space and generates a new set of solutions based on the features learned from high-quality solutions, thus more effectively guiding its evolution. Line 11 compresses each individual in $\mathbb{P}_2$ into its low-dimensional one in $\check{\mathbb{P}}_2$ through **encode**$(\cdot)$ obtained by autoencoder $\urcorner$. Then, $\check{\mathbb{P}}_2$ evolves with a proposed MGG in Line 12 to avoid stagnation and yield $\check{\mathbb{P}}_2^{'}$. Because of the dimensionality reduction, the original fitness function cannot be applied directly to the evaluation of $\check{\mathbb{P}}_2^{'}$. Thus, Line 13 decodes $\check{\mathbb{P}}_2^{'}$ through **decode**$(\cdot)$ obtained by autoencoder $\urcorner$ into high-dimensional $\mathbb{P}_2^{'}$ in the original search space with $\urcorner$. Line 14 updates $\widetilde{\Delta}(\boldsymbol{x}_i)$ of each $\boldsymbol{x}_i$ in $\mathbb{P}_2^{'}$. Line 15 selects the best $|\mathbb{P}_2|$ individuals from $\mathbb{P}_2$ and $\mathbb{P}_2^{'}$ for constructing $\mathbb{P}_2$.

4) Information exchange: After each iteration of the two sub-populations $\mathbb{P}_1$ and $\mathbb{P}_2$, the populations are recombined into a new population $\mathbb{P}$ at Line 16, allowing the dynamic exchange of information between them. To enhance the coverage of the search space by AMGG in the later iterations, Lines 18–20 utilizes MGG on the entire population $\mathbb{P}$ for $0.1\times\hat{g}_2$ counts to generate more promising offsprings. Line 21 selects $\boldsymbol{x}_\alpha$ from $\mathbb{P}$, and updates its $\widetilde{\Delta}_\alpha$. Finally, output the best solution $\boldsymbol{x}_\alpha$.

The complexity analysis of AMGG is given as follows. In stage 1, as shown in Section IV-A, the complexity of MGG in each iteration is $\mathcal{O}(|\mathbb{P}|D)$. According to $\mho$ in Section III-E, $D = M(2J + JK + 2) + 2JK$, and the complexity of each iteration is $\mathcal{O}|\mathbb{P}|(M(2J + JK + 2) + 2JK))$ in MGG. Thus, the complexity of MGG is $\mathcal{O}(\hat{g}_1|\mathbb{P}|(M(2J + JK + 2) + 2JK))$. The complexity of training autoencoder is $\mathcal{O}((M(2J + JK + 2) + 2JK)|\Omega|\hbar)$. $\hbar$ denotes the number of epochs in the autoencoder training, and $|\Omega|$ denotes the size of the training set Ω. In stage 2, the complexity of the high-dimensional optimization and autoencoder-assisted optimization is $\mathcal{O}(\hat{g}_2|\mathbb{P}|(M(2J + JK + 2) + 2JK))$. To sum up, the time complexity of AMGG is $\mathcal{O}((\hat{g}_1|\mathbb{P}| + |\Omega|\hbar + \hat{g}_2|\mathbb{P}|)(M(2J+JK+2)+2JK))$. In Algorithm 6, the **for** loop, which terminates after $\hat{g}_2$ iterations, contributes significantly to the execution overhead. As shown in Lines 7–16, the complexity of AMGG is $\mathcal{O}(|\mathbb{P}|(M(2J + JK + 2) + 2JK))$ in each iteration. Consequently, AMGG has a complexity of $\mathcal{O}(\hat{g}_2|\mathbb{P}|(M(2J + JK + 2) + 2JK))$. In summary, AMGG effectively mitigates the curse of dimensionality and tackles high-dimensional MINLP challenges.

V. Performance Evaluation

This work employs real-world data obtained from Google cluster[1] to simulate the proposed scenario model to validate the optimization performance of AMGG. AMGG is implemented and coded using MATLAB 2023a, and executed on a server equipped with an Intel(R) Xeon(TM) Gold 6248R CPU operating at 3.0 GHz, DDR4 REGS 2933 memory with a capacity of 8 × 32 GB, and two NVIDIA RTX 3090 24G graphics cards. MATLAB's Deep Learning Toolbox serves as the autoencoder's framework tool. This work can also adopt Pytorch as the framework of our autoencoder module. The autoencoder architecture consists of three layers of fully connected neural networks, comprising an input layer, a hidden layer, and an output layer. Additionally, the number of neurons in the hidden layer is half that of the input and output layers. The loss function for training the autoencoder utilizes the mean squared error. ReLU is employed as the activation function for the autoencoder neural network. The training data are collected from high-quality solutions generated during the evolutionary process of AMGG.

