Cost-Optimized Task Offloading for Dependent Applications in Collaborative Edge and Cloud Computing

Haitao Yuan, *Senior Member, IEEE*, Qinglong Hu, *Student Member, IEEE*, Shen Wang, *Student Member, IEEE*, Jing Bi, *Senior Member, IEEE*, Rajkumar Buyya, *Fellow, IEEE*, Jinhu Lü, *Fellow, IEEE*, Jinhong Yang, Jia Zhang, *Senior Member, IEEE*, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—A collaborative system that includes mobile devices (MDs), edge nodes (ENs), and the cloud is needed where ENs at the network edge can run offloaded tasks of MDs with limited resources and energy for timely processing for latency-sensitive applications. Unlike existing studies, we formulate a total cost minimization problem for the system for applications, which can be divided into several interdependent subtasks. Each subtask can be executed in MDs, ENs, and the cloud. This work formulates a mixed-integer nonlinear program to minimize the total system cost. To address it, a novel meta-heuristic optimization algorithm called Genetic Simulated-annealing-based Particle swarm optimization with Auto-Encoder (GSPAE) is proposed, which innovatively combines feature extraction of deep learning and global search of meta-heuristic optimization. Genetic operations provide diverse solutions, the Metropolis acceptance of annealing offers a robust global search, and autoencoders (AEs) extract distribution characteristics of particles toward high-quality regions for fast convergence. Thus, GSPAE optimizes the associations between ENs and MDs and the scheduling of subtasks among MDs, ENs, and the cloud. Experiments with large-scale Google cluster datasets show that compared to state-of-the-art benchmark methods, GSPAE reduces the total cost by at least 17% while strictly meeting limits of application latency, available energy, computing, and communication resources of ENs and MDs.**

Received 28 November 2024; accepted 23 December 2024. Date of publication 26 December 2024; date of current version 25 April 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62173013 and Grant 62473014; in part by the Beijing Natural Science Foundation under Grant L233005 and Grant 4232049; and in part by the Beihang World Top University Cooperation Program. *(Corresponding author: Haitao Yuan.)*

Haitao Yuan, Shen Wang, and Jinhu Lü are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: yuan@buaa.edu.cn; 18376420@buaa.edu.cn; jhlu@iss.ac.cn).

Qinglong Hu is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: qinglhu2-c@my.cityu.edu.hk).

Jing Bi is with the School of Software Engineering, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn).

Rajkumar Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

Jinhong Yang is with the CSSC Systems Engineering Research Institute, Beijing 100036, China (e-mail: yangjinhong.66@163.com).

Jia Zhang is with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX 75205 USA (e-mail: jiazhang@smu.edu).

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

Digital Object Identifier 10.1109/JIOT.2024.3522964

***Index Terms*—Autoencoders (AEs), cloud computing, deep learning, edge computing, particle swarm optimization, task offloading.**

I. Introduction

RECENT years have witnessed fast growth of computation-intensive applications in the emerging Internet of Things (IoT) [1]. Mobile devices (MDs) only have constrained energy capacities, communication, computing, and storage resources. It is highly challenging to finish MDs' tasks before their latency bounds. To realize this, a new paradigm of mobile edge computing (MEC) is designed to enable rapid execution for latency/computation-intensive tasks in MDs. Edge servers with high configurations in MEC are often installed and implemented in base stations. MDs can offload some tasks to MEC servers to provide quick services. Current 5G networks can connect billions of MDs directly associated with MEC servers [2]. Therefore, dynamically offloading tasks in MEC is feasible to optimize resource allocation among MDs associated with edge nodes (ENs) in the current literature [3], [4]. MDs all share constrained energy, communication, computing, and storage resources of ENs, making it highly challenging for ENs to realize swift services to available MDs [5]. Thus, the simple integration of MDs and ENs often fails to run MDs' tasks. To address this issue, an edge and cloud collaborative system is frequently used where computation-intensive tasks are dynamically scheduled among ENs, MDs, and a cloud data center (CDC) [6], [7] for realizing the lowest response time and the optimal load balancing [8]. Another method for decreasing the response time in MEC is dynamic partitioning of tasks. Most current methods for offloading dependent tasks fail to support it because the running of a task relies on the execution of its preceding tasks. Nevertheless, a complex task can be dynamically divided into many subtasks. Each subtask can be scheduled to be completed in ENs, MDs, and CDC, respectively, further making task offloading for dependent applications more complicated.

Furthermore, the offloading mentioned above and computing processes in MDs, ENs, and CDC can be completed simultaneously for quick response. Yet, partitioning of dependent tasks is complex because many critical factors, including available computing resources in MDs and ENs, wireless channel information between MDs and ENs, and execution

cost in CDC, need to be jointly considered [9]. The current existing studies have the following drawbacks. First, most existing task partitioning approaches are mainly suitable for the MEC systems with ENs only or simple tasks that do not depend on the execution results of their preceding or subsequent tasks [10]. Second, most current methods are only suitable for small-scale systems with a few MDs and ENs, making them infeasible for applications in large-scale and real-life complex MEC systems [11]. Third, most of the current methods are only designed for relatively simple architectures without CDC, e.g., those with many MDs and one EN [1] and those with multiple MDs and several ENs [12].

Unlike existing studies, we jointly and innovatively optimize the partitioning and offloading of tasks and the user associations in the edge–cloud collaborative system. It focuses on minimizing the collaborative system cost while satisfying the limits of tasks' response time, system energy consumption, and computing and communication resources of ENs and MDs. Significant contributions are summarized as follows.

1) Unlike the existing studies, this work innovatively considers tasks that include subtasks with interdependency, i.e., each task owns many subtasks with interdependency that need to be executed sequentially. This work proposes a hybrid and collaborative edge and cloud system with many ENs, MDs, and CDC, where user associations between ENs and MDs directly affect the subtask number in ENs, the running time of subtasks, and the system energy consumption. Specifically, we establish a constrained optimization problem for optimizing task offloading and partitioning and user associations for subtasks with interdependency in the collaborative system. It is formulated as a mixed integer nonlinear programming (MINLP) problem, which minimizes the system cost.
2) To solve the problem, we design a novel algorithm called Genetic Simulated-annealing-based Particle swarm optimizer with Auto-Encoder (GSPAE). GSPAE innovatively combines an autoencoder (AE) [13], the genetic algorithm (GA), and the rule of metropolis acceptance in simulated annealing (SA) [14] into a particle swarm optimizer (PSO) [15]. In each iteration of GSPAE, genetic operations, including selection, crossover, and mutation, are first used to yield diverse solutions. The metropolis acceptance criterion of SA then conditionally accepts a new solution probabilistically in each iteration, thus increasing global search ability and avoiding trapping into local optima. In each iteration, AE is integrated to extract features of higher-quality particles, speeding up the search process toward regions with better solutions. Thus, GSPAE provides a diverse, global, and fast optimization mechanism, which yields a unique solution that determines associations between MDs and ENs and subtask splitting ratios among MDs, ENs, and CDC. Experiments with real-life datasets from Google cluster demonstrate that GSPAE reduces the total system cost by 43.39% and 7.032% compared to SA-based PSO (SAPSO) and genetic SA-based PSO (GSPSO), respectively, which also combine global search of SA and fast convergence of PSO for yielding a high-quality solution by integrating merits of different algorithms.

The remainder of this article is organized as follows. Section II discusses the related work and shows the novelties of our proposed method. Section III presents a partial offloading framework of the edge-cloud collaborative system and the formulated total cost minimization problem, which has been proven to be NP-hard [17]. Section IV gives the details of GSPAE. Section V gives the extensive simulation results over real-life parameter settings in a typical scenario. Section VI gives the conclusion and points out the future work.

II. Related Work

A. Task Offloading

Task offloading has been a vital edge and cloud computing topic in recent years. Guo et al. [18] applied machine learning to task offloading at a network edge and proposed an intelligent task offloading method. It achieves quicker processing and higher prediction accuracy for resource-hungry applications than several traditional offloading methods. Haber et al. [22] presented a nonconvex and mixed-integer program, which is tackled by a branch and bound algorithm. It jointly optimizes the system execution cost and the energy consumption of MDs in a multiaccess MEC system to meet the latency needs of devices. The two studies only consider an MEC system, which is relatively simple to make task-offloading decisions. To extend them to more realistic cases, Zhang et al. considered a system consisting of a cloud radio access network and MEC. They [21] give an algorithm for joint resource allocation and task offloading in a cloud radio access network supporting MEC. An MINLP is given and addressed by a Lyapunov optimization algorithm to optimize task offloading and scheduling of computation and radio resources. To consider more complex scenarios, further studies consider hybrid systems consisting of MEC and cloud, which are closer to real-life cases in current literature. Specifically, the study in [19], designs an algorithm for offloading tasks in a hybrid edge and cloud system. An NP-hard problem is handled online, which maximizes the number of admitted requests and minimizes their operation cost within a given period. The study in [20], presents an algorithm for offloading tasks in a collaborative cloud and MEC environment. Different offloading decisions are designed for multiple tasks with various resource and latency needs. It effectively achieves load balancing and decreases data leakage risks. Above all, these studies realize intelligent task offloading strategies with machine learning and mathematical optimization methods to reduce the system cost and improve execution performance with minimized resources. However, they fail to consider complex applications, each consisting of multiple interdependent tasks.

Unlike the studies mentioned above, our work focuses on partial computation offloading and user associations for tasks with interdependency in hybrid edge and cloud systems. We consider the dependency of tasks and intelligently offload subtasks among MDs, ENs, and CDC in a cost-optimized manner.

