IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 22 December 2020; revised 10 December 2021; accepted 21 December 2021.
Date of publication 3 January 2022; date of current version 6 December 2022.

Digital Object Identifier 10.1109/TETC.2021.3137980

Energy-Minimized Partial Computation Offloading for Delay-Sensitive Applications in Heterogeneous Edge Networks

JING BI, (Senior Member, IEEE), HAITAO YUAN, (Senior Member, IEEE), KAIYI ZHANG, AND MENGCHU ZHOU, (Fellow, IEEE)

Jing Bi and Kaiyi Zhang are with the School of Software Engineering, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
Haitao Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100190, China
MengChu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA and also with Department of Cyber-Physical Systems, St. Petersburg State Marine Technical University, Lotsmanskaya St., 3, St. Petersburg 198262, Russia
CORRESPONDING AUTHOR: HAITAO YUAN (yuan@buaa.edu.cn)

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 62173013, 61802015, and 62073005, in part by the Major Science and Technology Program for Water Pollution Control and Treatment of China under Grant.

ABSTRACT Mobile Devices (MDs) support various delay-sensitive and computation-intensive applications. Yet they only have limited battery energy and computing resources, thereby failing to totally run all applications. A mobile edge computing (MEC) paradigm has been proposed to provide additional computation, storage, and networking resources for MDs. Servers in MEC are often deployed in both macro base stations (MBSs) and small base stations (SBSs). Thus, it is highly challenging to associate resource-limited MDs to them with high performance, and realize partial computation offloading among them for minimizing total energy consumption of an MEC system. To tackle these challenges, this work proposes a novel computation offloading approach for delay-sensitive applications with multiple separable tasks in hybrid networks including MDs, SBSs, and an MBS. To achieve it, this work formulates total energy consumption minimization as a constrained mixed integer non-linear program. To solve it, this work designs an improved meta-heuristic optimization algorithm called Particle swarm optimization based on Genetic Learning (PGL), which integrates strong local search capacity of a particle swarm optimizer, and genetic operations of a genetic algorithm. PGL jointly optimizes task offloading among MDs, SBSs, and MBS, users' connection to SBSs, MDs' CPU speeds and transmission power, SBSs and MBS, and bandwidth allocation of available channels. Simulations with real-world data collected from Google cluster trace demonstrate that PGL significantly outperforms other existing methods in total energy consumption of an entire system.

INDEX TERMS Computation offloading, mobile edge computing, energy optimization, resource allocation, particle swarm optimization, genetic algorithm

I. INTRODUCTION

The development in 5G and Internet of Things (IoTs) provides enabling technologies to support various applications, e.g., augmented reality, e-Health care, surveillance, and environmental monitoring [1], [2]. Yet, mobile devices (MDs), e.g., smart phones, sensors, and wearable devices, often have limited battery capacity and computing resources. Thus, they may not be able to completely run users' delay-sensitive and computation-intensive applications. To address this problem and prolong their running time, mobile edge computing (MEC) [3] is proposed. It combines advantages of a mobile network and cloud computing to provide a large number of resources including computation, storage, and networking. In MEC, delay-sensitive applications [4] that have strict delay limits can be executed and finished in time by offloading some of them from MDs to nearby servers in MEC.

To provide proximity to users, servers in MEC are densely deployed and distributed in both macro base stations (MBSs) and/or small base stations (SBSs). Therefore, mobile users have a challenge of how to associate themselves to SBSs,

i.e., each of them needs to choose a particular SBS to join for offloading its applications [5]. The strategy of user association has significant effect on performance of task offloading because it directly determines data transmission rates in communication channels. Different from current strategies of user association in existing heterogeneous mobile networks [6], proper user association strategies in MEC rely on many factors, e.g., channel bandwidth, available power, data size of tasks, and delay limits of user applications.

Each application in MEC includes multiple independent tasks that can be logically divided and concurrently executed. For instance, an application of augmented reality includes five tasks, i.e., video collection, object tracking, object mapping, object recognition, and data rendering [7]. Among them, object tracking, mapping, and recognition are all computation-intensive tasks that can be independently offloaded from MDs to their nearby MEC servers. Therefore, it is expected that proper task offloading can reduce MDs' energy consumption and completion time of applications. In addition, there is a specific relation of dependency among tasks, e.g., parallel, sequential, and hybrid dependency [8]. The dependency of tasks specifies their order of running and their delay time. To guarantee that each application with many tasks can be finished before its deadline, it is highly needed to consider dependency of tasks in each application for making the optimal task offloading strategy.

Computation offloading in MEC has received much attention recently. Nevertheless, most of current studies on offloading strategies assume that there is only one single server in MEC, and all MDs have to decide whether and how many tasks should be offloaded. In real-life cases, servers in MEC are distributed in a dense way. Consequently, it is essential to design an improved strategy of computation offloading that considers user association in MEC. Moreover, most of them assume that each application of each user only has one single task, and dependency among tasks has not been fully considered. Therefore, they cannot directly be applied to the offloading of applications with multiple tasks. Currently, only a few studies consider the offloading of multi-task applications. For example, the work [9] proposes an online algorithm for realizing joint resource and radio management for MEC systems with multiple users. It aims to minimize the average power consumed by MDs and an MEC server while ensuring the stability of a task buffer. The work [10] proposes an online multi-task computation offloading method for users in a green MEC system, and adopts a Lyaponuv optimization method to determine the optimal offloading of tasks. The work [11] proposes two strategies to offload applications with multiple tasks for mobile cloud computing, and considers sequential and parallel dependency of tasks. Nevertheless, all of them fail to decrease the consumption of energy by effectively offloading tasks of each application in distributed MEC systems where servers are densely distributed.

To solve the above problems, this work aims to minimize the energy consumed by all MDs, all SBSs, and an MBS for complex applications that consist of multiple separable tasks. It realizes computation offloading for delay-sensitive applications by smartly associating users' tasks with MDs and SBSs. It also considers dependency of tasks in the total completion time of each application. We jointly consider the available CPU and memory resources in SBSs and MBS, the available energy limits in each involved entity, and the bandwidth allocation among multiple available channels. We also consider the allocation of transmission power and computation resources in the proposed computation offloading method.

Specifically, this work aims to make the following contributions to the field of MEC:

1) This work designs a framework of partial computation offloading for multi-task delay-sensitive applications in heterogeneous edge networks, where MDs can offload their tasks to SBSs and/or MBS in an energy-minimized manner, respectively.
2) It minimizes the total energy consumption of an MEC system including heterogeneous edge networks, and formulates this problem as a constrained mixed integer non-linear program (MINLP).
3) It solves MINLP by proposing an improved meta-heuristic optimization algorithm called Particle swarm optimization based on Genetic Learning (PGL). PGL combines advantages of strong local search capacity in particle swarm optimization (PSO) [12], and stochastic and global search ability of genetic algorithm (GA) [13]. PGL jointly optimizes task offloading, user association, resource allocation, and transmission power among MDs, SBSs and MBS.

The related work is given in Section 2. Section 3 presents a proposed architecture of computation offloading for multi-task applications in hybrid networks. Section 4 formulates a problem of total energy consumption minimization. Section 5 gives details of PGL. Section 6 compares its results with other method's. Section 7 concludes this work.

II. RELATED WORK

A. COMPUTATION OFFLOADING

Recently, computation offloading has received an increasing amount of attention [14]–[21]. Ning *et al.* [14] design a three-layer framework of offloading in Internet of Vehicles, and propose a deep reinforcement learning method to reduce energy consumed by vehicles such that users' delay limits are met. Ning *et al.* [15] propose an heuristic iterative algorithm for resource allocation in MEC, which dynamically offloads tasks among MDs, MEC and cloud servers. They also propose an efficient computation offloading algorithm to maximize the system-wide profit with convex optimization and a non-cooperative game [16]. Wu *et al.* [17] propose a two-step method to minimize total energy consumed by smart terminals for executing tasks. An optimization problem is formulated and it jointly considers task offloading, allocation of local resources, and transmission duration in nonorthogonal multiple access. Then, different tasks are optimally and properly offloaded to edge servers. Xie *et al.* [18] design a pricing strategy for multi-dimensional resources in a green mobile edge cloud by using a two-sided market game. Then, a distributed algorithm is proposed to achieve efficient

allocation of resources and quality-of-service-aware application offloading. It aims to maximize the total system benefit. Different from it, this work aims to minimize the total energy consumption of heterogeneous edge networks. Chen *et al.* [19] achieve computation offloading at a mobile edge cloud system consisting of multiple wireless devices, each of which simultaneously fetches renewable energy from its nearby environment and offloads tasks to cloud server. A computation offloading problem of multiple users and tasks is formulated, and solved by using a method of Lyaponuv Optimization. Finally, distributed and centralized greedy maximal scheduling algorithms are proposed to efficiently achieve task offloading. Nevertheless, their work [19] considers a two-tier system including MDs and a remote cloud, which is different from our work. Zhao *et al.* [20] present a method of collaborative offloading for automobiles in vehicular networks by using MEC and cloud computing techniques. A computation offloading problem in a cloud-MEC system is stated for optimizing computation offloading and resource allocation. A distributed algorithm is presented to decrease computation time and increase system utility. However, different from this study, we aim to achieve the minimization of the total energy consumption in heterogeneous edge networks. Guo *et al.* [21] propose an approach to a computation offloading problem in an MEC system consisting of small cell networks. An MINLP is formulated to minimize the energy consumption of all MDs and tackled by a hybrid algorithm by integrating GA and PSO. Different from it, our work considers heterogeneous edge networks including MDs, SBSs and MBS, and minimizes their total energy consumption.