A. Experimental settings

This work considers a network topology comprising multiple MUs, ENs, and a CDC [41]. We set the number of

[1]https://github.com/google/cluster-data

MUs (M) in [1, 20] and the number of ENs (J) from 2 to 4, assuming three microservices per application, *i.e.*, K=3. Considering the practical scenario of MU mobility and the random deployment of ENs in our architecture, MUs are randomly distributed and continually move during experiments. We assume that each MU remains in the coverage range of at least one EN in a time slot and does not transition between the coverage ranges of different ENs in a time slot. MUs generate tasks randomly. Each time slot (t) in our work is set to a duration of 5 minutes, resulting in 288 time slots (T) per 24 hours [15]. Furthermore, to realistically simulate service migration during MU mobility, we employ a simulator [42] to generate MU movement trajectories. Specifically, 70% of MUs are simulated using a map-based mobility model, while 30% use a random walk mobility model [7]. Parameters for the actual model are derived from [41] as outlined in Tables III and IV, while parameters for AMGG are set as detailed in Table V according to [43]. We design two benchmark test scenarios to evaluate AMGG: one focusing on final optimization performance and the other concentrating on service migration strategies.

B. Experimental results for benchmark 1

To evaluate AMGG, this work compares it with four state-of-the-art algorithms, *i.e.*, SBAGO, GSP, HGGWO, and SAPSO. The reasons for selecting these algorithms as comparative counterparts are given as follows.

1) SBAGO [14]. It combines the genetic operations of GA extensively with the advanced search information of the bat algorithm, aiming to generate diverse and high-quality new exemplars of individuals, thereby enhancing the accuracy of the search process.
2) GSP [15]. It integrates the genetic operations of GA and the Metropolis acceptance criterion of the SA algorithm into PSO, aiming to enhance global search ability.
3) HGGWO [16]. It incorporates the dynamic crossover and mutation ratios from GA into the traditional GWO, aiming to address the scalability selection issue of GWO and enhance the accuracy of the search process.
4) SAPSO [17]. It integrates the Metropolis acceptance criterion from the SA algorithm into PSO, aiming to select elite particles to enhance the search efficiency and diversity.

To comprehensively evaluate the optimization performance of AMGG, we have selected multiple performance metrics for experimental validation, including system cost and penalty value for unsatisfied constraints.

(1) System cost: For a hybrid cloud-edge system, a key performance metric is the system cost of serving requests at ENs and CDC. System cost is characterized as the services communication, computation, and migration cost, which is crucial for the deployment and operation of service schedulers for mobile operators, facilitating the provisioning of cloud-edge services.

Fig. 4 presents the comparative results of the total cost for AMGG, SBAGO, GSP, HGGWO, and SAPSO at different time slots. It is evident from the figure that, under identical

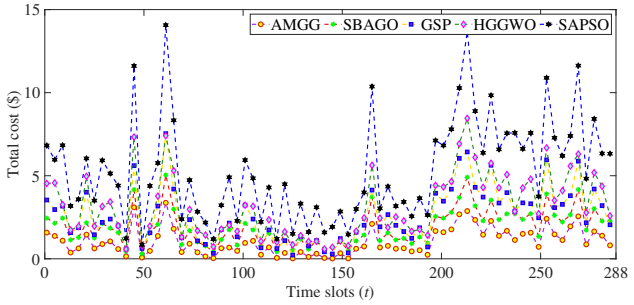


Fig. 4. Total cost ($) of AMGG, SBAGO, GSP, HGGWO, and SAPSO in terms of varying t.

conditions q_m^k for all algorithms in each time slot, SAPSO consistently exhibits higher total cost compared to the other algorithms, indicating its inferior optimization performance. The reason lies in SAPSO's tendency to converge to local optima in the optimization space and its limited capability to escape from such local optima, thereby failing to obtain a high-quality solution set in the high-dimensional decision space of the dynamic migration environment. HGGWO and GSP suffer from similar issues. Furthermore, the results in the graph indicate that although SBAGO generally yields lower total cost compared to GSP, HGGWO, and SAPSO, it still surpasses AMGG in most time slots. This is primarily due to SBAGO's deficiencies in population update mechanisms and its ability to select global optima in high-dimensional spaces despite possessing mechanisms for escaping local optima. In contrast, AMGG consistently outperforms all other algorithms in each time slot. Specifically, compared to SAPSO, HGGWO, GSP, and SBAGO, AMGG achieves average reductions in the total cost of 66.04%, 56.02%, 45.31%, and 31.42%, respectively. This superiority of AMGG is attributed to its combined approach, which incorporates an autoencoder-assisted optimization phase for extracting useful information and critical features from the population, as well as integrating genetic operations and SA conditional acceptance rules into the MGG optimization phase of the GWO in a multi-swarm and multi-strategy manner. This approach allows for better handling of high-dimensional complex problems and exhibits a stronger trade-off between local exploration and global search capabilities.

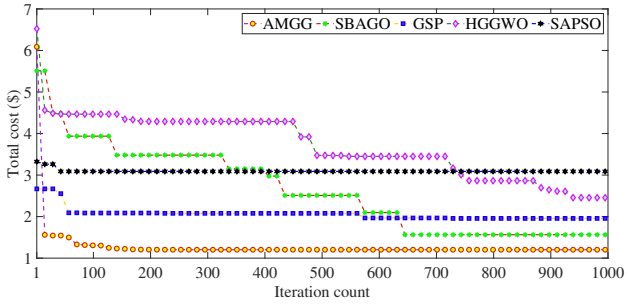


Fig. 5. Evolutionary curves of total cost in each iteration of AMGG, SBAGO, GSP, HGGWO, and SAPSO, respectively.