B. Hybrid Edge and Cloud Systems

The design of hybrid edge and multicloud systems has recently received a growing number of studies. Gao et al. [23] designed a dual-focus method to reduce the service latency in the edge cloud with two cloudlet servers. It has a dual focus on communication and computation elements that determine processing latency in the migration of virtual machines and the transmission latency. Fantacci and Picano [24] established an edge and cloud system and optimized the number of dropped tasks whose response time is more significant than their latency limits. They formulate a game-matching problem between applications and edge servers. These studies adopt offline mathematical methods to optimize resources and reduce task latency. To avoid the additional time caused by offline methods, Dinh et al. [25] designed an edge and cloud computing environment in which resource-limited edge devices rent resources from a cloud and provided services to its users. Offline and online algorithms are proposed to optimize resource allocation and procurement for ENs. It adopts offline or online methods with known or partial future demand. Thus, it can be applied to more real-life scenarios. Unlike the above studies, several studies consider more types of loads in hybrid MEC and cloud, which are more challenging to schedule. Mishra et al. [26] designed two types of resource allocation methods by using analytic hierarchy processes in hybrid fog and cloud systems. They aim to decrease the latency of each task by considering heterogeneous loads of network and computing. In addition to considering task routing, Chen et al. [27] further considered a joint task routing and placement problem for latency-sensitive applications. They present a hybrid computing architecture to provide sufficient resources to run real-time and resource-intensive tasks in edge networks. Above studies adopt traditional analytic methods to realize resource allocation in hybrid MEC and cloud. Several studies seek to use recently emerging reinforcement learning or deep learning techniques to realize task offloading. Li et al. [28] formulated the task execution in an MD as a directed acyclic graph and introduce a meta-reinforcement learning approach for dynamical and robust task offloading, thus improving the sampling efficiency of tasks. Tuli et al. [29] employed a tree-based search strategy and a deep neural network-based surrogate model to effectively optimize scheduling decisions, thus improving the efficiency of workflow application scheduling in distributed and parallel MEC computing systems and efficiently utilizing computing resources to meet users' performance requirements. Above all, these studies adopt classical mathematical methods, deep learning, or reinforcement learning to realize task offloading and resource allocation for applications in hybrid edge and cloud systems, thereby improving execution efficiency and reducing task latency. However, they ignore the dependency of multiple tasks belonging to the same complex application, and their methods cannot be directly applied in real-life edge and cloud systems because of task heterogeneity and their simplification of established models.

Unlike them, our work optimizes the cost of the collaborative edge and cloud system while strictly meeting the latency needs of tasks of dependent applications. In addition, it jointly optimizes task partitioning and offloading, as well as user associations, for the first time. Table I compares this work and its state-of-the-art peers in the literature.

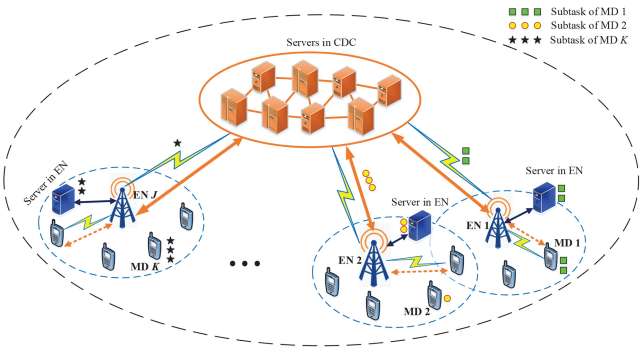


Fig. 1. Collaborative edge and cloud system architecture.

III. Problem Formulation

A. Problem Statement

It is worth noting that this work focuses on task offloading for dependent applications. Each application has multiple jobs, which is more general in the real world. For example, a virtual reality application comprises extracting sensor data, rendering new image frames, compressing data, transferring data, displaying data, etc., [31]. The dependent tasks belonging to the same application must be executed sequentially based on their dependencies, i.e., each subsequent task must wait for all its predecessor tasks to finish execution. The dependencies among tasks are constraints or factors that must be considered when making decisions about task offloading. The task offloading choices bring about the computation cost of ENs. Thus, minimizing the total system cost while considering dependencies among different tasks becomes an issue worth attention.

Fig. 1 shows the collaborative architecture of edge and cloud systems, including K MDs, J ENs, and a centralized CDC. Each EN includes a base station and an MEC server that provides wireless communication. J ENs are connected to CDC with backhaul networks. $x_{k,j}$ is a binary variable showing the association relation between MD k and EN j. If MD k is associated to EN j, $x_{k,j}=1$; otherwise, $x_{k,j}=0$. Similar to [32], it is assumed that each MD is associated with at most one EN in each time slot, i.e.,

$$\sum_{j=1}^{J} x_{k,j} \leq 1, \quad 1 \leq k \leq K \tag{1}$$

π_k is the EN set available to MD k, i.e.,

$$x_{k,j}=0 \ \forall \ j \notin \pi_k. \tag{2}$$

Suppose $x_{k,j}=1$, MD k's offloaded tasks are completed in EN j and transmitted to CDC. We assume an MD has only a task in each time slot and is divided into many subtasks. Each subtask has its input data. A typical example is video object tracking, which can be divided into many dependent segments.

TABLE I
COMPARISON BETWEEN THIS WORK AND ITS STATE-OF-THE-ART PEERS

Reference	User associations between MDs and ENs	Cloud data centers	Feature extraction	Task dependency	Deep learning
[18]	×	×	×	×	×
[19]	×	✓	×	×	×
[20]	✓	✓	×	✓	×
[21]	✓	✓	×	×	×
[22]	✓	✓	×	×	×
[23]	✓	✓	×	×	×
[24]	×	✓	×	×	×
[25]	×	✓	×	×	×
[26]	✓	✓	×	✓	×
[27]	×	✓	×	×	×
[28]	×	✓	✓	✓	✓
[29]	×	✓	×	✓	✓
This work	✓	✓	✓	✓	✓

MDs, ENs, and CDC can complete each task's subtasks simultaneously. α_k is a ratio of subtasks scheduled to MD k. $\beta_{k,j}$ is a ratio of MD k's subtasks scheduled to EN j. $\gamma_{k,j}$ is a ratio of MD k' subtasks scheduled to CDC through EN j. If $x_{k,j}$=0, no subtasks are scheduled to ENs or CDC, i.e., $\beta_{k,j}$=$\gamma_{k,j}$=0. Then

$$0\leq\beta_{k,j},\ \ \gamma_{k,j}\leq x_{k,j}. \tag{3}$$

For each MD k, we have

$$\alpha_k+\sum_{j=1}^{J}\beta_{k,j}+\sum_{j=1}^{J}\gamma_{k,j}=1. \tag{4}$$

Besides, similar to [30], we assume that each MD moves dynamically from one time slot to another, and their moving locations are changed at the start of each time slot. Thus, decision variables, including $x_{k,j}$, α_k, $\beta_{k,j}$, and $\gamma_{k,j}$ are changed once in each time slot.

B. Modeling of Computing Time

1) Computing Time in MDs: I_k is the size of a task's input data (in bits) in MD k. z_k is the CPU cycle number to run a bit for MD k. Thus, $\alpha_k I_k z_k$ is the total CPU cycle number required in MD k. f_k^1 is MD k's running speed (CPU cycles/s). τ_k^1 is MD k's computing time (in s). Then, following [33], we have

$$\tau_k^1=\frac{\alpha_k I_k z_k}{f_k^1}. \tag{5}$$

2) Computing Time in ENs: Similar to [34], each EN with multicore processors can execute subtasks in parallel. Its computing resources are equally allocated to its associated MDs [1]. The subtasks of each MD start execution directly in their associated EN. S_j is the number of channels for EN j, which provides services to S_j MDs at most. Then

$$\sum_{k=1}^{K}x_{k,j}\leq S_j \tag{6}$$

f_j^2 is EN j's running speed. Then, the total number of bits of MD k scheduled to EN j is $\beta_{k,j}I_k$. EN j's computational speed for MD k is $f_j^2/\sum_{k=1}^{K}x_{k,j}$. $\tau_{k,j}^2$ is the computing time of subtasks of MD k offloaded to EN j. Following [35], $\tau_{k,j}^2$ is obtained as

$$\tau_{k,j}^2=\frac{\beta_{k,j}I_k z_k}{\frac{f_j^2}{\sum_{k=1}^{K}x_{k,j}}}. \tag{7}$$

3) Computing Time in CDC: MD k's offloaded subtasks are first delivered to EN j, which first runs its scheduled subtasks and delivers other ones to CDC. $\gamma_{k,j}I_k z_k$ is the total number of CPU cycles needed for subtasks scheduled to CDC. Then, following [36], the computing time of subtasks of MD k scheduled to CDC, $\tau_{k,j}^3$, is calculated as

$$\tau_{k,j}^3=\frac{\gamma_{k,j}I_k z_k}{f_k^3} \tag{8}$$

where f_k^3 is a running speed of CDC.

C. Communication Time

We use an orthogonal and time-frequency sharing approach, i.e., OFDMA [37]. Similar to [38], the total bandwidth is shared by all MDs associated with each EN equally. Besides, a signal-to-interference-plus-noise ratio (SINR) of an uplink network between an EN and its connected MDs is measured [39] at the start of each time slot. W is the channel bandwidth for each EN, and $\theta_{k,j}$ is SINR of the network of uplink between MD k and EN j. According to [33], $\theta_{k,j}$ is given as

$$\theta_{k,j}=\frac{P_k\left(\varphi_{k,j}\right)^{-v}|h_1|^2}{N_0} \tag{9}$$

where P_k is MD k's transmission power. h_1 is a circularly symmetric complex Gaussian variable. v is an exponent constant of the path loss. N_0 is the white Gaussian noise power. $\varphi_{k,j}$ is the distance between MD k and EN j.