Different from them, this work proposes an improved partial computation offloading method for computation-intensive applications with multiple tasks in hybrid networks including MDs, SBSs, and MBS. It considers completion-time-guaranteed applications, available transmission power of MDs, maximum number of computation resources, maximum CPU speed, and available energy in MDs, SBSs, and MBS.

B. RESOURCE ALLOCATION IN MEC

Resource allocation is an important topic for realizing energy-efficient and smart MEC [22]–[27]. Nguyen *et al.* [22] present a collaborative task computing strategy in which an MEC server attracts nearby resource-rich servers to execute allocated computing tasks. The minimal cost is achieved by efficiently decreasing redundant data and utilizing resources in nearby MEC servers. Ali *et al.* [23] propose a deep learning-based resource allocation algorithm that schedules requests to the optimal server and determines the optimal number of resources to user equipments. Nevertheless, these methods are not designed for heterogeneous edge networks. Na *et al.* [24] present an IoT system including edge gateways, an edge server, and MDs that are connected in a wireless way. A resource orchestration method is designed for the optimal allocation of resources by using a Lagrangian method. Although they evaluate the performance of resource allocation with respect to latency, throughput, data reception rate and fairness of workload, they fail to minimize total energy consumption. Wang *et al.* [25] design a system of three-layer data flow processing to use computing resources in a cloud, MEC servers, and MDs. The resources of wireless transmission are optimally allocated to minimize the system latency. Different from it, our work aims to achieve the minimal total energy consumed by multi-task applications while ensuring their timely completion. Liu *et al.* [26] design a mechanism for an MEC system with a vehicle-mounted edge system. The resource allocation is formulated as MINLP, which is converted into a mixed integer linear one. Then, piecewise linear relaxation and approximation are adopted to find a near-optimal solution. However, it aims to maximize the number of completed tasks with sensitive deadlines in the system. Zhan *et al.* [27] consider the resource allocation for the joint minimization of completion time and energy consumption of applications in unmanned aerial vehicles (UAVs) provided that energy limits are met. Specially designed for an UAV-enabled MEC system, it aims to jointly balance its task completion time and energy consumption. Different from it, our work considers energy-minimized resource allocation for heterogeneous edge networks including MDs, SBSs and MBS.

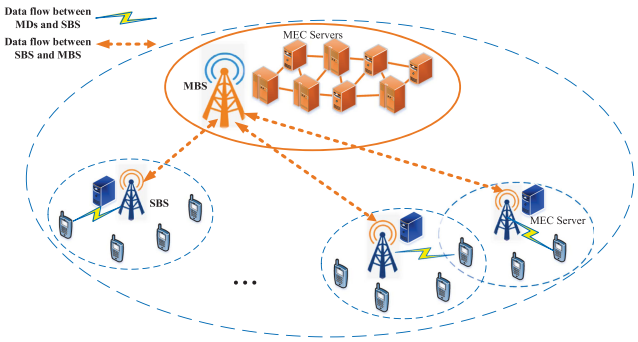

FIGURE 1. The proposed computation offloading architecture for multi-task applications in a hybrid MEC system.

Different from these mentioned studies, this work aims to achieve the minimal energy consumed by MDs, SBSs, and MBS by considering dependency of tasks in each application. It jointly optimizes a ratio of task offloading among MDs, SBSs, and MBS, users' association with SBSs, CPU speeds of MDs, and SBSs, transmission power of MDs and allocation of channel bandwidth.

III. MOTIVATION

Figure 1 illustrates a proposed architecture of computation offloading for multi-task applications in a hybrid MEC system including M MDs, J SBSs, and a single MBS. M MDs send their tasks through a single transmission channel, provides functions of uplink and downlink. Each MD is only connected to a single SBS. Each MD runs a delay-sensitive and computation-intensive application, which needs to be completed before a given deadline. This work considers such

applications as vehicle navigation [28], which enables data partition provided by edge computing.

Similar to [29], [30], this work considers partition-oriented applications, e.g., figure compression, augmented reality and virus scan. Each application of this kind can be split into multiple logically independent tasks. For example, an application of augmented reality includes five tasks, i.e., video production, recognizer, mapper, tracker, and renderer [31]. Among them, recognizer, mapper, and tracker are computation-intensive parts that can be offloaded from MDs to SBSs/MBS for processing. Similar to [31], [32], each task can be split arbitrarily into three partitions to provide parallelism, i.e., each partition can be executed in MDs, SBSs and MBS concurrently. As shown in Figure 1, users' tasks of such applications can be completed in their own MDs, SBSs or MBS.

We assume that tasks completed in MDs and those completed in SBSs/MBS are executed in a parallel manner [33]. This work designs a task scheduler that obtains information about MDs, channels of uplink/downlink, users' completion time needs, and available resources in SBSs and MBS. The scheduler runs the proposed PGL to achieve the optimized partial computation offloading among MDs, SBSs, and MBS, users' association with SBSs, CPU speeds of MDs, and SBSs, transmission power and channel bandwidth.

In Figure 1, each task can be divided into three parts to provide parallelism. One part is run in the MDs, the second one is offloaded to a user-associated SBS, and the third one can be further offloaded to MBS. This is because computation resources in each SBS is limited and MBS usually has much more computation resources. Therefore, each offloaded task of each MD is first transmitted to its associated SBS, and some of tasks are further transmitted to MBS by its associated SBS. MBS is connected to all J SBSs *via* relatively fixed (e.g., optical) links [34]. MDs and SBSs are randomly distributed in the coverage area of MBS. It is assumed that each application consists of K independent tasks.

In addition, different SBSs may have different computation resource capacity and channel bandwidth. Thus, it is important to connect an MD to a suitable SBS provided that tasks of MDs can be finished within their latency constraints and their consumption of energy can be optimized accordingly. This work achieves partial computation offloading, user association, and the setting of MDs and SBSs for multi-task applications to achieve the minimization of total energy consumption for all three entities, i.e., MDs, SBSs, and MBS.

IV. PROBLEM FORMULATION

This section states a total energy consumption minimization problem for hybrid networks including MDs, SBSs, and MBS. Then, energy consumption and completion time models are presented. For clarity, main notations are summarized in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TETC.2021.3137980. Let M denote the number of MDs. Let λ_m^{k1}, λ_m^{k2} and λ_m^{k3} ($0 \le \lambda_m^{k1} \le 1$, $0 \le \lambda_m^{k2} \le 1$ and $0 \le \lambda_m^{k3} \le 1$) denote the portion of task k ($1 \le k \le K$) executed in MD m ($1 \le m \le M$), SBSs, and MBS, respectively. For task k received by MD m, we have

$$\lambda_m^{k1} + \lambda_m^{k2} + \lambda_m^{k3} = 1. \tag{1}$$

This work adopts the communication model of orthogonal frequency division multiple access between each MD and its associated SBS. It is assumed that the spectrums between MDs and its associated SBS are allocated in an orthogonal manner. Similarly, spectrums between SBSs and MBS are also orthogonal [32]. Similar to [29], this work does not consider inter-channel or inter-SBS interferences. It is assumed that each channel supports frequency-flat and block-fading Rayleigh. According to [29], the loss of path between MD m and its corresponding SBS j ($1 \le j \le J$) is $(d_{mj})^{-v}$. Here, d_{mj} means the distance from MD m to SBS j, and v means a path loss parameter. h_1 and h_2 mean fading coefficients of uplink and downlink channels, respectively. They are modelled as random variables that are produced by using the circularly symmetric complex Gaussian distribution. G_0 means a power parameter of white Guassian noise. R_{mj}^U and R_{mj}^D mean uplink and downlink rates between MD m and SBS j, respectively. Then, we have

$$R_{mj}^U = \mu_{mj} W_{Uj} \log_2 \left(1 + \frac{P_m^t (d_{mj})^{-v} |h_1|^2}{G_0} \right) \tag{2}$$

$$R_{mj}^D = \mu_{mj} W_{Dj} \log_2 \left(1 + \frac{P_{Fj}^t (d_{mj})^{-v} |h_2|^2}{G_0} \right), \tag{3}$$

where W_{Uj} and W_{Dj} denote the bandwidth of uplink and downlink channels for SBS j, P_m^t denotes the transmission power of MD m, P_{Fj}^t denotes the transmission power of SBS j for each MD, and μ_{mj} ($0 \le \mu_{mj} \le 1$) is the bandwidth in SBS j's uplink and downlink channels used for MD m.