TABLE III
PARAMETER SETTING-PART 1

v	h_1	G_0	W_j^U	P_m^0 (P_δ^1)	ρ_m	β_1 (β_2)	ϵ (ν)	$\hat{P}_m$	σ_j
4	0.98	1.6×10^{-11}	[10,20] MHz	0.2 W (0.001 W)	[16,20]	1	1.16×10^{-10} (0.5×10^{-10})	0.1 W $	[0.8×10^{-25},1.3×10^{-25}]

TABLE IV
PARAMETER SETTING-PART 2

e_0	r_0	$\hat{E}_j$	$\tilde{F}_j$	τ	$\hat{z}$	$\tilde{C}$	$\hat{A}_j$	$\hat{G}_j$	ϑ_m^k	$\hat{S}_j$	w_0
1 W/GHz	1 Gbps	15 J	5×10^6 cycles/sec.	2.44×10^{-4} $/J	[0,1] $	45 $	2×10^{10} cycles	2 GB	[50,100]	[5,300] GB	7×10^{14} cycles/sec.

TABLE V
PARAMETER SETTING-PART 3

Ω	$\lvert\mathbb{P}\rvert$	$\lvert\mathbb{P}_1\rvert$ ($\lvert\mathbb{P}_2\rvert$)	$\mathbb{U}$	$c_1(c_2,c_3)$	ζ	Υ_0	Φ	$\hat{g}_1$	S	$\hat{g}_2$	$\hbar$
100	120	60 (60)	$\{30,20,15,12,10,6,5,4,3,2,1\}$	0.5	0.04	10^8	0.95	1000	10	1000	300

Fig. 5 presents the evolution curves of total cost over 1000 iterations for AMGG, SBAGO, GSP, HGGWO, and SAPSO when M=10, J=4, K=3, and $D = 244$. The figure shows that although SAPSO converges to the final solution by the 43-rd iteration, its total cost is significantly higher than that of the other four algorithms, indicating the poorest convergence result. Even though HGGWO achieves a final converged total cost 20.50% lower than SAPSO, it takes until the 939-th iteration to converge. GSP reaches convergence at the 575-th iteration, while SBAGO achieves convergence at the 645-th iteration, with their final converged total cost being 61.37% and 38.46% higher than AMGG, respectively. However, AMGG converges to a lower total cost value than the other four algorithms in only 225 iterations. Experimental results demonstrate AMGG's superior ability to explore high-dimensional spaces and obtain high-quality solution sets and its excellent search accuracy during the evolution phase. MGG ensures excellent global search capability for AMGG when the dimensionality is low. Furthermore, the autoencoder enhances AMGG's optimization capability when dealing with higher-dimensional optimization problems.

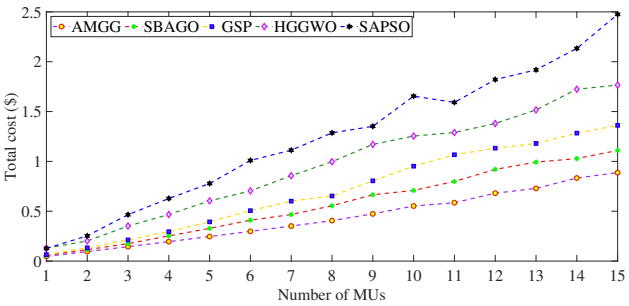


Fig. 6. Total cost ($) of AMGG, SBAGO, GSP, HGGWO, and SAPSO in terms of varying M.

Fig. 6 compares the total cost with AMGG, SBAGO, GSP, HGGWO, and SAPSO for varying MUs (M). The results reveal that as the number of MUs increases, so does the dimensionality of decision variables. Among the algorithms considered, AMGG demonstrates the least significant impact on the escalating number of decision variables while achieving the lowest total cost in optimization outcomes. Specifically, the total cost of optimization results with AMGG is, on average, 20.02%, 31.72%, 46.35%, and 62.09% lower than those with SBAGO, GSP, HGGWO, and SAPSO, respectively. Furthermore, it should be mentioned that when M increases, the total cost decrease of AMGG becomes more noticeable compared to the other four algorithms. The problem dimension grows as M increases. As a result, AMGG's overall cost is lower than that of the other four algorithms, and its ability to solve high-dimensional problems becomes more apparent. This notable advantage can be primarily attributed to AMGG's balanced exploration-exploitation strategy during optimization. In this strategy, two subgroups evolve in a distributed manner. One subgroup employs an autoencoder to compress the high-dimensional landscape into an informative low-dimensional space, thereby facilitating effective global search operations by MGG in the low-dimensional space to drive the population toward optimal solutions. Simultaneously, the other subgroup undergoes MGG's multi-swarm and multi-strategy evolution process. The dynamic exchange of information between these two subgroups fosters both high-dimensional local exploration and low-dimensional global search.