$R_{k,j}$ is MD k's rate of data transmission associated with EN j. It is associated with the channel bandwidth, SINR, and the offloading strategy of MDs [38], i.e.,

$$R_{k,j}=\frac{W\log\left(1+\theta_{k,j}\right)}{\sum_{k=1}^{K}x_{k,j}}. \tag{10}$$

Application programs are assumed to have already been installed in hybrid edge and cloud systems servers. Subtasks of each MD offloaded to CDC need to be transmitted through their connected EN, along with ones to be offloaded to the EN. Thus, each MD has to transmit the input data of subtasks offloaded to its connected EN and CDC via an available uplink channel. Let $\tau^a_{k,j}$ denote the offloading time from MD k to EN j. It is calculated by dividing the total offloaded data by the data transmission rate, where the total offloaded data equals the total data offloaded to EN and CDC. Then

$$\tau^a_{k,j}=\frac{(\beta_{k,j}+\gamma_{k,j})I_k}{R_{k,j}}=\frac{(\beta_{k,j}+\gamma_{k,j})I_k\left(\sum_{k=1}^{K}x_{k,j}\right)}{W\log\left(1+\frac{P_k(\varphi_{k,j})^{-v}|h_1|^2}{N_0}\right)}. \tag{11}$$

Each EN is connected to CDC through a wired backhaul network, and M_j is the transmission rate of the backhaul network. The backhaul network bandwidth is shared equally among all MDs associated with the EN. $\tau^b_{k,j}$ is the time for transmitting subtasks of MD k to CDC by EN j, which is given as

$$\tau^b_{k,j}=\frac{\gamma_{k,j}I_k\left(\sum_{k=1}^{K}x_{k,j}\right)}{M_j}. \tag{12}$$

This work ignores the time of transmitting the output result from each EN or CDC back to MDs because its data size is less than that of the input data [1], [40].

D. Completion Time of MDs

We focus on applications consisting of multiple subtasks with interdependency. For example, a mobile visual search is a typical dependent application for electrical tourism services. It has six significant subtasks, i.e., region of interest positioning, object detection, classification of video images, semantic analysis of video images, feature extraction, and information searching [41]. Each subtask relies on the execution result of its preceding ones. Each subtask's output is used as input for the following subtask. We split the subtasks into three interdependent parts completed in the beginning, middle, and final stages. In the beginning stage, the subtasks are executed in MDs. Each MD begins to run its subtasks and offloads others to its connected EN simultaneously. Each MD delivers the executed output of its subtasks to its associated EN. Upon getting the output, the EN executes its scheduled subtasks and delivers them simultaneously to the CDC. After the EN completely executes its subtasks, the EN further transmits the yielded output to CDC. Finally, the CDC transmits the finally yielded output back to MD. $\widetilde{\tau}^1_k$ is the time when the EN begins executing its scheduled subtasks, which is given as [38]

$$\widetilde{\tau}^1_k=\max\left\{\tau^1_k,\ \tau^a_{k,j}\right\} \tag{13}$$

$\widetilde{\tau}^a_{k,j}$ is the time from the beginning of running in EN to that of running in CDC, i.e.,

$$\widetilde{\tau}^a_{k,j}=\max\left\{\tau^2_{k,j},\ \tau^b_{k,j}\right\} \tag{14}$$

$\widetilde{T}_k$ is the time of completing the execution of all subtasks from MD k, which is given as

$$\widetilde{T}_k=\widetilde{\tau}^1_k+\sum_{j=1}^{J}x_{k,j}\widetilde{\tau}^a_{k,j}+\sum_{j=1}^{J}x_{k,j}\tau^3_{k,j} \tag{15}$$

$\hat{T}_k$ is the upper limit of $\widetilde{T}_k$, which is given as

$$\widetilde{T}_k\le\hat{T}_k. \tag{16}$$

E. CPU and Memory Usage in ENs

$\hat{\phi}^1_j$ is the number of EN j's total CPU cycles. The total number of CPU cycles required by all MDs associated with EN j cannot be larger than $\hat{\phi}^1_j$, i.e.,

$$\sum_{k=1}^{K}I_k\beta_{k,j}z_k\le\hat{\phi}^1_j \tag{17}$$

$\hat{\phi}^2_j$ is the number of EN j's total memory units. The total number of memory units required by all MDs associated with EN j cannot be larger than $\hat{\phi}^2_j$, i.e.,

$$\sum_{k=1}^{K}I_k\beta_{k,j}w_k\le\hat{\phi}^2_j \tag{18}$$

where w_k is the number of memory units required to execute a bit of data of MD k.

F. Modeling of Execution Cost

Following [33], MD k's power consumption is $q^1_k(f^1_k)^3$. q^1_k is a parameter depending on MD k's chip architectures. f^1_k is MD k's running speed (CPU cycles/s). $\hat{f}^1_k$ is the upper limit of f^1_k, i.e.,

$$0\le f^1_k\le\hat{f}^1_k,\ f^1_k\in N^+. \tag{19}$$

The executing time of the scheduled subtasks in MD k is $I_kz_k\alpha_k/f^1_k$. The energy consumption of executing the scheduled subtasks in MD k is $I_kz_k\alpha_kq^1_k(f^1_k)^2$. $\hat{E}_k$ is the upper limit of MD k's available energy, i.e.,

$$I_kz_k\alpha_kq^1_k\left(f^1_k\right)^2\le\hat{E}_k \tag{20}$$

ψ^1_k is MD k's electricity price. F_1 is the energy consumption of running subtasks in MDs, i.e.,

$$F_1=\sum_{k=1}^{K}\left(\psi^1_kI_kz_k\alpha_kq^1_k\left(f^1_k\right)^2\right) \tag{21}$$

F_2 is the cost of all ENs and ψ^2_j is the price of electricity for EN j. q^2_j is a parameter showing EN j's chip architectures. f^2_j is EN j's running speed (cycles/s). $\hat{E}^2_j$ is EN j's upper limit of available energy. The total energy consumption of running EN j's subtasks cannot be larger than $\hat{E}^2_j$, i.e.,

$$\sum_{k=1}^{K}\left(I_kz_k\beta_{k,j}q^2_j\left(f^2_j\right)^2\right)\le\hat{E}^2_j. \tag{22}$$

Following [42], F_2 is given as

$$F_2=\sum_{k=1}^{K}\sum_{j=1}^{J}\left(\psi_j^2 I_k z_k \beta_{k,j} q_j^2\left(f_j^2\right)^2\right) \tag{23}$$

F_3 is the cost of CDC. μ_3 is the cost of executing a CPU cycle of CDC. Then

$$F_3=\mu_3\left(\sum_{k=1}^{K}\left(I_k z_k \sum_{j=1}^{J}\gamma_{k,j}\right)\right) \tag{24}$$

F is the cost of the collaborative edge and cloud system, which is obtained as

$$F=F_1+F_2+F_3. \tag{25}$$

G. Constrained Optimization Problem

Following the discussion above, the problem of total cost minimization for the collaborative edge and cloud system is given as follows:

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{x}}{\mathbf{Min}} \quad \{F\}$$

subject to

$$I_k z_k \alpha_k q_k^1\left(f_k^1\right)^2 \le \hat{E}_k \tag{26}$$

$$\sum_{k=1}^{K}\left(I_k z_k \beta_{k,j} q_j^2\left(f_j^2\right)^2\right) \le \hat{E}_j^2 \tag{27}$$

$$\tilde{\tau}_k^1+\sum_{j=1}^{J} x_{k,j}\tilde{\tau}_{k,j}^a+\sum_{j=1}^{J} x_{k,j}\tau_{k,j}^3 \le \hat{T}_k \tag{28}$$

$$\sum_{k=1}^{K} I_k \beta_{k,j} z_k \le \hat{\phi}_j^1 \tag{29}$$

$$\sum_{k=1}^{K} I_k \beta_{k,j} w_k \le \hat{\phi}_j^2 \tag{30}$$

$$\alpha_k+\sum_{j=1}^{J}\beta_{k,j}+\sum_{j=1}^{J}\gamma_{k,j}=1 \tag{31}$$

$$0\le f_k^1 \le \hat{f}_k^1,\ f_k^1 \in N^+ \tag{32}$$

$$\sum_{j=1}^{J} x_{k,j} \le 1 \tag{33}$$

$$\sum_{k=1}^{K} x_{k,j} \le S_j \tag{34}$$

$$\beta_{k,j},\ \gamma_{k,j} \le x_{k,j} \tag{35}$$

$$0\le \alpha_k,\ \beta_{k,j},\ \gamma_{k,j} \le 1 \tag{36}$$

$$x_{k,j} \in \{0, 1\} \tag{37}$$

$$x_{k,j}=0\ \forall j \notin \pi_k \tag{38}$$

F is nonlinear regarding decision variables. α_k, $\beta_{k,j}$, and $\gamma_{k,j}$ are real and $x_{k,j}$ is integer. Besides, many constraints, e.g., (26) and (27), are also nonlinear regarding decision variables. The optimization objective F of the problem and several of its constraints, (26)–(28), are nonlinear, and its decision variables include continuous and discrete types. This problem combines the combinatorial difficulty of optimizing over discrete and continuous variable sets with the challenge of handling the nonlinear optimization function F. It is an NP-hard problem, and its solution usually needs to find enormous search trees. Thus, it is a typical MINLP problem [43].