Each MD is only connected to a single SBS. Then, if tasks from MD m are processed by SBS j, $x_{mj} = 1$; otherwise, $x_{mj} = 0$. Then, for each MD m, we have

$$\sum_{j=1}^{J} x_{mj} = 1. \tag{4}$$

All MDs connected by SBS j share the bandwidth of its channels. Then, for each SBS j, we have

$$\sum_{m=1}^{M} x_{mj} \mu_{mj} = 1. \tag{5}$$

A. COMPLETION TIME MODEL

T_m^k denotes the time of executing task k in MD m. Let I_m^k denote the size (bit number) of total input data of task k received by MD m. The number of bits of task k executed in MD m is $I_m^k \lambda_m^{k1}$. Then,

$$T_m^k = \frac{I_m^k \lambda_m^{k1} \alpha_m^k}{f_m^k}, \tag{6}$$

where α_m^k ($\alpha_m^k > 0$) denotes the number of CPU cycles required by each bit of the input data of task k executed in MD m. f_m^k denotes the running speed (cycles/sec.) for executing task k in MD m.

Let T_{mj}^k denote the time of executing task k of MD m in SBS j and MBS. It is worth noting that tasks of MD m are first transmitted to a specific SBS, and then some of them are further delivered from it to MBS. Let $\hat{T}_{mj}^k$ denote the total time needed by data uploading, downloading, and processing for task k of MD m in SBS j. Let $\tilde{T}_{mj}^k$ denote the total time needed by data uploading, downloading, and processing for task k of MD m in MBS. Thus, T_{mj}^k is the sum of $\hat{T}_{mj}^k$ and $\tilde{T}_{mj}^k$, i.e.,

$$T_{mj}^k = \hat{T}_{mj}^k + \tilde{T}_{mj}^k. \tag{7}$$

Let β_1 ($\beta_1 > 0$) denote the overhead of data transmission in each uplink channel from each MD to each SBS [30]. Let β_2 ($\beta_2 > 0$) denote the overhead of data transmission in each downlink channel from each SBS to each MD. f_{mj}^k denotes the running speed (cycles/sec.) for executing task k from MD m in SBS j. Then, the data uploading time of task k from MD m to SBS j is $\beta_1(\lambda_m^{k2} + \lambda_m^{k3}) I_m^k / R_{mj}^U$. Similarly, the data downloading time of task k from SBS j to MD m is $\beta_2(\lambda_m^{k2} + \lambda_m^{k3}) I_m^k / R_{mj}^D$. The processing time of task k of MD m in SBS j is $\lambda_m^{k2} I_m^k \alpha_m^k / f_{mj}^k$. Thus, we have

$$\hat{T}_{mj}^k = \frac{\beta_1(\lambda_m^{k2} + \lambda_m^{k3}) I_m^k}{R_{mj}^U} + \frac{\lambda_m^{k2} I_m^k \alpha_m^k}{f_{mj}^k} + \frac{\beta_2(\lambda_m^{k2} + \lambda_m^{k3}) I_m^k}{R_{mj}^D}. \tag{8}$$

In addition, SBSs and MBS are connected by fiber transmission with low latency [35], [36]. r_0 denotes the transmission rate (bits/sec.) of backhaul among SBSs and MBS. f_0 denotes the computational speed of MBS, which is a given value for each MD. Then, the data uploading time of task k in MD m from SBS j to MBS is $\beta_3 \lambda_m^{k3} I_m^k / r_0$. Similarly, the data downloading time of task k in MD m from MBS to SBS j is $\beta_4 \lambda_m^{k3} I_m^k / r_0$. The processing time of task k in MD m in MBS is $\lambda_m^{k3} I_m^k \alpha_m^k / f_0$. Thus, we have

$$\tilde{T}_{mj}^k = \frac{\beta_3 \lambda_m^{k3} I_m^k}{r_0} + \frac{\lambda_m^{k3} I_m^k \alpha_m^k}{f_0} + \frac{\beta_4 \lambda_m^{k3} I_m^k}{r_0}. \tag{9}$$

Each task k of MD m can be executed in MD m, SBSs, and MBS concurrently. Thus, its latency is the maximum value of T_m^k and $\sum_{j=1}^{J} x_{mj} T_{mj}^k$. It is worth noting that T_{mj}^k is only counted if $x_{mj} = 1$ according to (4) and at most one SBS may handle task k. Let T_m denote the total completion time needed by executing all K tasks of applications in MDs, SBSs, and MBS. Then, T_m can be calculated as

$$T_m = \sum_{k=1}^{K} \max\left(T_m^k, \sum_{j=1}^{J} x_{mj} T_{mj}^k \right). \tag{10}$$

Let T_m^* denote the upper limit of T_m. Thus, T_m cannot exceed T_m^*, i.e.,

$$T_m \le T_m^*. \tag{11}$$

B. ENERGY CONSUMPTION MODEL

1) Energy Consumption Model of MDs

P_m^k denotes the amount of power consumption of executing a part of task k in MD m. According to [29], P_m^k is obtained as

$$P_m^k = \varsigma_m \left(f_m^k\right)^3, \tag{12}$$

where ς_m denotes a constant determined by chip architectures of MD m.

According to (6) and (12), the energy consumed by executing task k in MD m per cycle is $\varsigma_m \left(f_m^k\right)^2$. Let E_m denote the energy consumption of executing all K tasks in MD m. According to [29], E_m is obtained as

$$E_m = \sum_{k=1}^{K} \left(\varsigma_m I_m^k \lambda_m^{k1} \alpha_m^k \left(f_m^k\right)^2 \right). \tag{13}$$

$\hat{E}_m$ denotes the maximum amount of available energy in MD m. Thus, E_m cannot exceed $\hat{E}_m$, i.e.,

$$\sum_{k=1}^{K} \left(\varsigma_m I_m^k \lambda_m^{k1} \alpha_m^k \left(f_m^k\right)^2 \right) \le \hat{E}_m. \tag{14}$$

Let F_m denote a maximum CPU speed of MD m. Thus, the total sum of running speeds for executing all tasks in MD m cannot exceed F_m, i.e.,

$$\sum_{k=1}^{K} f_m^k \le F_m,\ f_m^k \in N^+. \tag{15}$$

2) Energy Consumption Model of Data Transmission

P_m^0 denotes the power consumption of MD m when it is in the idle state. ρ_m is the a power amplifier coefficient for uploading data from MD m. P_m^t denotes the power consumed by uploading data from MD m. Then, the power consumed by MD m when it uploads data to SBS j is $(P_m^0 + \rho_m \mu_{mj} P_m^t)$. Let P_m^r denote the power consumption when it downloads data from SBSs to MD m. ϕ_{mj}^1 denotes the energy consumption of uploading and downloading data between MD m and SBS j. ϕ_{mj}^1 is calculated as

$$\phi_{mj}^1 = \sum_{k=1}^{K} \left((P_m^0 + \rho_m \mu_{mj} P_m^t) \frac{\beta_1(\lambda_m^{k2} + \lambda_m^{k3}) I_m^k}{R_{mj}^U} + \frac{P_m^r \beta_2(\lambda_m^{k2} + \lambda_m^{k3}) I_m^k}{R_{mj}^D} \right). \tag{16}$$

In addition, let $\hat{P}_m^t$ denote the upper limit of P_m^t. Then,

$$0 \le P_m^t \le \hat{P}_m^t. \tag{17}$$

3) Energy Consumption Model of SBSs/MBS

Let E_{mj} denote the energy consumption of executing MD m's offloaded tasks in SBS j and MBS. E_{mj} consists of two parts, i.e., ϕ_{mj} and ϕ_{m0}. ϕ_{mj} denotes the energy consumption of executing MD m's offloaded tasks in SBS j. ϕ_{m0} denotes the energy consumption of executing MD m's offloaded tasks in MBS. E_{mj} is obtained as

$$E_{mj} = \phi_{mj} + \phi_{m0}. \tag{18}$$

Let ϕ^2_{mj} denote the amount of energy consumption of executing MD m's offloaded tasks in SBS j. Then,

$$\phi^2_{mj} = \sum_{k=1}^{K} \left(\varsigma_j I^k_m \lambda^{k2}_m \alpha^k_m (f^k_{mj})^2 \right), \tag{19}$$

where ς_j denotes a constant determined by the chip architecture of SBS j.