Fig. 7 illustrates the results of optimizing hybrid cloud-edge system cost with AMGG, SBAGO, GSP, HGGWO, and SAPSO under different maximum completion time constraints. As shown in Fig. 7, AMGG significantly outperforms the other four algorithms regarding cost optimization. On average, the optimized cost of AMGG is reduced by 13.56%, 25.00%, 34.62%, and 41.71% compared to SBAGO, GSP, HGGWO, and SAPSO, respectively. Additionally, the optimization trend of AMGG is more pronounced compared to other algorithms, highlighting its superior optimization performance. AMGG consistently identifies the optimal service migration strategy in the hybrid system, maintaining strict latency boundaries for services. These results demonstrate the feasibility of AMGG

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3457488

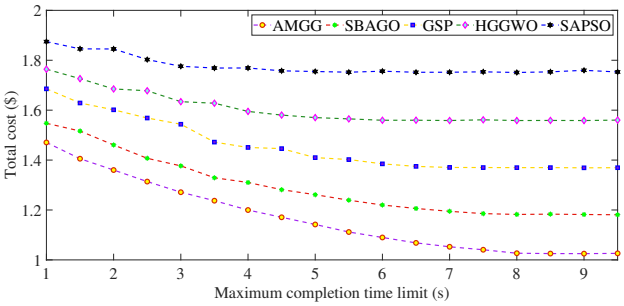


Fig. 7. Total cost ($) of AMGG, SBAGO, GSP, HGGWO, and SAPSO in terms of varying $\hat{T}_m$.

in balancing QoS and cost. This is attributed to AMGG's utilization of a two-stage evolutionary framework aided by an autoencoder to enhance the capability of evolutionary algorithms in solving high-dimensional optimization problems. Furthermore, AMGG employs a multi-swarm optimization framework, MGG, which combines hybrid GWO, *i.e.*, GG, and three dynamic optimization strategies to improve the ability of evolutionary algorithms to achieve the exploration and exploitation balance for enhanced search accuracy.

Table VI presents the ablation studies demonstrating the impact of the two key components of AMGG on optimizing the system cost. Each component is independently executed 50 times to obtain statistical results regarding the best, average, standard deviation, and p values. Furthermore, a logical p-value of 1 indicates that AMGG significantly outperforms the comparative components. The addition of each component effectively improves the optimization performance.

TABLE VI
ABLATION STUDIES OF AMGG WITH TWO COMPONENTS

Components	Best	Avg.	Std.	p value
w/o high-dimensional optimization	1.3791	1.6034	1.4757	1
w/o autoencoder-assisted optimization	1.6876	1.8526	1.9267	1
AMGG	**1.0468**	**1.2572**	**1.0693**	N/A

To validate and compare the efficiency and effectiveness of AMGG in addressing HEPs, AMGG is compared with a recent state-of-the-art approach that uses deep learning, *i.e.*, genetic simulated annealing-based particle swarm optimizer with autoencoders (GSPAE) [44]. The experimental setup follows the configuration in Table VI. Table VII presents the comparison results between GSPAE and AMGG. The experiment demonstrates that AMGG achieves a 13.22% improvement in the system cost compared to GSPAE.

TABLE VII
COMPARISON BETWEEN GSPAE AND AMGG

Methods	Best	Avg.	Std.	p value
GSPAE	1.1749	1.2856	1.5452	1
AMGG	**1.0035**	**1.1792**	**1.3409**	N/A

(2) Penalty: In addition, penalty value is a key performance metric for the service scheduler to operate the hybrid cloud-edge system. Thus, we utilize it to evaluate whether different algorithms strictly satisfy the constraints in the formulated high-dimensional optimization problem.

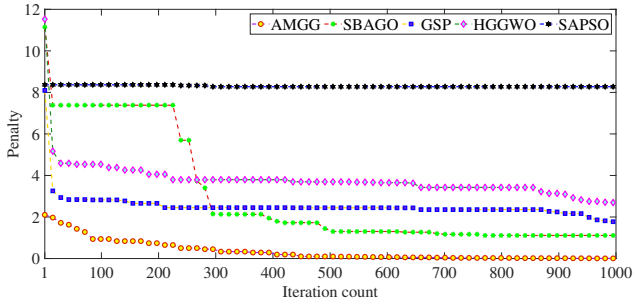


Fig. 8. Evolutionary curves of penalty in each iteration of AMGG, SBAGO, GSP, HGGWO, and SAPSO, respectively.

Fig. 8 presents the penalty evolution curves over 1000 iterations for the algorithms AMGG, SBAGO, GSP, HGGWO, and SAPSO, with M=10, J=4, K=3, and D=244. It can be observed from the figure that, except AMGG, the penalty values of the other four algorithms do not approach zero as the iterations progress. Indeed, these results indicate that the other four peers are unsuccessful in addressing high-dimensional optimization problems and satisfying the constraints. Overall, Fig. 8 demonstrates that AMGG achieves the best results for high-dimensional problems compared with other peers.