This real-world problem cannot be solved to global optimality with deterministic, plane-cutting, relaxation-based, or branch-and-bound methods because it is too large and generates a huge search tree, and it has to be solved in real-time. Thus, obtaining a good solution faster than waiting for the optimal one is more desirable. This work resorts to meta-heuristic search algorithms that quickly provide a close-to-optimal point without optimality guarantees. Meta-heuristics accelerate deterministic methods. We adopt an approach of penalty function [44] to handle these constraints. Each constraint is first transformed into a non-negative penalty value. Then, a new evaluation metric is obtained as its new objective function. As shown in (39), the penalty of each inequality constraint χ_1 is $(\max\{0, -\overline{=}{h}_{\chi_1}(\vec{h})\})^2$, and that of each equality constraint χ_2 is $|\overline{\neq}{h}_{\chi_2}(\vec{h})|^2$. The total penalty, Ω, is 0 if all constraints are met

$$\begin{aligned}
&\underset{\vec{h}}{\mathbf{Min}}\left\{\widetilde{F}=\overset{\infty}{\mathcal{N}}\Omega+F\right\}\\
&\Omega=\sum_{\chi_1=1}^{\overline{\mathbb{N}}}\left(\max\{0, -\overline{=}{h}_{\chi_1}(\vec{h})\}\right)^2+\sum_{\chi_2=1}^{\overline{=}{\mathbb{N}}}\left|\overline{\neq}{h}_{\chi_2}\left(\vec{h}\right)\right|^2\\
&\overline{=}{h}_{\chi_1}\left(\vec{h}\right)\ge 0\\
&\overline{\neq}{h}_{\chi_2}\left(\vec{h}\right)=0
\end{aligned} \tag{39}$$

where $\vec{h}$ is a vector of decision variables. $\widetilde{F}$ is a transformed objective function. $\overset{\infty}{\mathcal{N}}$ is a positive number. $\overline{\neq}{\mathbb{N}}$ ($\overline{=}{\mathbb{N}}$) is the number of inequality (equality) constraints.

IV. Genetic Simulated-Annealing-Based Particle Swarm Optimizer With Auto-Encoder

Traditional approaches cannot well solve MINLP, e.g., dynamic programming methods [45] and gradient descent ones [46]. These methods can only be applied if specific mathematical properties are met, and they can only obtain low-quality solutions given a limited time. Meta-heuristic algorithms can effectively solve MINLP problems, considered the most complex optimization. PSO follows simple exploratory behaviors of bird movements and fish schooling. In PSO, the position and velocity of each particle are randomly initialized. PSO memorizes the best position of the swarm and that of its own, as well as the velocity. In each generation, a particle adjusts the velocity of each dimension according to search experiences of its own and the swarm. Each particle updates its position iteratively in the multidimensional search space until it reaches the stopping criterion or the optimal result. Connections among different dimensions of the problem space

are realized via objective functions. However, PSO is fast but has an issue of early convergence [15].

SA starts with an arbitrarily generated solution to a problem. A newly generated solution is rejected or accepted based on probability. An improved solution with a better objective function value is accepted with certainty. In contrast, a deteriorating solution is received with a probability depending on the degree of worsening of the solution and the current temperature. SA yields a sequence of solutions. The temperature at early stages in SA is higher, increasing the acceptance probability of a worsening solution. This probabilistic acceptance of the worse solution enables SA to escape local optima. As iterations continue, the temperature decreases following a chosen cooling schedule, decreasing the acceptance probability of worse solutions. However, SA has a high global search ability and relatively slow convergence [14].

GA mimics natural selection (survival of the fittest) and biological reproduction of the fittest individual. The optimal solution (the fittest individual) is yielded from generation to generation without relying on the strict mathematical formulation. The optimal solution comprises the best components (genes) of the fittest individual in previous generations. GA is a nonlinear, stochastic process and discrete event rather than a mathematically guided algorithm. GA works on a population of individuals and chooses a parent pool from the population using the selection criteria to prepare the next generation. The crossover and mutation operations provide the population with new individuals. The crossover operation yields offspring by exchanging partial bit strings and inverting bits between two parents. The mutation operation may flip some genes of the new offspring. GA evaluates each individual with a specific fitness function. In the last generation, the fittest individual is viewed as the optimal solution and yielded. GA has a high diversity of individuals due to its genetic operations, yet its running speed is unsatisfying [47].

The AE structure is a symmetrical network, including an encoder and a decoder [48]. The former encodes the original input (the locally best positions of particles) into low-dimensional hidden variables to learn important features. The encoding function is denoted by $\Gamma(\cdot)$. The latter restores hidden variables to those with the original dimension. The decoding operation is denoted by $\Theta(\cdot)$. The output of the optimal AE perfectly yields the original input. ϖ denotes a training sample (the locally best position). $\widetilde{\varpi}$ is the output of the optimal AE, i.e.,

$$\widetilde{\varpi}=\Theta(\Gamma(\varpi))\approx\varpi. \tag{40}$$

We propose a novel meta-heuristic optimization algorithm, GSPAE, to solve the optimization problem in Section III. GSPAE adopts genetic operations to yield a superior exemplar for each particle. GSPAE combines GA and PSO cohesively. It consists of three major steps: 1) producing superior exemplars with GA; 2) changing particles in PSO; and 3) utilizing AE to guide the evolution of particles with distribution characteristics of the locally best positions. Specifically, each particle in GSPAE is affected by its superior exemplar, the globally best position, and its locally best position.

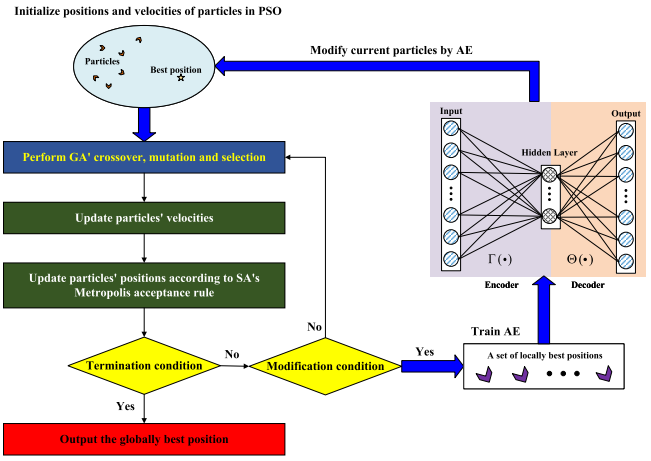


Fig. 2. Process of GSPAE.

Specifically, the main characteristics of GSPAE are listed here. First, PSO's particles learn from superior exemplars, which are high-quality and more diversified, thereby avoiding a problem of premature convergence. Second, the SA's rule of metropolis acceptance is used to update superior exemplars and find global optima with high possibility. Third, the genetic information of high-quality particles is propagated back to GA for yielding better superior exemplars. Fourth, AE utilizes high-quality particles and updates particles to optimize the optimization process. GSPAE finds close-to-optimal decision variables. The flowchart of GSPAE is given in Fig. 2.

GSPAE first initializes the positions and velocities of particles in PSO. Then, it performs GA's crossover, mutation, and selection operations. According to the SA's metropolis acceptance rule, GSPAE then updates particles' velocities and positions. If the termination condition is met, it outputs the globally best position; otherwise, if the modification condition is met, GSPAE selects a set of locally best positions and modifies them with AE, which extracts features of higher-quality particles and guides the population toward regions with better solutions. Then, GSPAE continues its iterations. In addition, if the modification condition (every certain number of iterations) is not met, GSPAE directly evolves. GSPAE terminates and yields the globally best solution if the termination condition is met.

The particle position includes D entries. The first K elements keep α_k. The next KJ elements keep $\beta_{k,j}$. The next KJ elements keep $\gamma_{k,j}$. Then, the next KJ elements keep $x_{k,j}$. Thus, $D=K+KJ+KJ+KJ=K(3J+1)$. $\mathbb{X}$ is a set of particles in each population. Let $|\mathbb{X}|$ denote the particle number in PSO. p_i is particle i's locally best position. g is the global best position of the current population. D means the number of entries in each position. e_i means particle i's superior exemplar. $e_{i,d}$ is entry d ($1 \le d \le D$) of e_i, which is obtained as

$$e_{i,d}=\frac{c_1\cdot r_1\cdot p_{i,d}+c_2\cdot r_2\cdot g_d}{c_1\cdot r_1+c_2\cdot r_2} \tag{41}$$

where $p_{i,d}$ means entry d of p_i, g_d is entry d of g, c_1 (c_2) is a cognitive (social) acceleration constant for $p_{i,d}$ and g_d. r_1 and r_2 mean random values in (0,1).

GSPAE includes five major components: crossover, mutation, selection, position update, and updating particles with AE. They are presented next.

A. Crossover Component

It is conducted on p_i and g to yield a new individual o_i. p_κ is a position of a random particle, and $p_{\kappa,d}$ is its entry d. If $\widetilde{F}(p_i)<\widetilde{F}(p_\kappa)$, o_i inherits more dimensions from p_κ. Its entry d ($o_{i,d}$) is obtained as

$$o_{i,d}=\begin{cases} r_d\cdot p_{i,d}+(1-r_d)\cdot g_d, & \widetilde{F}(p_i)<\widetilde{F}(p_\kappa) \\ p_{\kappa,d}, & \text{otherwise} \end{cases} \tag{42}$$

where r_d is a random value in (0,1). Unlike the random selection in GA, the previous search information is adopted to enhance the search efficiency.

B. Mutation Component

ζ denotes the given mutation possibility. For each entry d, if $r_d<\zeta$, $o_{i,d}$ is changed with a value in $(\check{\lambda}_d, \hat{\lambda}_d)$, i.e.,

$$o_{i,d}=\mathbf{rand}\left(\check{\lambda}_d, \hat{\lambda}_d\right), \text{ if } r_d<\zeta \tag{43}$$

where $\check{\lambda}_d$ and $\hat{\lambda}_d$ mean lower and upper limits of element d of particle i. In this way, the mutation component increases the diversity and distribution of superior exemplars.