ϕ_{mj} consists of two parts, i.e., ϕ^1_{mj} and ϕ^2_{mj}. Then,

$$\phi_{mj} = \phi^1_{mj} + \phi^2_{mj}. \tag{20}$$

P^1_δ denotes the power consumed by data transmission from each SBS to MBS. P^2_δ denotes the power consumed by data transmission from MBS to each SBS. Let β_3 ($\beta_3 > 0$) denote the overhead of data transmission from each SBS to MBS. Let β_4 ($\beta_4 > 0$) denote the overhead of data transmission from MBS to each SBS. Let e_0 denote the amount of energy consumed by each CPU cycle in MBS. Then, ϕ_{m0} is obtained as

$$\phi_{m0} = \sum_{k=1}^{K} \left(\frac{P^1_\delta \beta_3 \lambda^{k3}_m I^k_m}{r_0} + \frac{P^2_\delta \beta_4 \lambda^{k3}_m I^k_m}{r_0} + I^k_m \lambda^{k3}_m \alpha^k_m e_0 \right). \tag{21}$$

$\hat{E}_j$ is the maximum amount of available energy in SBS j. The total amount of energy consumed by MDs' tasks offloaded to SBS j cannot exceed $\hat{E}_j$. Thus,

$$\sum_{m=1}^{M} x_{mj} \phi_{mj} \le \hat{E}_j. \tag{22}$$

$\hat{E}_0$ is the maximum amount of available energy in MBS, which cannot be exceeded by the total amount of energy consumed by MDs' tasks offloaded to MBS. Thus,

$$\sum_{m=1}^{M} \phi_{m0} \le \hat{E}_0. \tag{23}$$

Let $\tilde{F}_j$ denote a maximum CPU speed of SBS j. Thus, the total running speeds for executing all tasks from all MDs cannot exceed $\tilde{F}_j$, i.e.,

$$\sum_{m=1}^{M} \sum_{k=1}^{K} x_{mj} f^k_{mj} \le \tilde{F}_j, \ f^k_{mj} \in N^+. \tag{24}$$

4) CPU and Memory Modeling in SBSs and MBS

Let $\hat{A}^j_1$ and $\hat{G}^j_1$ denote the maximum number of CPU cycles and the maximum amount of memory in SBS j, respectively. Then, the total number of CPU cycles and the total amount of memory consumed by MDs' tasks offloaded to SBS j cannot exceed their corresponding limits, i.e.,

$$\sum_{m=1}^{M} \sum_{k=1}^{K} \left(x_{mj} I^k_m \lambda^{k2}_m \alpha^k_m \right) \le \hat{A}^j_1 \tag{25}$$

$$\sum_{m=1}^{M} \sum_{k=1}^{K} \left(x_{mj} I^k_m \lambda^{k2}_m \chi^k_m \right) \le \hat{G}^j_1, \tag{26}$$

where χ^k_m denotes the amount of memory for executing each bit of data of task k of MD m.

Let $\hat{A}_2$ and $\hat{G}_2$ denote the maximum number of CPU cycles and the maximum amount of memory in MBS, respectively. Then, the total number of CPU cycles and the total amount of memory consumed by all tasks from MDs executed in MBS cannot exceed their corresponding limits, i.e.,

$$\sum_{m=1}^{M} \sum_{k=1}^{K} \left(I^k_m \lambda^{k3}_m \alpha^k_m \right) \le \hat{A}_2 \tag{27}$$

$$\sum_{m=1}^{M} \sum_{k=1}^{K} \left(I^k_m \lambda^{k3}_m \chi^k_m \right) \le \hat{G}_2. \tag{28}$$

5) Total Cost Model

Δ denotes the total energy consumption of all tasks executed in MDs, SBSs, and MBS, i.e.,

$$\Delta = \sum_{m=1}^{M} \left(E_m + \sum_{j=1}^{J} x_{mj} E_{mj} \right). \tag{29}$$

C. OPTIMIZATION PROBLEM

This work presents the formulation of a constrained problem of total energy consumption minimization as follows. In (30), $\hbar$ denotes a vector of decision variables including λ^{k1}_m, λ^{k2}_m, λ^{k3}_m, x_{mj}, f^k_m, f^k_{mj}, P^t_m and μ_{mj}.

$$\arg \ \mathbf{Min}_{\hbar} \left\{ \Delta = \sum_{m=1}^{M} \left(E_m + \sum_{j=1}^{J} x_{mj} E_{mj} \right) \right\} \tag{30}$$

subject to

$$\sum_{k=1}^{K} \mathbf{max} \left(T^k_m, \sum_{j=1}^{J} x_{mj} T^k_{mj} \right) \le T^*_m \tag{31}$$

$$\lambda_m^{k1}+\lambda_m^{k2}+\lambda_m^{k3}=1 \tag{32}$$

$$\sum_{j=1}^{J} x_{mj}=1 \tag{33}$$

$$x_{mj}\in\{0,1\} \tag{34}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{K}\left(x_{mj}I_m^k\lambda_m^{k2}\alpha_m^k\right)\le\hat{A}_1^j \tag{35}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{K}\left(x_{mj}I_m^k\lambda_m^{k2}\chi_m^k\right)\le\hat{G}_1^j \tag{36}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{K}\left(I_m^k\lambda_m^{k3}\alpha_m^k\right)\le\hat{A}_2 \tag{37}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{K}\left(I_m^k\lambda_m^{k3}\chi_m^k\right)\le\hat{G}_2 \tag{38}$$

$$\sum_{k=1}^{K}\left(\varsigma_m I_m^k\lambda_m^{k1}\alpha_m^k\left(f_m^k\right)^2\right)\le\hat{E}_m \tag{39}$$

$$\sum_{m=1}^{M}x_{mj}\phi_{mj}\le\hat{E}_j \tag{40}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{K}\left(\frac{P_\delta^1\beta_3\lambda_m^{k3}I_m^k}{r_0}+\frac{P_\delta^2\beta_4\lambda_m^{k3}I_m^k}{r_0}+I_m^k\lambda_m^{k3}\alpha_m^k e_0\right)\le\hat{E}_0 \tag{41}$$

$$\sum_{m=1}^{M}x_{mj}\mu_{mj}=1 \tag{42}$$

$$0\le\mu_{mj}\le 1 \tag{43}$$

$$0\le P_m^t\le\hat{P}_m^t \tag{44}$$

$$\sum_{k=1}^{K}f_m^k\le F_m,\ f_m^k\in N^+ \tag{45}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{K}x_{mj}f_{mj}^k\le\tilde{F}_j,\ f_{mj}^k\in N^+. \tag{46}$$

Here, x_{mj}, f_m^k and f_{mj}^k are discrete integer variables. Besides, λ_m^{k1}, λ_m^{k2}, λ_m^{k3}, P_m^t and μ_{mj} are continuous variables. It is clear that constraints (42) and (46) are nonlinear with respect to x_{mj}, μ_{mj} and f_{mj}^k. Constraint (39) is nonlinear with respect to f_m^k. Constraint (2) is nonlinear with respect to P_m^t, and therefore, constraints (7), (8), and (31) are nonlinear with respect to it. Consequently, this problem of the total energy consumption minimization is a constrained MINLP. It is proven that its solution complexity is NP-hard [37], and there are no available polynomial-time methods [38].

To handle these constraints, we apply a penalty function method [39] to transform each constraint into a positive penalty. Then, the above constrained problem is transformed into an unconstrained one, i.e.,

$$\underset{\hbar}{\mathbf{Min}}\left\{\widetilde{\Delta}=\overset{\infty}{\mathcal{N}}\Omega+\Delta\right\}. \tag{47}$$

In (47), $\widetilde{\Delta}$ is a transformed new objective function, which is used to calculate the fitness value of each solution in PGL. $\mathcal{N}$ is a positive constant that is very large. Ω is the sum of penalties of all constraints. To handle constraints (31), (32), (33), (34), (35), (36), (37), (38), (39), (40), (41), (42), (43), (44), (45), and (46), each equality or inequality constraint is first converted into a non-negative penalty. If a constraint is strictly met, its penalty is 0; otherwise, its penalty is positive. Thus, Ω is a positive value if any constraint is not met. As $\mathcal{N}$ is a very large constant, e.g., 10^{10}, $\widetilde{\Delta}$ is extremely large if Ω is not 0. PGL aims to minimize $\widetilde{\Delta}$, and therefore, it gradually converges towards the optimal solution that meets all constraints, i.e., Ω=0, in its search process. The rationale is that the introduction of Ω into $\widetilde{\Delta}$ allows some infeasible solutions in the initial search process to speed up the convergence process. It is time-consuming if we only allow feasible solutions that can be difficult to be obtained in each iteration. Specifically, Ω is obtained as

$$\Omega=\sum_{p=1}^{\mathcal{N}^{\neq}}\left(\max\{0,-g_p(x)\}\right)^2+\sum_{q=1}^{\mathbb{N}^{=}}\left|h_q(x)\right|^2. \tag{48}$$