C. Experimental results for benchmark 2

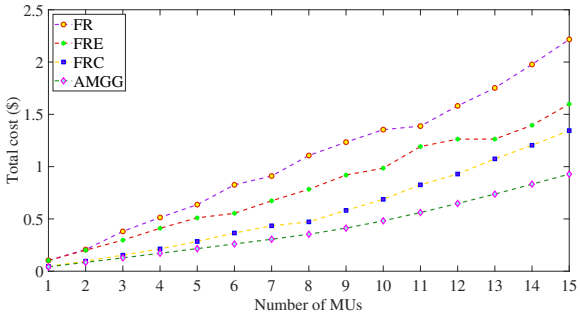


Fig. 9. Total cost ($) comparison of different routing strategies.

To demonstrate the effectiveness of the proposed AMGG service migration strategy, we compare its optimized strategy results with three advanced benchmark strategies, *i.e.*, Fixed Routing (FR) [32], Full Routing to ENs (FRE) [41], and Full Routing to CDC (FRC) [45]. The details of each strategy are shown as follows:

- FR [32]. Following a pragmatic and established approach, tasks from each MU are routed to their respective EN. In some instances, when the resources of each EN are deemed inadequate to handle the tasks, they may be rerouted to CDC.
- FRE [41]. All MU tasks are directed to ENs for remote execution.

- FRC [45]. All MU tasks are directed to the CDC for remote execution.
- The proposed AMGG. Different from FR, all MU tasks are directed to ENs or the CDC dynamically, *i.e.*, the stage 1 of autoencoder training in AMGG is deployed in the CDC, while the stage 2 of population co-evolution in AMGG is deployed either in ENs or the CDC.

Fig. 9 validates the cost optimization results of AMGG and three benchmark strategies as M increases. The graph shows that the cost optimization results for each strategy increase with M. Furthermore, keeping M constant, the optimization results of AMGG are significantly superior to FRC, FRE, and FR. Specifically, the total cost of AMGG is on average 24.88%, 45.56%, and 58.04% lower than that of FRC, FRE, and FR, respectively. The rationale is that FR heavily relies on its predetermined strategies, resulting in significantly higher total cost than AMGG with the same M. Additionally, in FRC and FRE, CDC and EN serve as the sole routers for all tasks, leading to a substantial increase in data transmission cost between MU, ENs, and CDC, as well as execution cost, thereby inflating the total cost. AMGG optimizes the proposed strategies by reasonably allocating task routing decisions among MU, ENs, and CDC, optimizing CPU speeds in ENs, computing appropriate EN service deployment nodes, optimizing service migration cost caused by MU's dynamic movement, and focusing on MU's data transmission power and channel bandwidth allocation to achieve minimization of system cost.

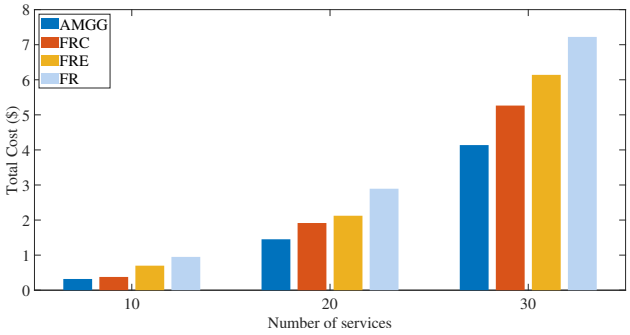


Fig. 10. Total cost ($) of AMGG, FRC, FRE, and FR with varying K.

Fig. 10 illustrates the impact of the number K of decoupled microservices on the optimization total cost results for four different strategies in the application. As K affects the number of decision variables in the optimization space, the experimental results from the graph show that as the number of K increases, the total cost optimized by each strategy sharply rises. It is noteworthy that among the four strategies, AMGG exhibits the best optimization performance. Specifically, compared to FRC, FRE, and FR, AMGG achieves an average reduction of 24.22%, 31.61%, and 49.83% in total cost, respectively. Additionally, an interesting trend in the graph is observed: when K=30, the number of decision variables sharply increases, yet AMGG demonstrates significantly better optimization performance in total cost compared to the other strategies, indicating that AMGG's effectiveness becomes more pronounced in addressing higher-dimensional and more complex problems.