C. SA-Based Selection Component

SA's metropolis acceptance rule is utilized to conditionally select o_i or e_i. If $\widetilde{F}(o_i)<\widetilde{F}(e_i)$, o_i is used. η_t is the temperature in iteration t. If $\widetilde{F}(o_i)\geq\widetilde{F}(e_i)$ and $\exp^{(-\frac{\widetilde{F}(o_i)-\widetilde{F}(e_i)}{\eta_t})}>\xi$, o_i is used. ξ is a random value in (0,1). If $\exp^{(-\frac{\widetilde{F}(o_i)-\widetilde{F}(e_i)}{\eta_t})}\leq\xi$ and $\widetilde{F}(o_i)\geq\widetilde{F}(e_i)$, e_i is fixed. Thus

$$e_i=\begin{cases} o_i, \ \widetilde{F}(o_i)<\widetilde{F}(e_i) \\ o_i, \ \widetilde{F}(o_i)\geq\widetilde{F}(e_i) \text{ and } \exp^{\left(-\frac{\widetilde{F}(o_i)-\widetilde{F}(e_i)}{\eta_t}\right)}>\xi \\ e_i, \ \widetilde{F}(o_i)\geq\widetilde{F}(e_i) \text{ and } \exp^{\left(-\frac{\widetilde{F}(o_i)-\widetilde{F}(e_i)}{\eta_t}\right)}\leq\xi \end{cases}. \tag{44}$$

D. Position Update of Particles

v_i means particle i's velocity and $\vec{\hbar}_i$ denotes its position. $\vec{\hbar}_{i,d}$ is entry d of $\vec{\hbar}_i$. They are updated as follows:

$$v_{i,d}=\omega_t\cdot v_{i,d}+c\cdot r_d\cdot\left(e_{i,d}-\vec{\hbar}_{i,d}\right) \tag{45}$$

$$\vec{\hbar}_{i,d}=\vec{\hbar}_{i,d}+v_{i,d} \tag{46}$$

$$\omega_t=\hat{\omega}-\frac{t\left(\hat{\omega}-\check{\omega}\right)}{\hat{t}} \tag{47}$$

where $\hat{t}$ means the number of iterations, ω_t means the inertia weight in iteration t, c is a parameter reflecting the variation between $e_{i,d}$ and $x_{i,d}$, and $\check{\omega}$ and $\hat{\omega}$ are lower and upper limits of ω_t.

E. Particle Update With AE

A particle ($\vec{\hbar}$) is modified by the trained $\Gamma(\cdot)$ and $\Theta(\cdot)$. $\widetilde{\vec{\hbar}}$ is the yielded output of AE, i.e.,

$$\widetilde{\vec{\hbar}}=\Theta\left(\Gamma(\vec{\hbar})\right). \tag{48}$$

Algorithm 1 GSPAE

1: Initialize the positions and velocities
2: Obtain $\widetilde{F}(\vec{\hbar}_i)$ with (39)
3: Determine p_i and g
4: Perform the parameter initialization of SA, GA, PSO, and AE
5: $t\leftarrow 1$
6: $\eta_t\leftarrow\eta_1$
7: **while** $t\leq\hat{t}$ **do**
8: Perform the crossover (42) to yield o_i
9: Perform the mutation (43) on o_i
10: Perform the SA-based selection to update e_i
11: Update v_i and $\vec{\hbar}_i$ of particle i with (45) and (46)
12: Obtain $\widetilde{F}(\vec{\hbar}_i)$ of particle i with (39)
13: Change p_i and g
14: $\omega_t\leftarrow\hat{\omega}-\frac{t(\hat{\omega}-\check{\omega})}{\hat{t}}$
15: $t\leftarrow t+1$
16: $\eta_t\leftarrow\eta_t*\delta$
17: **if** $t\%\widetilde{t}==0$ **then**
18: Train the AE
19: Reconstruct each particle ($\vec{\hbar}$) with AE
20: **end if**
21: **end while**
22: **return** g

The locally best positions are utilized as training data to retrain AE every $\widetilde{t}$ iterations. During this reconstruction, particles are infused with high-quality particles. This procedure enhances the population's overall quality, guiding it toward the solution space inhabited by high-quality solutions. Consequently, it facilitates the population's exploration of the solution space where higher-quality solutions exist.

Algorithm 1 gives pseudocodes of GSPAE. Line 1 initializes the velocities and positions. Line 2 obtains $\widetilde{F}(\vec{\hbar}_i)$ of each particle i with (39). Line 3 updates p_i and g, respectively. Line 4 performs the parameter initialization of SA, GA, PSO, and AE. Line 6 initializes η with η_1, which denotes the initial temperature of SA. The while loop continues if $t\leq\hat{t}$ in line 7. Line 8 performs the crossover (42) to update o_i. Line 9 performs the mutation (43) on o_i. Line 10 performs the SA-based selection to change e_i. Line 11 updates v_i and $\vec{\hbar}_i$ of particle i with (45) and (46), respectively. Line 12 obtains $\widetilde{F}(\vec{\hbar}_i)$ with (39). Line 13 updates p_i and g, respectively. Line 14 decreases ω_t from $\hat{\omega}$ to $\check{\omega}$ linearly. Line 16 reduces η_t by δ. Line 15 changes t. If $t\%\widetilde{t}$ is 0, line 18 retrains the AE. Line 19 reconstructs $\vec{\hbar}$ with the AE. Finally, line 22 returns g, including decision variables of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and $\boldsymbol{x}$.

We further give the complexity of Algorithm 1. The main complexity is caused by the while loop. Lines 8–16 show that the complexity of the crossover, the mutation, the selection with SA, and the update OF positions in each iteration is $\mathcal{O}(|\mathbb{X}|D)$. The complexity of training AE is $\mathcal{O}(D|\mathcal{T}|\varrho)$. ϱ means the number of epochs in AE, and $|\mathcal{T}|$ is the number of training samples. Thus, $D=K(3J+1)$, and the complexity of each iteration is $\mathcal{O}((|\mathbb{X}|+|\mathcal{T}|\varrho)K(3J+1))$. Algorithm 1's complexity is $\mathcal{O}((|\mathbb{X}|+|\mathcal{T}|\varrho)K(3J+1)\hat{t})$.

TABLE II
MAIN PARAMETER SETTING OF GSPAE

$\|\mathbb{X}\|$	c	$c_1(c_2)$	ζ	η_1	δ	$\hat{t}$	$\widetilde{t}$	$\check{\omega}$	$\hat{\omega}$
50	1.49618	0.5	0.2	10^8	0.95	2,400	200	0.4	0.95

V. PERFORMANCE EVALUATION

We evaluate GSPAE by considering a region with multiple ENs and J=10. MDs handle real-world application tasks, i.e., Linpack benchmarks for Android [49], which can either be fully completed in MDs or partially scheduled to remote servers. GSPAE is implemented with PyCharm 2022 in a server with an Intel Core i7-12700H CPU with 12-core 4.9 GHz processors and 32-GB memory.

A. Parameter Setting

Similar to [33], MDs' parameters are given here. $w_k = 8$, $\hat{f}_k^1 = 4\times10^8$, $\hat{E}_k = 6$ J, $P_k = 0.1$ W, $\hat{T}_k = 2.5$ s, $f_k^1 = 4\times10^8$, $q_k^1 = 10^{-26}$, $\psi_k^1 = 5.06\times10^{-6}$ \$/J, $z_k = 100$ cycles/bit, and $I_k = [3\,072\,000, 21\,504\,000]$. For each MD, if its distance from EN j is beyond 800 m, $j \notin \pi_k$; otherwise, $j \in \pi_k$. ENs' parameters are given here. $v = 4$, $M_j = 100$ Mb/s, $W = 10$ Mhz, $h_1 = 0.98$, $\hat{\phi}_j^2 = 2$ GB, $S_j = 5$, $\hat{E}_j^2 = 20$ J, $\hat{\phi}_j^1 = 8\times10^9$, $\psi_j^2 = 2.5\times10^{-6}$ \$/J, $q_j^2 = 10^{-27}$, and $f_j^2 = 8\times10^8$. CDC's parameters are given here. $f_k^3 = 2.5\times10^9$ cycles/s, and $\mu_3 = 1.142\times10^{-9}$ \$ per CPU cycle. Following [1], GSPAE's parameters are given in Table II. The number of neurons in the hidden layer in AE is $\frac{2}{3}$ of that in the layer of input, the number of epochs is 200, the rate of learning is 10^{-3}, $\overset{\infty}{\mathcal{N}}$=$10^3$, and the function of activation is sigmoid. Here, we have two parameter settings for demonstrating the performance of GSPAE. In parameter setting 1, $K = 10$, $J = 10$, $S_j = 5$, and $I_k \in [3\,072\,000, 21\,504\,000]$. In parameter setting 2, $I_k = 21\,504\,000$, $K \in [1, 11]$, $J = 10$, and $S_j = 5$.

B. Experimental Results

We first compare GSPAE with its typical peers, including SA, GA, SA-based PSO (SAPSO), and GSPSO.

1) SA [50] jumps from the locally best solutions and finds global optima in the theory. The comparison between it and GSPAE proves GSPAE's search accuracy.
2) GA [51] owns high individual quality due to its genetic operations. Thus, GSPAE's comparison with GA can show its search accuracy.
3) SAPSO [52] owns quick convergence of PSO and SA's high accuracy. Therefore, the comparison between GSPAE and SAPSO can demonstrate the accuracy and convergence of GSPAE's search.
4) GSPSO [1] has excellent optimization ability in solving low-dimensional optimization problems. In contrast to it, GSPAE integrates an AE to enhance global search accuracy. Specifically, the locally best positions are training samples for updating AE every $\widetilde{t}$ iterations, which extracts features of high-quality solutions and reconstructs outputs based on them. Then, the trained AE modifies each particle in the current population according to the grasped features. In addition, GSPAE's search capability in high-dimensional MINLP is enhanced due to our novel integration of AE. The comparison between GSPSO and GSPAE proves the improvement brought by AE to GSPAE.