In (48), $\mathbb{N}^{=}$ and $\mathcal{N}^{\neq}$ are the number of equality and inequality constraints. Each inequality constraint p $(1\le p\le\mathcal{N}^{\neq})$ is transformed into $g_p(\boldsymbol{x})\ge 0$. If it is not met, its penalty is $(\max\{0,-g_p(\boldsymbol{x})\})^2$; otherwise, its penalty is 0. Each equality constraint q $(1\le q\le\mathbb{N}^{=})$ is transformed into $h_q(\boldsymbol{x})=0$. If it is not met, its penalty is $\left|h_q(\boldsymbol{x})\right|^2$; otherwise, its penalty is 0. For example, (40) is transformed into $\hat{E}_j-\sum_{m=1}^{M}x_{mj}\phi_{mj}\ge 0$, and its penalty is $(\max\{0,-(\hat{E}_j-\sum_{m=1}^{M}x_{mj}\phi_{mj})\})^2$. Other inequality constraints are transformed similarly. (42) is converted into $\sum_{m=1}^{M}x_{mj}\mu_{mj}-1=0$, and its penalty is $\left|\sum_{m=1}^{M}x_{mj}\mu_{mj}-1\right|^2$. Other equality constraints are transformed similarly. Once Ω is obtained, constraints (31), (32), (33), (34), (35), (36), (37), (38), (39), (40), (41), (42), (43), (44), (45), and (46) are all handled. Then, $\widetilde{\Delta}$ in (47) is further obtained. In this way, the original problem (30), (31), (32), (33), (34), (35), (36), (37), (38), (39), (40), (41), (42), (43), (44), (45), and (46) is converted into (47).

To solve the unconstrained problem, this work designs an improved meta-heuristic optimization algorithm, and its details are given next.

V. PARTICLE SWARM OPTIMIZATION BASED ON GENETIC LEARNING (PGL)

There are many widely used mathematical and deterministic methods, e.g., dynamic programming [40], and conjugate gradient descent [41] to address the unconstrained problem.

Nevertheless, they usually require that objective functions have some specific mathematical structures. For example, the second-order derivatives are needed. In addition, their solutions are often low-quality for solving complex problems given limited time. Typical meta-heuristic optimization methods own several advantages, e.g., quick convergence, ability to tackle complex nonlinearities and discontinuities, good robustness and easy implementation. Thus, they are widely used by many existing studies to tackle above-mentioned drawbacks and solve various complicated real-life problems. However, each meta-heuristic method has its advantages and disadvantages.

Specifically, the convergence of PSO is fast; yet it often easily converges to local optima when solving complex problems with high-dimension and complex solution spaces [42]. The search process of PSO oscillates if a locally best position of each particle and a globally best position of all particles vary significantly. On the other hand, genetic operations in GA, e.g., crossover, mutation, and selection, can produce diverse individuals, thus enhancing both search efficiency and accuracy [43].

It is worth noting that most of existing similar algorithms split the population into two independent subpopulations, each evolves with GA and PSO, respectively. The subpopulations are then recombined together. In these hybrid algorithms, GA and PSO are connected in a loose way, and it is difficult to recognize their interaction and effect. For example, it is difficult to judge how individuals produced from GA affect individuals in PSO. Similarly, it is also hard to characterize whether individuals migrated from PSO survive in GA.

To enhance PSO's performance, this work combines GA's genetic operations and PSO, and designs a hybrid algorithm named PGL. PGL provides a genetic learning manner that adopts GA for yielding superior exemplars. In PGL, PSO and GA are combined in a highly cascade and cohesive way. PGL includes two cascading layers: the first for yielding superior exemplars with GA and the second for updating particles in PSO. In PGL, each particle is guided by the locally and globally best solutions, and superior exemplars yielded by GA. Main advantages of PGL are summarized as follows. First, search of particles learns from superior exemplars of GA, and it is more diversified, thereby alleviating the premature convergence problem of PSO. Second, superior exemplars survived from the selection operation in GA are high-quality, and they can effectively guide the search of particles. Third, the search history of particles propagates high-quality genetic information back to GA and helps GA yield better superior exemplars. Therefore, in PGL, the interaction between GA and PSO is clear, and it improves GA and PSO alternatively in a search process. In summary, each particle in PSO adopts the operations of crossover, mutation, and selection in GA in each generation to yield a high-quality and good-diversity superior exemplar, and learns from it for higher exploitation and exploration.

$|\mathbb{X}|$ denotes the total number of particles in the population. p_i denotes a locally best position of particle i ($1 \le i \le |\mathbb{X}|$). g denotes a globally best position of all particles. D is the dimension of each individual. e_i denotes the superior exemplar corresponding to particle i, and it is a linear combination of p_i and g. $e_{i,d}$ denotes the dth ($1 \le d \le D$) entry of e_i. $p_{i,d}$ denotes the dth entry of p_i. g_d denotes the dth entry of g. Then,

$$e_{i,d} = \frac{c_1 \cdot r_1 \cdot p_{i,d} + c_2 \cdot r_2 \cdot g_d}{c_1 \cdot r_1 + c_2 \cdot r_2}, \tag{49}$$

where c_1 denotes a cognitive acceleration parameter reflecting the effect of $p_{i,d}$, c_2 denotes a social acceleration parameter reflecting the effect of g_d, and r_1 and r_2 denote two random numbers that are uniformly distributed in (0,1), i.e., $r_1, r_2 \in (0,1)$.

In each iteration of PGL, there are four core operations, i.e., crossover, mutation, and selection for exemplars, and the update of positions of particles. They are described as follows. $\widetilde{\Delta}(\hbar_i)$ denotes the value of the unconstrained objective function of particle i.

1) Crossover Operation: For each dimension d of particle i, the following operation of crossover is conducted. A particle z ($z \in \{1, 2, \ldots, |\mathbb{X}|\}$) is randomly chosen. Then, the operation of crossover is performed on g and p_i to produce an offspring $o_i = <o_{i,1}, o_{i,2}, \ldots, o_{i,D}>$. Thus,

$$o_{i,d} = \begin{cases} r_d \cdot p_{i,d} + (1 - r_d) \cdot g_d & \widetilde{\Delta}(p_i) < \widetilde{\Delta}(p_z) \\ p_{z,d} & \text{otherwise} \end{cases}, \tag{50}$$

where r_d denotes a random number that is uniformly distributed in (0,1), i.e., $r_d \in (0,1)$.

Then, a crossover between g and p_i of particle i can enhance the quality of o_i. However, if particle i is an inferior individual in the population, o_i needs to adopt more entries from p_z of a randomly chosen particle z satisfying $\widetilde{\Delta}(p_z) \le \widetilde{\Delta}(p_i)$. Different from GA's random selection of two particles for the crossover operation, the proposed operation of crossover uses previous search experiences of particles to enhance the search ability.

2) Mutation Operation: The following operation of mutation is performed on o_i with a given probability of ζ. For each dimension d, if $r_d < \zeta$, $o_{i,d}$ is updated with a random number uniformly distributed in $(\check{b}_d, \hat{b}_d)$, i.e.,

$$o_{i,d} = \mathbf{rand}(\check{b}_d, \hat{b}_d), \text{ if } r_d < \zeta, \tag{51}$$

where $\hat{b}_d$ and $\check{b}_d$ are upper and lower bounds of the entry d of $\hbar_i$. It is worth noting that the operation is used to improve the diversity of superior exemplars.

3) Selection Operation: Following the crossover and mutation operations, this work adopts the selection operation to determine whether o_i or e_i is chosen in the next iteration. Specifically, if $\widetilde{\Delta}(o_i) < \widetilde{\Delta}(e_i)$, o_i is selected; otherwise, e_i keeps the same.

4) Update of Particle Position: $\hat{t}$ denotes the total number of iterations in PGL. ω_t denotes an inertia weight in iteration t ($1 \le t \le \hat{t}$). ω_t linearly reduces from its maximum value $\hat{\omega}$ to its minimum one $\check{\omega}$.

$$\omega_t = \hat{\omega} - \frac{t(\hat{\omega} - \check{\omega})}{\hat{t}}, \tag{52}$$

v_i denotes a velocity of each particle i. Then, in PGL, $v_{i,d}$ and $\hbar_{i,d}$ are changed as

$$v_{i,d} = \omega \cdot v_{i,d} + c \cdot r_d \cdot (e_{i,d} - \hbar_{i,d}) \tag{53}$$

$$\hbar_{i,d} = \hbar_{i,d} + v_{i,d}, \tag{54}$$

where c is a parameter of acceleration that reflects the effect of the difference between $x_{i,d}$ and $e_{i,d}$.