VI. Conclusion

The emergence of hybrid cloud-edge systems, comprising mobile users (MUs), edge nodes (ENs), and cloud data centers, enhances traditional cloud computing and mobile edge computing (MEC) paradigms. It caters better to the diverse data processing needs of modern enterprises. However, dynamic and stochastic characteristics of MEC networks, including the mobility of MUs and temporal variability of tasks, pose significant challenges to MU request routing and service deployment. Furthermore, existing studies often overlook the cost of transferring tasks due to MU mobility. Mobile network operators have difficulty efficiently directing tasks in distributed hybrid cloud-edge systems to decrease the overall cost of intricate applications involving microservices. As the number of MUs in 5G/6G communications grows quickly, these challenges become high-dimensional and complex. We formulate a total cost minimization problem as a high-dimensional mixed-integer nonlinear program (MINLP) to address this. We propose a novel Autoencoder-based Multi-swarm Grey wolf optimizer based on Genetic learning (AMGG) to minimize the high-dimensional MINLP problem. Real-life data-driven simulations demonstrate that the cost of AMGG consistently outperforms the other state-of-the-art algorithms. Our future work aims to further enhance AMGG by integrating more improved variants of autoencoders, *e.g.*, sparse and denoising autoencoders. Moreover, we further plan to extend our approach to the industrial Internet of Things applications and Fog-Cloud-IoT frameworks [26] can also enhance the efficiency and scalability of our proposed approach while considering workflows and load balancing problems. Specifically, by leveraging fog computing's proximity to the data source and cloud computing's scalability, we can enhance the system's ability to handle large-scale and real-time tasks more effectively.

References

[1] W. Zhang, G. Zhang, and S. Mao, "Joint Parallel Offloading and Load Balancing for Cooperative-MEC Systems with Delay Constraints," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4249–4263, Apr. 2022.

[2] A. Irshad, S. A. Chaudhry, O. A. Alomari, K. Yahya, and N. Kumar, "A Novel Pairing-Free Lightweight Authentication Protocol for Mobile Cloud Computing Framework," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3664–3672, Sept. 2021.

[3] N. Piovesan, D. López-Pérez, M. Miozzo, and P. Dini, "Joint Load Control and Energy Sharing for Renewable Powered Small Base Stations: A Machine Learning Approach," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 1, pp. 512–525, Mar. 2021.

[4] S. Mao, J. Wu, L. Liu, D. Lan, and A. Taherkordi, "Energy-Efficient Cooperative Communication and Computation for Wireless Powered Mobile-Edge Computing," *IEEE Systems Journal*, vol. 16, no. 1, pp. 287–298, Mar. 2022.

[5] S. Long, Y. Zhang, Q. Deng, T. Pei, J. Ouyang, and Z. Xia, "An Efficient Task Offloading Approach Based on Multi-Objective Evolutionary Algorithm in Cloud-Edge Collaborative Environment," *IEEE Transactions on Network Science and Engineering,* vol. 10, no. 2, pp. 645-657, 1 March-April 2023.

[6] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939–951, Mar. 2021.

[7] T. Ouyang, Z. Zhou, and X. Chen, "Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[8] X. Ge, S. Tu, G. Mao, C. -X. Wang, and T. Han, "5G Ultra-Dense Cellular Networks," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72–79, Feb. 2016.

[9] D. Calabuig et al., "Resource and Mobility Management in the Network Layer of 5G Cellular Ultra-Dense Networks," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 162–169, Jun. 2017.

[10] F. Zeng, K. Zhang, L. Wu, and J. Wu, "Efficient Caching in Vehicular Edge Computing Based on Edge-Cloud Collaboration," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 2, pp. 2468–2481, Feb. 2023.

[11] X. Xiao et al., "Novel Workload-Aware Approach to Mobile User Reallocation in Crowded Mobile Edge Computing Environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8846–8856, Jul. 2022.

[12] X. Zhu and M. Zhou, "Multiobjective Optimized Deployment of Edge-Enabled Wireless Visual Sensor Networks for Target Coverage," *IEEE Internet of Things Journal*, vol. 10, no. 17, pp. 15325–15337, 1 Sept.1, 2023.

[13] H. El-Fiqi, M. Wang, K. Kasmarik, A. Bezerianos, K. C. Tan, and H. A. Abbass, "Weighted Gate Layer Autoencoders," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 7242–7253, Aug. 2022.

[14] J. Bi, H. Yuan, J. Zhai, M. Zhou, and H. V. Poor, "Self-adaptive Bat Algorithm With Genetic Operations," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 7, pp. 1284–1294, Jul. 2022.

[15] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-Optimized Partial Computation Offloading in Mobile-Edge Computing With Genetic Simulated-Annealing-Based Particle Swarm Optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[16] E. Daniel, "Optimum Wavelet-Based Homomorphic Medical Image Fusion Using Hybrid Genetic-Grey Wolf Optimization Algorithm," *IEEE Sensors Journal*, vol. 18, no. 16, pp. 6804–6811, Aug. 2018.

[17] F. Javidrad and M. Nazari, "A New Hybrid Particle Swarm and Simulated Annealing Stochastic Optimization Method," *Applied Soft Computing*, vol. 60, pp. 634–654, Nov. 2017.

[18] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-Me Cloud: When Cloud Services Follow Mobile Users," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 369–382, 1 April-June 2019.

[19] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A Joint Service Migration and Mobility Optimization Approach for Vehicular Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9041–9052, Aug. 2020.

[20] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic Service Placement for Mobile Micro-Clouds with Predicted Future Costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 1 April 2017.