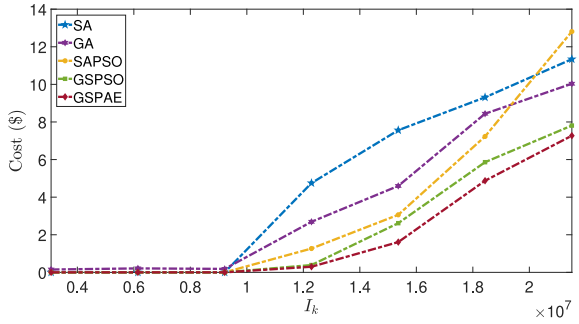


Fig. 3. Cost of SA, SAPSO, GA, GSPSO, and GSPAE versus I_k.

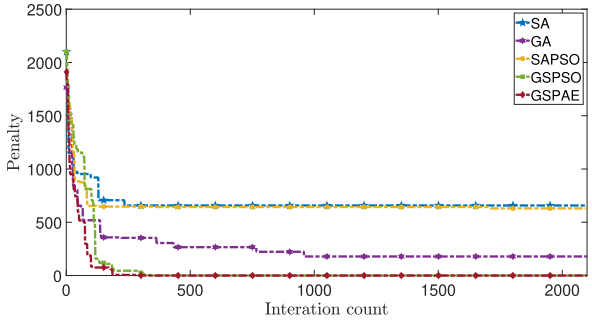


Fig. 4. Penalties of SA, GA, SAPSO, GSPSO, and GSPAE.

C. Algorithm Comparison With Parameter Setting 1

Fig. 3 illustrates the cost of SA, GA, SAPSO, GSPSO, and GSPAE given different I_k with parameter setting 1. We change I_k and fix other parameters here. The cost of each algorithm increases when I_k is larger than 9 216 000. MDs and ENs have limited computing abilities and cannot satisfy users' latency bounds. Therefore, in this way, MDs must offload some subtasks to CDC, increasing the cost. The cost of GSPAE is decreased by 17% and 33% compared to GSPSO when I_k=18 432 000 and I_k=15 360 000, respectively. Figs. 4–6 show penalties, cost, and fitness values of SA, GA, SAPSO, GSPSO, and GSPAE in each iteration with parameter setting 1. Here, the fitness is calculated by (39). Given this MINLP, larger I_k means fewer solutions satisfy the latency constraint (16). This means that the problem with larger I_k is more difficult to solve by GSPAE.

Fig. 4 shows that only GSPAE and GSPSO obtain valid solutions at the end of their iterations. The result demonstrates that GSPAE and GSPSO have excellent search capabilities that surpass their peers. It is also shown that the final cost (7.2698 \$) of GSPAE is the least among the five algorithms. In addition, unlike GA, SA, and SAPSO, after the penalties are close to 0, the cost of GSPAE and GSPSO is constantly reduced, indicating that they jump out of locally best solutions. Furthermore, Fig. 6 shows the fitness values of five algorithms

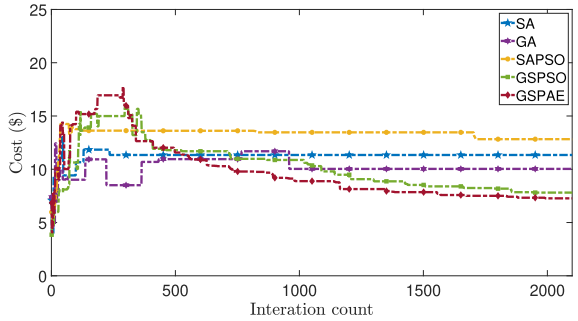


Fig. 5. Cost of SA, GA, SAPSO, GSPSO, and GSPAE.

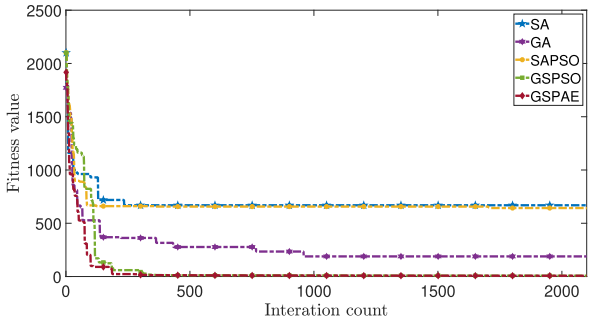


Fig. 6. Fitness values of SA, GA, SAPSO, GSPSO, and GSPAE when K=10, J = 10, and I_k=21 504 000.

in each iteration with parameter setting 1. It is demonstrated that SA and SAPSO converge with fewer iterations than GSPAE, but their final fitness values are much larger than GSPAE's. GSPAE yields the least fitness value with the fewest iterations, the reason is that it combines strong global search ability of SA and fast convergence of PSO to yield a high-quality solution by integrating the merits of different algorithms, which verifies its convergence speed and search accuracy.

D. Algorithm Comparison With Parameter Setting 2

As K becomes larger, the dimension of each solution increases dramatically, and the search space also grows exponentially. Figs. 7 and 8 show the penalties and the cost of SA, GA, SAPSO, GSPSO, and GSPAE versus K with the parameter setting 2. It is shown that when $K \leq 5$, all algorithms obtain their valid solutions, and the cost of GSPSO and GSPAE is the same and the smallest. When $K > 5$, SA, SAPSO, and GA fail to find valid solutions with penalties of 0 in the huge solution space, yet GSPSO and GSPAE successfully obtain valid ones. Furthermore, when $K > 5$, the cost of GSPAE is smaller than that of GSPSO, which illustrates that the AE effectively improves the optimization ability of GSPAE in solving the high-dimensional MINLP. Thus, the results with two parameter settings demonstrate that GSPAE achieves the best among all algorithms with good robustness.

E. Allocation of Subtasks

Fig. 9 shows the allocation ratios of subtasks versus I_k. More subtasks are scheduled to ENs when $I_k < 6\,144\,000$. This is because ENs have more resources than MDs, and

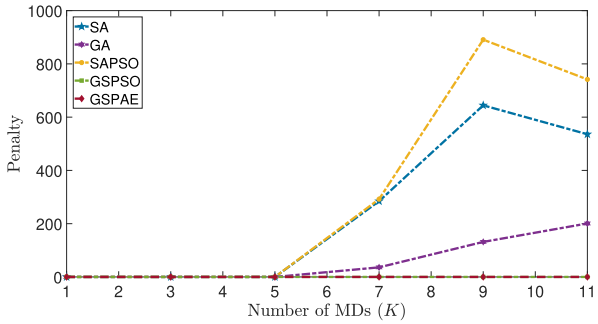


Fig. 7. Penalties of SA, GA, SAPSO, GSPSO, and GSPAE versus K.

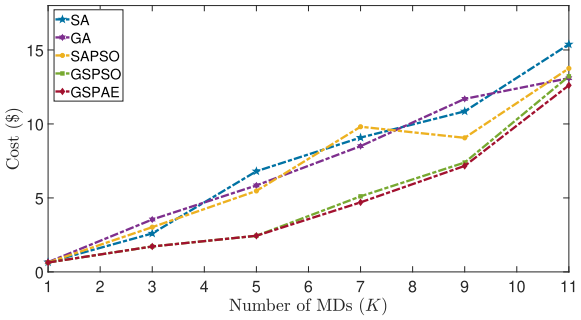


Fig. 8. Cost of SA, GA, SAPSO, GSPSO, and GSPAE versus K.

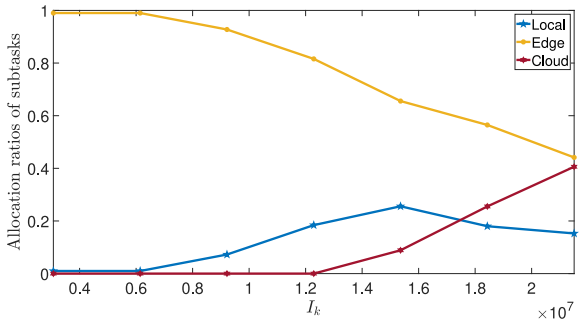


Fig. 9. Allocation ratios of subtasks versus I_k.

their cost is less than that of MDs. When $I_k \in [6\,144\,000, 12\,288\,000]$, more subtasks are scheduled to MDs because ENs own limited computing resources and fail to run all subtasks. Besides, MDs' prices are cheaper than those of the CDC. When $I_k > 12\,288\,000$, due to CDC's almost unlimited resources, more subtasks are offloaded to it. The ratio of subtasks allocated to MDs decreases when $I_k > 15\,375\,000$. The number of subtasks allocated to MDs does not decrease, but it reaches the processing limits of MDs. Therefore, as I_k increases, α decreases accordingly, which is reflected in Fig. 9.

Fig. 10 shows allocation ratios of subtasks versus K. The allocation ratio for MDs is almost fixed, ENs are reduced, and CDC is reduced as K increases. The subtask number scheduled to ENs reaches their capacities because of their limited capacities of CPU cycles and memory units as K increases. In addition, a more significant number of subtasks are scheduled for CDC because of its sufficient resources. Nevertheless, the execution cost in CDC is larger, increasing the total system cost.

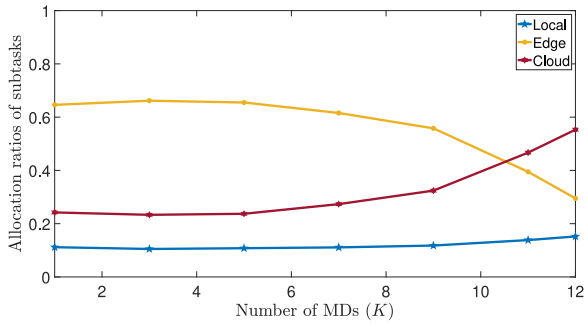


Fig. 10. Allocation ratios of subtasks versus K.