Each position of each particle consists of D elements. The first $3MK$ elements store λ_m^{k1}, λ_m^{k2} and λ_m^{k3}. The next MJ elements store x_{mj}. The next MK elements store f_m^k. Then, the next MJK elements store f_{mj}^k. The next M elements store P_m^t. The next MJ elements store μ_{mj}. Thus, $D = 3MK + MJ + MK + MJK + M + MJ = M(4K + 2J + JK + 1)$.

Algorithm 1. PGL

1: Randomly initialize the velocity and position of each particle
2: Calculate the fitness value ($\widetilde{\Delta}$) of each particle with (47)
3: Update a locally best position of particle i of each particle i, p_i, and a globally best position of all particles, g
4: Initialize ζ of GA, and parameters of PSO including c_1, c_2, c, $\hat{\omega}$, $\check{\omega}$, $\hat{t}$ and $|\mathbb{X}|$
5: $t \leftarrow 1$
6: **while** $t \leq \hat{t}$ **do**
7: Perform the crossover operation (50) on each dimension d of particle i to generate o_i
8: Perform the mutation operation (51) of GA on o_i with a given probability of ζ
9: Perform the selection operation of GA to determine whether o_i or e_i is chosen for each particle i in the next iteration
10: Update the velocity of each particle i with (53)
11: Update the position of each particle i in (54)
12: Calculate the fitness value ($\widetilde{\Delta}$) of each particle i by using (47)
13: Update p_i and g
14: $\omega_t \leftarrow \hat{\omega} - \frac{t(\hat{\omega}-\check{\omega})}{\hat{t}}$
15: $t \leftarrow t + 1$
16: **end while**
17: **return** g

Algorithm 1 presents the pseudo codes of PGL. Line 1 randomly initializes the velocity and position of each particle. Line 2 calculates the fitness value ($\widetilde{\Delta}$) of each particle with (47). Line 3 updates a locally best position of particle i of each particle i, p_i, and a globally best position of all particles as reached by the present iteration, g. Line 4 initializes ζ of GA, and parameters of PSO including c_1, c_2, c, $\hat{\omega}$, $\check{\omega}$, $\hat{t}$ and $|\mathbb{X}|$. Line 6 shows that the **while** loop stops if $t > \hat{t}$. Line 7 performs the crossover operation (50) on each dimension d of particle i to generate o_i. Line 8 performs the mutation operation (51) of GA on o_i with a given probability of ζ. Line 9 performs the selection operation of GA to determine whether o_i or e_i is chosen for each particle i in the next iteration. Lines 10–11 update the velocity and position of each particle i with (53) and (54), respectively. Line 12 calculates the fitness value ($\widetilde{\Delta}$) of each particle i by using (47). Line 13 updates p_i and g, respectively. Line 14 decreases ω_t in iteration t linearly from $\hat{\omega}$ to $\check{\omega}$. Line 17 returns g and converts it into decision variables including λ_m^{k1}, λ_m^{k2}, λ_m^{k3}, x_{mj}, f_m^k, f_{mj}^k, $\boldsymbol{P}_m^t$ and μ_{mj}.

In addition, this work further provides the analysis of complexity of Algorithm 1 as follows. In Algorithm 1, the execution overhead is largely caused by the **while** loop, which stops after $\hat{t}$ iterations. As shown in Lines 7–15, the complexity in each iteration of PGL is $\mathcal{O}(|\mathbb{X}|D)$. Besides, it is worth noting that $D = M(4K + 2J + JK + 1)$, and the complexity in each iteration is $\mathcal{O}(|\mathbb{X}|M(4K + 2J + JK + 1))$. Thus, the complexity of Algorithm 1 is $\mathcal{O}(\hat{t}|\mathbb{X}|MJK)$.

VI. PERFORMANCE EVALUATION

We give simulation experiments of PGL with real-world data in Google cluster.[1] According to the data, the size of each task for each MD is randomly produced. PGL is implemented in MATLAB 2018a and executed in a computer with a 16-GB DDR3 memory and an Intel Core i7 processor at 2.60 GHz with Intel HD Graphics 530.

A. PARAMETER SETTING

By following existing studies [32], this work considers a network topology consisting of one MBS and three SBSs, i.e., $J = 3$. This is for an illustration purpose, and it is worth noting that our proposed algorithm can scale up to deal with larger J. In addition, according to [35], the number of MDs, M, varies from 1 to 20. All SBSs and MDs are randomly distributed, and distances among SBSs and MDs vary from 50 m to 100 m, i.e., $d_{mj} \in [50, 100]$ m. For computation-intensive tasks, we assume that there are three separable tasks in each application, i.e., $K = 3$. The parameters of each MD are set in Table 1. According to [32], the parameters of each SBS and MBS are set in Tables 2 and 3. According to [35], [44], PGL's parameters are set in Table 4.

B. EXPERIMENTAL RESULTS

Figure 2 illustrates the total energy consumption in terms of different number of MDs (M), different number of SBSs (J) and different number of separable tasks (K) in each application. It is observed that given the same J and K, the total energy consumption increases almost linearly with M. This is because the number of total tasks in the system increases linearly when M increases, thereby leading to a nearly linear increase in the total energy consumption. In addition, given the same M and J, the total energy consumption increases with K. This is because as K increases, more tasks are offloaded to the edge side, i.e., SBSs and MBS due to limited computing resources in MDs. It further leads to more energy consumption in the processes of data transmission and computation offloading.

[1]https://github.com/google/cluster-data (accessed on Dec. 5, 2020)

TABLE 1. Parameter setting of each MD.

$\hat{E}_m$	T_m^*	$\hat{P}_m^t$	F_m	I_m^k	α_m^k	χ_m^k (ρ_m)	ς_m	P_m^0 (P_m^r)
[3, 6] J	[3,5] sec.	0.1 W	1×10^9 cycles/sec.	[0.5,1] MB	[50,100] cycles/bit	[50,100] ([16,20])	$[0.8\times10^{-25}, 1.3\times10^{-25}]$	[0.2,0.8] W

TABLE 2. Parameter setting of each SBS and MBS-Part 1.

$\hat{E}_j$	P_{Fj}^t	$\tilde{F}_j$	W_{Uj}	W_{Dj}	ς_j	$\hat{A}_1^j$	$\hat{G}_1^j$	$\hat{E}_0$
15 J	[0.1,0.4] W	5×10^9 cycles/sec.	[10,20] MHz	[10,20] MHz	$[0.8\times10^{-27}, 1.3\times10^{-27}]$	2×10^{10} cycles	2 GB	30 J

TABLE 3. Parameter setting of each SBS and MBS-Part 2.

f_0	r_0	e_0	$\hat{A}_2$	$\hat{G}_2$	v	h_1 (h_2)	G_0	P_δ^1 (P_δ^2)	β_1 (β_2)	β_3 (β_4)
5×10^9 cycles/sec.	1 Gbps	1 W/GHz	4×10^{10} cycles	4 GB	4	0.98 (0.98)	1.6×10^{-11}	0.001 W (0.001 W)	1 (0.2)	1 (0.2)

Figure 3 illustrates the effect of the maximum number of CPU cycles ($\hat{A}_1^j$) in SBSs on the total energy consumption with different number of MDs. Here, $J=3$, K=3, and the size of input data of each task ranges from 3 to 5 MB, i.e., $I_m^k \in [3,5]$. It is observed that given the same M, the total energy consumption reduces at first with the increase of $\hat{A}_1^j$, and then stabilizes. The reason is that to meet (35), more tasks are executed locally when $\hat{A}_1^j$ is insufficient at the beginning. It further leads to lager energy consumption. Then, as $\hat{A}_1^j$ increases, more tasks are offloaded to SBSs and MBS, and the total energy consumption is reduced accordingly. Finally, the total energy consumption is stable when $\hat{A}_1^j$ increases to a certain level. The reason is that no more tasks are offloaded to SBSs and MBS as the latency constraint (31) needs to be met. Meanwhile, it is worth noting that the total energy consumption tends to stabilize with lower $\hat{A}_1^j$ when M is smaller, e.g., $M=2$. This is because fewer MDs have fewer tasks that need to be processed and offloaded to SBSs and MBS without violating (35).