[21] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, "Mobility-Aware and Delay-Sensitive Service Provisioning in Mobile Edge-Cloud Networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 196–210, 1 Jan. 2022.

[22] S. Schneider, R. Khalili, A. Manzoor, H. Qarawlus, R. Schellenberg, H. Karl, and A. Hecker, "Self-Learning Multi-Objective Service Coordination Using Deep Reinforcement Learning," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3829–3842, Sept. 2021.

[23] T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein, "It's Hard to Share: Joint Service Placement and Request Scheduling in Edge Clouds with Sharable and Non-Sharable Resources," *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Austria, 2018, pp. 365–375.

[24] Y. Yu, J. Yang, C. Guo, H. Zheng, and J. He, "Joint optimization of service request routing and instance placement in the microservice system," *Journal of Network and Computer Applications*, vol. 147, Dec. 2019.

[25] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service Placement and Request Routing in MEC Networks With Storage, Computation, and Communication Constraints," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1047–1060, Jun. 2020.

[26] Sahil, S. Sood, and V. Chang, "Fog-Cloud-IoT Centric Collaborative Framework for Machine Learning-based Situation-aware Traffic Management in Urban Spaces," *Computing*, vol. 106, no. 4, pp. 1193–1225, Oct. 2022.

[27] J. Zhai, J. Bi, H. Yuan, and J. Zhang, "Cost-Effective and Dynamic Migration for Microservices in Hybrid Cloud-Edge Systems," *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Honolulu, Oahu, HI, USA, 2023, pp. 3110–3115.

[28] Y. Yu, J. Liu and J. Fang, "Online Microservice Orchestration for IoT via Multiobjective Deep Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17513–17525, Sept. 2022.

[29] Y. Yu, J. Mo, Q. Deng, C. Zhou, B. Li, X. Wang, N, Yang, Q. Tang, and X. Feng, "Memristor Parallel Computing for a Matrix-Friendly Genetic Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 901–910, Oct. 2022.

[30] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, no. 1, pp. 46–61, Mar. 2014.

[31] N. Nouri, J. Abouei, M. Jaseemuddin, and A. Anpalagan, "Joint Access and Resource Allocation in Ultradense mmWave NOMA Networks With Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1531–1547, Feb. 2020.

[32] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[33] M. -H. Chen, M. Dong, and B. Liang, "Resource Sharing of a Computing Access Point for Multi-User Mobile Cloud Offloading with Delay Constraints," *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2868–2881, Dec. 2018.

[34] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[35] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative Cloud and Edge Computing for Latency Minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[36] Y. Kim, J. Kwak, and S. Chong, "Dual-Side Optimization for Cost-Delay Tradeoff in Mobile Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 2, pp. 1765–1781, Feb. 2018.

[37] W. Li, X. Sun, B. Wan, H. Liu, J. Fang, and Z. Wen, "A Hybrid GA-PSO Strategy for Computing Task Offloading towards MES Scenarios," *PeerJ Computer Science*, 9:e1273, Apr. 2023.

[38] F. Boukouvala, R. Misener, and C. A. Floudas, "Global Optimization Advances in Mixed-Integer Nonlinear Programming, MINLP, and Constrained Derivative-Free Optimization, CDFO," *European Journal of Operational Research*, vol. 252, no. 3, pp. 701–727, Aug. 2016.

[39] A. Jayswal, "An Exact L1 Penalty Function Method for Multidimensional First-order PDE Constrained Control Optimization Problem," *European Journal of Control*, vol. 52, pp. 34–41, Mar. 2020.

[40] M. Cui, L. Li, M. Zhou, and A. Abusorrah, "Surrogate-Assisted Autoencoder-Embedded Evolutionary Optimization Algorithm to Solve High-Dimensional Expensive Problems," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 676–689, Aug. 2022.

[41] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint Computation Offloading and User Association in Multi-Task Mobile Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12313–12325, Dec. 2018.

[42] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE simulator for DTN protocol evaluation," *Proc. 2nd Int. Conf. Simulat. Tools Techn. (ICST)*, Mar. 2009, pp. 50-55.

[43] J. Zhai, J. Bi, and H. Yuan, "Collaborative Computation Offloading for Cost Minimization in Hybrid Computing Systems," *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Prague, Czech Republic, 2022, pp. 1772–1777.

[44] H. Yuan, Q. Hu, J. Bi, G. Gong, J. Zhang, and M. Zhou, "Machine-Level Collaborative Manufacturing and Scheduling for Heterogeneous Plants," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16591–16603, May 1, 2024.

[45] Y. Shi, S. Chen, and X. Xu, "MAGA: A Mobility-Aware Computation Offloading Decision for Distributed Mobile Cloud Computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 164–174, Feb. 2018.

Jiahui Zhai is currently a Ph.D. student in the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. Before that, he received his B.E. degree in Software Engineering from Zhengzhou University in 2019 and M.E. degree in Software Engineering from Beijing University of Technology in 2022. His research interests include cloud/edge computing, data center, task scheduling, computation offloading, intelligent optimization algorithms, machine learning, and reinforcement learning. He was the recipient of the Best Paper Award-Finalist at the 18th IEEE International Conference on Networking, Sensing, and Control (ICNSC).