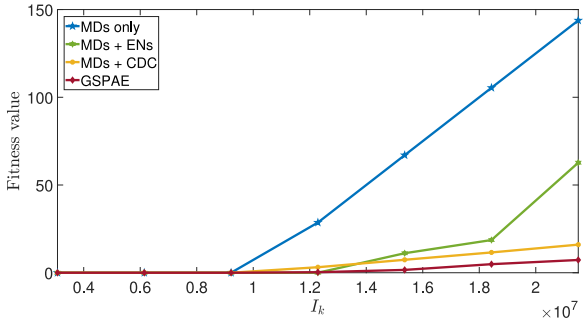


Fig. 11. Fitness value of each algorithm versus I_k.

F. Comparison of Different Strategies

To prove the performance of GSPAE, we compare GSPAE with three typical methods [38] on adaptive offloading and partitioning.

1) *M1:* Each subtask is scheduled for MDs.
2) *M2:* Each subtask is scheduled between MDs and ENs.
3) *M3:* Each subtask is scheduled between MDs and CDC.

Fig. 11 shows the fitness value of each algorithm versus I_k. It is shown that when $I_k<9\,216\,000$, the fitness values of the four methods are all small. The reason is that when I_k is small, subtasks in M1 and M3 are directly executed in MDs. In addition, subtasks in M2 and GSPAE are directly executed in ENs, and smaller fitness values are achieved. When $I_k>9\,216\,000$, MDs fail to perform all subtasks in their latency limits with their limited resources. Thus, the penalty of M1 increases linearly, increasing its fitness value. For M2, M3, and GSPAE, some subtasks are offloaded to ENs and CDC. When $I_k\in[9\,216\,000,\ 12\,288\,000]$, M3 offloads some subtasks to CDC whose price is higher than that of ENs. Thus, M3's cost is higher than that of M2 and GSPAE. When $I_k>12\,288\,000$, M2's penalty increases due to the limited resources of ENs and MDs. Due to the unlimited computing resources in CDC, M2, and GSPAE, both satisfy all constraints. Thus, GSPAE's cost is the lowest, and the increase in its fitness value is also the slowest.

Fig. 12 illustrates the fitness value of each algorithm versus K. The fitness values of M1, M3, and GSPAE increase linearly when K increases and I_k remains unchanged. For M1, the reason is that MDs have limited energy and resources, which increases its penalty. For M3 and GSPAE, the increase in their fitness values is the increase in their cost. M2's fitness

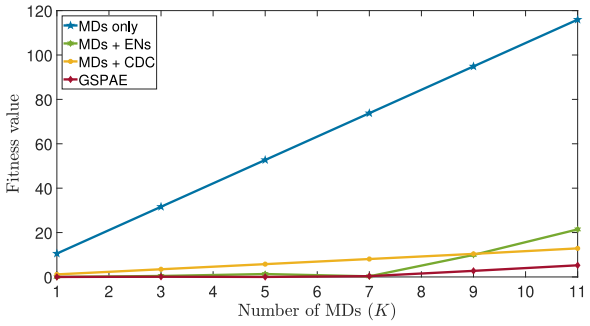


Fig. 12. Fitness value of each algorithm versus K.

value increases suddenly when $K>7$. This is because when K is very large, the energy and resources of MDs and ENs are insufficient to execute all subtasks, and therefore, the penalty of M2 increases. Finally, GSPAE yields the least fitness value, which proves its superiority.

VI. Conclusion and Future Work

ENs are usually on the network's edge and run offloaded MD tasks to support fast services for latency-sensitive applications. However, ENs and MDs usually have constrained computing and communication resources and cannot wholly run offloaded subtasks. Therefore, a collaborative edge and cloud system will utilize a CDC to provide fast services. We aimed to optimize the total cost of the collaborative edge and cloud system by optimizing task split, task offloading, and user associations of dependent subtasks. We presented an MINLP problem and propose a novel meta-heuristic optimization algorithm named GSPAE to solve it. GSPAE obtains a near-optimal mechanism to determine association relations between ENs and MDs and offloading ratios of subtasks in MDs, ENs, and CDC. Extensive experiments demonstrate that GSPAE reduces the total cost by at least 17% compared to typical benchmark methods while adhering to latency limits and resource capacities for energy, communication, and computing of both ENs and MDs.

In the future, we plan to consider more complicated and collaborative edge and cloud systems and design more cost-effective user associations and task-offloading strategies. In addition, more deep learning models, e.g., stacked sparse autoencoders, and radial basis functions can be further used to improve our proposed GSPAE. Besides, we will integrate surrogate models to predict fitness values of low-dimensional particles to reduce the number of computations of costly fitness values.

References

[1] 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 Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[2] C. Brecher, M. Buchsbaum, and S. Storms, "Control from the cloud: Edge computing, services, and digital shadow for automation technologies," in *Proc. Int. Conf. Robot. Autom.*, Montreal, QC, Canada, 2019, pp. 9327–9333.

[3] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multihop offloading of multiple DAG tasks in collaborative edge computing," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4893–4905, Mar. 2021.

[4] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultradense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[5] F. Y. Wu and H. H. Asada, "Decoupled motion control of wearable robot for rejecting human induced disturbances," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 4103–4110.

[6] J. Bi, H. Yuan, M. Zhou, and Q. Liu, "Time-dependent cloud workload forecasting via multitask learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2401–2406, Jul. 2019.

[7] C. T. Recchiuto and A. Sgorbissa, "A feasibility study of culture-aware cloud services for conversational robots," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6559–6566, Oct. 2020.

[8] R. A. C. da Silva and N. L. S. da Fonseca, "Location of fog nodes for reduction of energy consumption of end-user devices," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 593–605, Jun. 2020.

[9] L. Pan, "ECG: Edge-aware point cloud completion with graph convolution," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4392–4398, Jul. 2020.

[10] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5506–5519, Aug. 2018.

[11] X. Wang, J. Ye, and J. C. S. Lui, "Online learning-aided decentralized multiuser task offloading for mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 3328–3342, Apr. 2024.

[12] M. Gao, R. Shen, L. Shi, W. Qi, J. Li, and Y. Li, "Task partitioning and offloading in DNN-task enabled mobile edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2435–2445, Apr. 2023.

[13] M. Tajmirriahi, R. Kafieh, Z. Amini, and H. Rabbani, "A lightweight MIMIC convolutional auto-encoder for denoising retinal optical coherence tomography images," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–8, 2021.

[14] Y. Koshka and M. A. Novotny, "Comparison of D-wave quantum annealing and classical simulated annealing for local minima determination," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 2, pp. 515–525, Aug. 2020.

[15] D. Wu, N. Jiang, W. Du, K. Tang, and X. Cao, "Particle swarm optimization with moving particles on scale-free networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 497–506, Mar. 2020.

[16] H. Yuan, Q. Hu, M. Wang, J. Bi, and M. Zhou, "Cost-minimized user association and partial offloading for dependent tasks in hybrid cloud–edge systems," in *Proc. IEEE 18th Int. Conf. Autom. Sci. Eng. (CASE)*, 2022, pp. 1059–1064.

[17] Y. Cong, K. Xue, C. Wang, W. Sun, S. Sun, and F. Hu, "Latency-energy joint optimization for task offloading and resource allocation in MEC-assisted vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 12, pp. 16369–16381, Dec. 2023.

[18] H. Guo, J. Liu, and J. Lv, "Toward intelligent task offloading at the edge," *IEEE Netw.*, vol. 34, no. 2, pp. 128–134, Mar./Apr. 2020.

[19] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao, "Task offloading with network function requirements in a mobile edge-cloud network," *IEEE Trans. Mobile Comput.*, vol. 18, no. 11, pp. 2672–2685, Nov. 2019.

[20] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud–MEC collaborative task offloading scheme with service orchestration," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5792–5805, Jul. 2020.

[21] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3282–3299, Apr. 2020.

[22] E. El Haber, T. M. Nguyen, and C. Assi, "Joint optimization of computational cost and devices energy for task offloading in multitier edge-clouds," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3407–3421, May 2019.

[23] H. Gao, W. Ma, S. He, L. Wang, and J. Liu, "Time-segmented multilevel reconfiguration in distribution network: A novel cloud-edge collaboration framework," *IEEE Trans. Smart Grid*, vol. 13, no. 4, pp. 3319–3322, Jul. 2022.

[24] R. Fantacci and B. Picano, "A matching game with discard policy for virtual machines placement in hybrid cloud-edge architecture for Industrial IoT systems," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7046–7055, Nov. 2020.

[25] T. Q. Dinh, B. Liang, T. Q. S. Quek, and H. Shin, "Online resource procurement and allocation in a hybrid edge-cloud computing system," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2137–2149, Mar. 2020.

[26] S. Mishra, M. N. Sahoo, S. Bakshi, and J. J. P. C. Rodrigues, "Dynamic resource allocation in fog-cloud hybrid systems using multicriteria AHP techniques," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8993–9000, Sep. 2020.

[27] W. Chen, B. Liu, H. Huang, S. Guo, and Z. Zheng, "When UAV swarm meets edge-cloud computing: The QoS perspective," *IEEE Netw.*, vol. 33, no. 2, pp. 36–43, Mar. 2019.

[28] Y. Li, J. Li, Z. Lv, H. Li, Y. Wang, and Z. Xu, "GASTO: A fast adaptive graph learning framework for edge computing empowered task offloading," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 2, pp. 932–944, Jun. 2023.