TABLE 4. Parameter setting of PGL.

c_1	c_2	c	$\hat{\omega}$	$\check{\omega}$	ζ	$\lvert\mathbb{X}\rvert$	$\hat{t}$	$\overset{\infty}{\mathcal{N}}$
0.5	0.5	1.49618	0.95	0.4	0.01	100	500	10^{10}

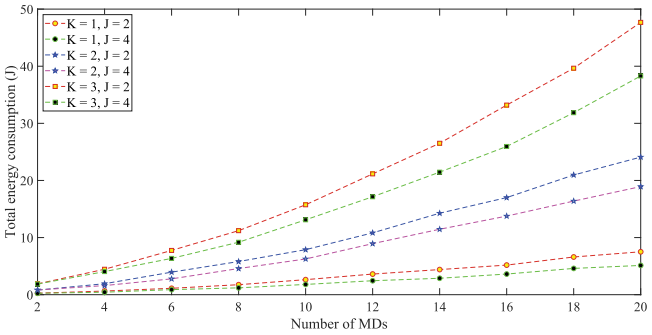

FIGURE 2. Total energy consumption with varying M, J, and K.

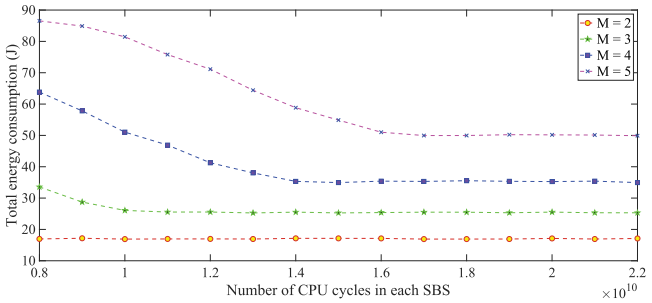

FIGURE 3. Total energy consumption with varying $\hat{A}_1^j$.

Figure 4 illustrates the total energy consumption in terms of different settings of T_m^* and $\hat{A}_1^j$. Here, $M=5$, $J=3$, $K=3$, and $I_m^k \in [2,4]$ MB. The total energy consumption reduces as T_m^* or $\hat{A}_1^j$ increases. This is because more tasks are offloaded to SBSs and MBS when $\hat{A}_1^j$ increases, thereby reducing the total energy consumption. In addition, when T_m^* is smaller, more tasks need to be executed in MDs to reduce the transmission time of tasks offloaded to SBSs and MBS. On the other hand, as T_m^* becomes larger, more tasks are offloaded to SBSs and MBS without violating (31), and therefore, the total energy consumption reduces accordingly.

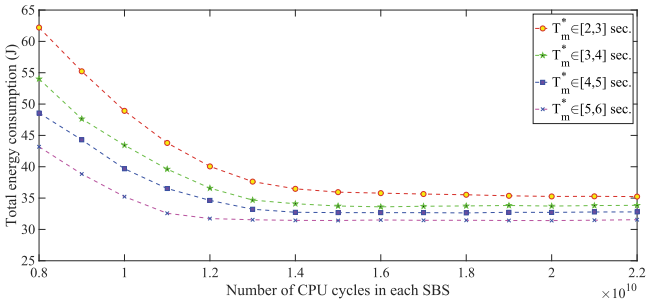

FIGURE 4. Total energy consumption with varying T_m^* and $\hat{A}_1^j$.

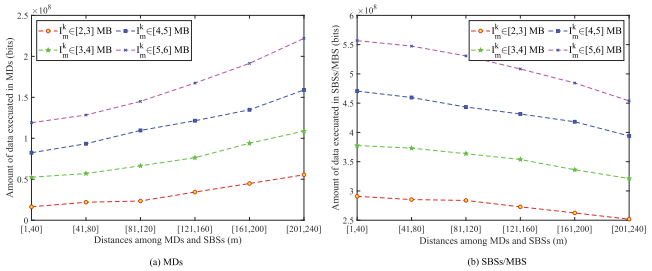

FIGURE 5. Data processed in MDs and SBSs/MBS with varying d_{mj} and I_m^k.

Figure 5 illustrates the amount of data executed in MDs and SBSs/MBS in terms of different settings of d_{mj} and I_m^k. Given the same I_m^k, as d_{mj} increases, more tasks are executed in MDs. The reason is that according to (2) and (3), the uplink and downlink rates between MD m and SBS j reduce as d_{mj} increases. Therefore, the total time needed by data uploading, downloading, and processing increases according to (8), and the energy consumption of uploading and downloading data between MDs and SBSs increases according to (17). Therefore, to decrease the total energy consumption and meet the constraint of (31), more tasks are executed in MDs as d_{mj} increases.

Figure 6 shows the amount of energy consumed by MDs and SBSs/MBS with respect to different settings of d_{mj} and I_m^k. It is shown that as d_{mj} increases, the amount of energy consumed by MDs and SBSs/MBS increases. The reason is that as d_{mj} increases, more tasks are executed in MDs, thereby increasing the energy consumed by MDs, as shown in Figure 6(a). In addition, as d_{mj} increases, the energy consumption of uploading and downloading data between MDs and SBSs increases according to (17), thereby increasing energy consumed by SBSs/MBS, as shown in Figure 6(b).

To demonstrate the effectiveness of our proposed PGL, this work compares it with its three widely used benchmark strategies including Full Offloading to MBS (FOM) [32], Full Offloading to SBSs (FOS) [32], and Partial Offloading without optimized user association (PO) [29].

- FOM [32]. All tasks of MDs are offloaded to servers in MBS for remote execution. In FOM, all MDs are associated with MBS.
- FOS [32]. All tasks of MDs are offloaded to servers in the SBSs for remote execution. In FOS, all MDs are associated with the SBSs. It is worth noting that all tasks of MDs are offloaded to SBSs on average.
- PO [29]. Given a feasible and fixed strategy of user association meeting (42), tasks of each MD are offloaded to their associated SBS, and some of them are further offloaded to MBS if resources of SBS are not sufficient to execute them.

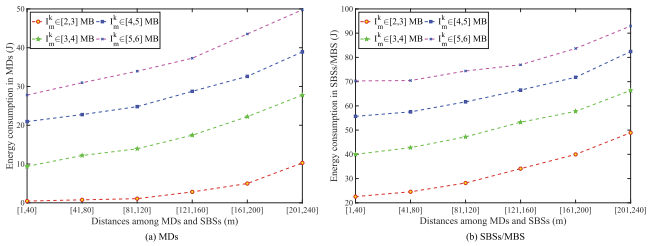

FIGURE 6. Energy consumed by MDs and SBSs/MBS with varying d_{mj} and I_m^k.

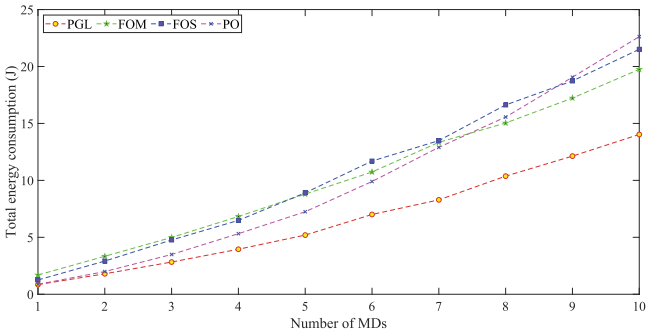

FIGURE 7. Total energy consumption of PGL, FOM, FOS, and PO with varying M.

Figure 7 illustrates the comparison of the total energy consumption of PGL, FOM, FOS, and PO with respect to different number of MDs, respectively. As shown in Figure 7, the total energy consumption of each algorithm increases with M. In addition, PGL outperforms FOM, FOS, and PO given the same number of MDs. The reason is that PGL jointly optimizes task offloading among MDs, SBSs, and MBS, users' association to SBSs, CPU speeds of MDs, SBSs, and MBS, MDs' transmission power and bandwidth allocation of available channels. On the other hand, PO is highly dependent on its given and fixed strategy of user association, and therefore, its total energy consumption is much larger than that of PGL given the same number of MDs. Specifically, compared with FOM, FOS, and PO, the total energy consumption of PGL is reduced by 34.68%, 37.53%, and 32.86%, respectively on average. In addition, in FOM and FOS, all tasks are offloaded to MBS and SBSs, respectively. Therefore, this greatly increases the energy consumption in the data transmission among MDs and SBSs/MBS, and the task execution in SBSs/MBS, thereby increasing the total energy consumption.

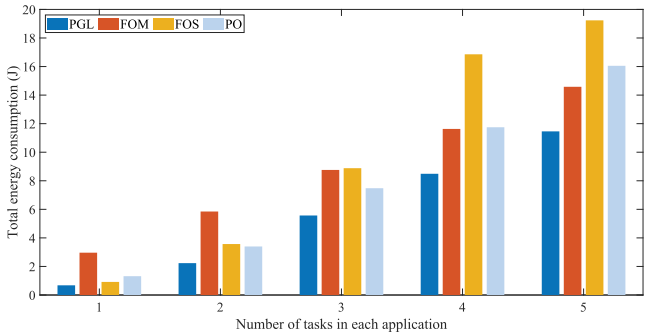

FIGURE 8. Total energy consumption of PGL, FOM, FOS, and PO with varying K.