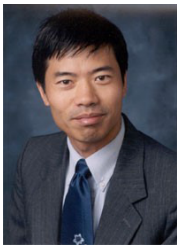
Jing Bi (M'13–SM'16) received her B.S., and Ph.D. degrees in Computer Science from Northeastern University, Shenyang, China, in 2003 and 2011, respectively. From 2013 to 2015, she was a Postdoc researcher in the Department of Automation, Tsinghua University, Beijing, China. From 2011 to 2013, she was a research scientist at the Beijing Research Institute of Electronic Engineering Technology, Beijing, China. From 2009 to 2010, she was a research assistant and participated in research on cloud computing at the IBM Research, Beijing, China. From 2018 to 2019, she was a Visiting Research Scholar with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. She is currently a Professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. She has over 150 publications in international journals and conference proceedings. Her research interests include distributed computing, cloud computing, large-scale data analytics, machine learning and performance optimization. Dr. Bi was the recipient of the IBM Fellowship Award, the Best Paper Award in the 17th IEEE International Conference on Networking, Sensing and Control, and the First-Prize Progress Award of Chinese Institute of Simulation Science and Technology. She is now an Associate Editor of IEEE Transactions on Systems Man and Cybernetics: Systems. She is a senior member of the IEEE.

Haitao Yuan (S'15–M'17–SM'21) received the Ph.D. degree in Computer Engineering from New Jersey Institute of Technology (NJIT), Newark, NJ, USA in 2020. He is currently an Associate Professor at the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, and he is named in the world's top 2% of Scientists List. His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning, and optimization algorithms. He received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, and the Best Paper Award in the 17th ICNSC.

Mengyuan Wang received her Ph.D. degree in Mechanical Engineering from University of Connecticut (UConn), USA in 2021. She worked as a research specialist/scholar in the Center for Clean Energy Engineering of UConn from 2021 to 2023, and joined Beihang University as an associate professor in December 2022. Her research interests focus on the chemical kinetic studies of alternative transportation fuels, including the experimental measurements of combustion characteristics, as well as the development, numerical simulation, and analysis of chemical kinetic mechanisms. She was involved in as one of the core members of five projects from Lawrence Livermore National Laboratory and Department of Energy in U.S., and served as reviewers in American Chemical Society (ACS) Publications and Proceedings of Combustion Institute.

Jia Zhang received the PhD degree in computer science from the University of Illinois at Chicago. She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in Engineering, Professor of Department of Computer Science in the Lyle School of Engineering at Southern Methodist University. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a recent focus on scientific workflows, provenance mining, software discovery, knowledge graph, and interdisciplinary applications of all of these interests in earth science. She is a senior member of the IEEE.

Yebing Wang received the B.Eng. degree in mechatronics engineering from Zhejiang University, Hangzhou, China, in 1997, the M.Eng. degree in control theory and control engineering from Tsinghua University, Beijing, China, in 2001, and the Ph.D. degree in electrical engineering from the University of Alberta, Edmonton, AB, Canada, in 2008. He has been with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, since 2009, where he is currently a Senior Principal Research Scientist and Team Leader. From 2001 to 2003, he was a Software Engineer, a Project Manager, and the Manager of the R&D Department in automation industries, Beijing, China. His current research interests include nonlinear control and estimation, optimal control, adaptive and learning systems, and their applications, including mechatronic systems, robotics, batteries, electric machines, and vehicles.



MengChu Zhou (S'88-M'90-SM'93-F'03) received his B.S. degree in Control Engineering from Nanjing University of Science and Technology, Nanjing, China in 1983, M.S. degree in Automatic Control from Beijing Institute of Technology, Beijing, China in 1986, and Ph. D. degree in Computer and Systems Engineering from Rensselaer Polytechnic Institute, Troy, NY in 1990. He joined the Department of Electrical and Computer Engineering, New Jersey Institute of Technology in 1990, and is now a Distinguished Professor. His interests are in intelligent automation, robotics, Petri nets, Internet of Things, edge/cloud computing, and big data analytics. He has over 1200 publications including 17 books, over 850 journal papers including over 650 IEEE Transactions papers, 31 patents and 32 book-chapters. He is a recipient of Excellence in Research Prize and Medal from NJIT, Humboldt Research Award for US Senior Scientists from Alexander von Humboldt Foundation, and Franklin V. Taylor Memorial Award and the Norbert Wiener Award from IEEE Systems, Man, and Cybernetics Society, and Edison Patent Award from the Research & Development Council of New Jersey. He is a life member of Chinese Association for Science and Technology-USA and served as its President in 1999. He is Fellow of IEEE, International Federation of Automatic Control (IFAC), American Association for the Advancement of Science (AAAS), Chinese Association of Automation (CAA) and National Academy of Inventors (NAI).