[29] S. Tuli, G. Casale, and N. R. Jennings, "MCDS: AI augmented workflow scheduling in mobile edge cloud computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 2794–2807, Nov. 2022.

[30] S. Liu, P. Cheng, Z. Chen, W. Xiang, B. Vucetic, and Y. Li, "Contextual user-centric task offloading for mobile edge computing in ultradense network," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5092–5108, Sep. 2023.

[31] F. Khoramnejad, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Energy and delay aware general task dependent offloading in UAV-aided smart farms," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 5, pp. 5033–5048, Oct. 2024.

[32] H. Zhang, H. Wang, Z. Li, D. Wu, R. Wang, and Y. Hu, "Latency guarantee for task computation in wireless-powered cloud radio access networks," *IEEE Internet Things J.*, vol. 10, no. 21, pp. 19199–19207, Nov. 2023.

[33] Y. Li, J. Shen, S. Ji, and Y.-H. Lai, "Blockchain-based data integrity verification scheme in AIoT cloud–edge computing environment," *IEEE Trans. Eng. Manag.*, vol. 71, pp. 12556–12565, 2024.

[34] C. Hou and Q. Zhao, "Optimal task-offloading control for edge computing system with tasks offloaded and computed in sequence," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 2, pp. 1378–1392, Apr. 2023.

[35] Z. Yang et al., "Differentially private federated tensor completion for cloud–edge collaborative AIoT data prediction," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 256–267, Jan. 2024.

[36] J. Bi, H. Yuan, K. Zhang, and M. Zhou, "Energy-minimized partial computation offloading for delay-sensitive applications in heterogeneous edge networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 4, pp. 1941–1954, Oct.–Dec. 2022.

[37] F. Khoramnejad, M. Rasti, H. Pedram, E. Hossain, and S. Valaee, "Load management, power and admission control in downlink cellular OFDMA networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1477–1493, Apr. 2021.

[38] M. Feng, M. Krunz, and W. Zhang, "Joint task partitioning and user association for latency minimization in mobile edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 8108–8121, Aug. 2021.

[39] M. Salehi and E. Hossain, "Federated learning in unreliable and resource-constrained cellular wireless networks," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 5136–5151, Aug. 2021.

[40] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multiserver mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[41] Z. Zhang, L. Li, Z. Li, and H. Li, "Mobile visual search compression with Grassmann manifold embedding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 11, pp. 3356–3366, Nov. 2019.

[42] X. He, H. Xu, X. Xu, Y. Chen, and Z. Wang, "An efficient algorithm for microservice placement in cloud-edge collaborative computing environment," *IEEE Trans. Services Comput.*, vol. 17, no. 5, pp. 1983–1997, Sep. 2024.

[43] A.-J. Ulusoy, F. Pecci, and I. Stoianov, "An MINLP-based approach for the design-for-control of resilient water supply systems," *IEEE Syst. J.*, vol. 14, no. 3, pp. 4579–4590, Sep. 2020.

[44] P. Srivastava and J. Cortés, "Nesterov acceleration for equality-constrained convex optimization via continuously differentiable penalty functions," *IEEE Control Syst. Lett.*, vol. 5, no. 2, pp. 415–420, Apr. 2021.

[45] M. N. Hjelmeland, J. Zou, A. Helseth, and S. Ahmed, "Nonconvex medium-term hydropower scheduling by stochastic dual dynamic integer programming," *IEEE Trans. Sustain. Energy*, vol. 10, no. 1, pp. 481–490, Jan. 2019.

[46] N. Costilla-Enriquez, Y. Weng, and B. Zhang, "Combining Newton–Raphson and stochastic gradient descent for power flow analysis," *IEEE Trans. Power Syst.*, vol. 36, no. 1, pp. 514–517, Jan. 2021.

[47] N. T. Hanh, H. T. T. Binh, N. X. Hoai, and M. S. Palaniswami, "An efficient genetic algorithm for maximizing area coverage in wireless sensor networks," *Inf. Sci.*, vol. 488, pp. 58–75, Jul. 2019.

[48] S. Harford, F. Karim, and H. Darabi, "Generating adversarial samples on multivariate time series using variational autoencoders," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 9, pp. 1523–1538, Sep. 2021.
[49] H. Mazouzi, N. Achir, and K. Boussetta, "DM2-ECOP: An efficient computation offloading policy for multiuser multicloudlet mobile edge computing environment," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–24, Apr. 2019.
[50] Z. Ye, K. Xiao, Y. Ge, and Y. Deng, "Applying simulated annealing and parallel computing to the mobile sequential recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 243–256, Feb. 2019.
[51] Y. Shi, X. Lu, K. Gao, J. Zhu, and S. Wang, "Genetic algorithm-aided OFDM with all index modulation," *IEEE Commun. Lett.*, vol. 23, no. 12, pp. 2192–2195, Dec. 2019.
[52] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li, and J. Li, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

Haitao Yuan (Senior Member, IEEE) received the Ph.D. degree in computer engineering from New Jersey Institute of Technology (NJIT), Newark, NJ, USA, in 2020.

He is currently a Deputy Director with the Department of Science and Technology Innovation, Wenchang International Aerospace City, Hainan, China. He is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include Internet of Things, edge computing, deep learning, data-driven optimization, and computational intelligence algorithms.

Dr. Yuan received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, the Best Paper Award in the 17th ICNSC, and the Best Student Paper Award Nominees in 2024 IEEE SMC. He is an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, IEEE INTERNET OF THINGS JOURNAL, and *Expert Systems With Applications*. He is named in the World's Top 2% of Scientists List.

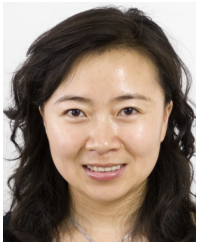
Qinglong Hu (Student Member, IEEE) received the bachelor's degree in automation from Shandong University, Jinan, China, in 2021, and the master's degree in electronic and information engineering from Beihang University, Beijing, China, in 2024. He is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong, Hong Kong.

His research interests include deep learning to optimization, evolutionary computing, big data analysis, machine learning, and large language models.

Shen Wang (Student Member, IEEE) received the B.S. degree in quality and reliability of aircraft from Beihang University, Beijing, China, in 2022, where he is currently pursuing the master's degree with the School of Automation Science and Electrical Engineering.

His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning, and optimization algorithms.

Mr. Wang received the Merit Student Award from Beihang University in 2019.



Jing Bi (Senior Member, IEEE) received the 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 Postdoctoral Researcher with the Department of Automation, Tsinghua University, 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 a Professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing. Her research interests include distributed computing, cloud computing, large-scale data analytics, machine learning, and performance optimization.

Dr. Bi received the IBM Fellowship Award, the Best Paper Award at the 17th IEEE International Conference on Networking, Sensing and Control, and the First-Prize Progress Award of the Chinese Institute of Simulation Science and Technology. She is currently an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS: SYSTEMS.

Rajkumar Buyya (Fellow, IEEE) received the B.E. degree in computer science and engineering from the University of Mysore, Mysuru, India, in 1992, the M.E. degree in computer science and engineering from the University of Bangalore, Bengaluru, India, in 1995, and the Ph.D. in computer science and software engineering from Monash University, Melbourne, VIC, Australia, in 2002.

He is a Redmond Barry Distinguished Professor and the Director of the Cloud Computing and Distributed Systems Laboratory, University of Melbourne, Melbourne. He has authored over 800 publications and seven textbooks.

Dr. Buyya was recognized as a "Web of Science Highly Cited Researcher" from 2016 to 2021 by Thomson Reuters. He is one of the highly cited authors in computer science and software engineering worldwide, with over 146 470 citations and an h-index of 166. He was a Future Fellow of the Australian Research Council from 2012 to 2016.

Jinhu Lü (Fellow, IEEE) received the Ph.D. degree in applied mathematics from the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China, in 2002.

He is currently the Dean of the School of Automation Science and Electrical Engineering, Beihang University, Beijing. He is a Chief Scientist with the National Key Research and Development Program of China, Beijing, and a Leading Scientist with the Innovative Research Groups, National Natural Science Foundation of China, Beijing. His research interests include cooperation control, complex networks, and industrial Internet.

Dr. Lü was a recipient of the prestigious Ho Leung Ho Lee Foundation Award in 2015, the State Natural Science Award three times from the Chinese Government in 2008, 2012, and 2016, the National Natural Science Fund for Distinguished Young Scholars Award, and the Highly Cited Researcher Award from 2014 to 2020. He is a Fellow of CAA.

Jinhong Yang received the Ph.D. degree in computer application technology from Harbin Engineering University, Harbin, China in 2017.

She is a Senior Engineer with the CSSC Systems Engineering Research Institute, Beijing, China. Her research interests include machine learning, data mining, knowledge reasoning, deep learning, and intelligent optimization.

Dr. Yang is a reviewer of *Expert Systems With Applications* and *International Journal of Machine Learning and Cybernetics*.

Jia Zhang (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA, in 2000.

She is the Cruse C. and Marjorie F. Calahan Centennial Chair of Engineering and a Professor with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA. 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 graphs, and interdisciplinary applications of all of these interests in earth science.

MengChu Zhou (Fellow, IEEE) received the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined New Jersey Institute of Technology, Newark, NJ, USA, where he has been a Distinguished Professor since 2013. He has 1300+ publications, including 17 books, 900+ journal papers (680+ in IEEE transactions), 31 patents, and 32 book-chapters. His interests include Petri nets, automation, robotics, big data, Internet of Things, cloud/edge computing, and AI.

Dr. Zhou is a Fellow of IFAC, AAAS, CAA, and NAI.