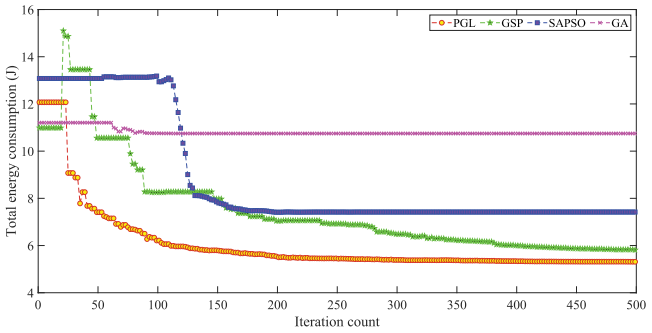

FIGURE 9. Evolutionary curves of total energy consumption in each iteration of PGL, GSP, SAPSO, and GA, respectively.

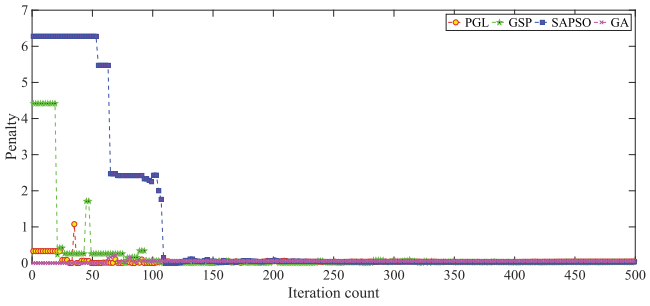

FIGURE 10. Evolutionary curves of total penalty in each iteration of PGL, GSP, SAPSO, and GA, respectively.

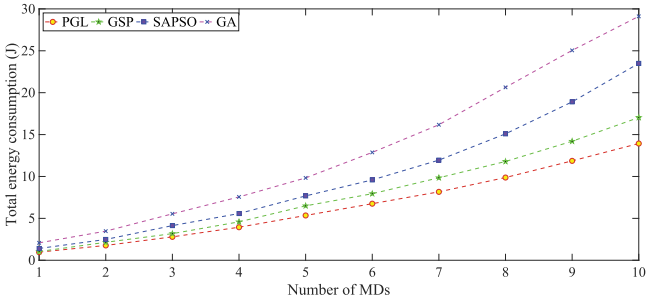

FIGURE 11. Total energy consumption of PGL, GSP, SAPSO, and GA in terms of different numbers of MDs.

Figure 8 illustrates the comparison of the total energy consumption of PGL in terms of different numbers of tasks in each application. Here, $M = 5$. It is observed that the total energy consumption of each strategy dramatically increases as K increases from 1 to 5. In addition, the total energy consumption of PGL is the least among four strategies. Specifically, compared with FOM, FOS, and PO, the total energy consumption of PGL is reduced by 35.07%, 42.52%, and 28.92%, respectively on average.

To demonstrate the effectiveness of PGL, this work compares it with its three recently-developed or deployed algorithms, i.e., Genetic simulated annealing (SA)-based PSO (GSP) [35], SA-based PSO (SAPSO) [45], and GA [46]. The reasons for choosing them for comparison are:

1) GSP [35] combines advantages of GA, SA, and PSO. It integrates GA's genetic operations and SA's Metropolis acceptance rule [47] into PSO. Thus, the comparison between PGL and GSP can demonstrate the search accuracy of PGL.
2) SAPSO [45] combines advantages of PSO and SA, and therefore it has PSO's quick convergence and SA's high global search ability. Thus, the comparison between PGL and SAPSO can prove the convergence speed of PGL.
3) GA [46] has genetic operations, strong ability to solve different types of complex problems, and owns high individual diversity. Thus, the comparison between PGL and GA can prove the convergence accuracy of PGL.

Figures 9 and 10 show the evolutionary curves of total energy consumption and total penalty in each iteration of PGL, GSP, SAPSO, and GA, respectively. Here, $M = 5$ and $K = 3$. It is observed that GA converges to its final solution at iteration 98, yet its final total energy consumption is the largest among four algorithms. SAPSO converges to its final solution at iteration 235. Its final total energy consumption is lower by 31.33% than that of GA, but still larger than that of PGL. GSP onverges to its final solution at iteration 478. Its final total energy consumption is lower by 21.59% than that of SAPSO, but still larger than that of PGL. PGL needs 353 iterations to converge to its final solution, and its final total energy consumption is the least among four algorithms. Figure 10 illustrates that the penalty of each algorithm is almost 0 at the end of iterations, and this means that constraints (31), (32), (33), (34), (35), (36), (37), (38), (39), (40), (41), (42), and (43) are well met. Therefore, Figures 9 and 10 prove that PGL outperforms other algorithms in terms of total energy consumption and is competitive in terms of convergence speed.

Figure 11 illustrates that the total energy consumption of PGL, GSP, SAPSO, and GA in terms of different number of MDs. It is observed that PGL's total energy consumption is the least given any number of MDs. Specifically, PGL's total energy consumption is lower than those of GSP, SAPSO, and GA by 16.42%, 34.80%, and 50.58% on average. The reason is that PGL adopts genetic operations including crossover, mutation, and selection to produce high-quality and diverse superior exemplars that guide the search process of PSO. Therefore, PGL keeps global search ability of GA and addresses PSO's early convergence as well. As shown in Figure 9, GSP's convergence speed is the slowest and it only finds a worse solution with higher total energy consumption than PGL's. The reason is that GSP is sensitive to the temperature cooling rate parameter of SA, and the Metropolis acceptance rule of SA enables it to accept a worse solution in its early iterations when the early temperature is very high. In addition, it is worth noting that PGL's total energy reductions compared with other three algorithms all increase with M. This is because the problem dimension increases with M. Then, the advantage of PGL's ability to solve high-dimension problems is more obvious, thereby decreasing the total energy consumption in comparison with other three algorithms.

VII. CONCLUSION

Currently, there are various resource-hungry and delay-sensitive applications running in mobile devices (MDs). MDs only have limited energy and resources, thereby failing to totally run all applications by themselves before their deadlines. Therefore, it is essential to partially offload some tasks of applications to nearby mobile edge computing (MEC) systems. Current studies fail to minimize energy consumption for complex applications with multiple tasks by smartly offloading tasks in distributed MEC systems including macro base stations (MBSs) and small base stations (SBSs). To solve it, this work first designs a computation offloading architecture for multi-task and delay-sensitive applications in hybrid networks including MDs, SBSs, and an MBS. The problem of total energy consumption minimization is formulated as a constrained mixed integer non-linear program. Then, it is solved by a proposed meta-heuristic optimization algorithm named Particle swarm optimization based on Genetic Learning (PGL). PGL jointly optimizes task offloading among MDs, SBSs, and MBS, user connection to SBSs, their CPU speeds, transmission power of MDs and bandwidth allocation of available channels. Real-world data-based simulations demonstrate that the proposed PGL realizes less energy consumption in less time than its three widely used optimization algorithms.

As future work, we plan to use a larger-scale hybrid network with numerous MDs and SBSs, and test the existing and proposed methods in real-life emerging 5G and 6G environments [48]–[50]. We also plan to consider moving MDs that dynamically change their locations, and further evaluate the performance of our proposed method.

REFERENCES

[1] A. Jin, W. Song, and W. Zhuang, "Auction-based resource allocation for sharing cloudlets in mobile cloud computing," *IEEE Trans. Emerg. Top. Comput.*, vol. 6, no. 1, pp. 45–57, First Quarter 2018.

[2] S. Ghanavati, J. Abawajy, and D. Izadi, "Automata-based dynamic fault tolerant task scheduling approach in fog computing," *IEEE Trans. Emerg. Top. Comput.*, to be published, doi: 10.1109/TETC.2020.3033672.

[3] H. Chang, Y. Chen, B. Zhang, and D. Doermann, "Multi-UAV mobile edge computing and path planning platform based on reinforcement learning," *IEEE Trans. Emerg. Top. Comput. Intell.*, to be published, doi: 10.1109/TETCI.2021.3083410.

[4] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Top. Comput.*, vol. 9, no. 3, pp. 1529–1541, Third Quarter 2021.

[5] M. Asim, Y. Wang, K. Wang, and P.-Q. Huang, "A review on computational intelligence techniques in cloud and edge computing," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 4, no. 6, pp. 742–763, Dec. 2020.

[6] Q. Han, B. Yang, G. Miao, C. Chen, X. Wang, and X. Guan, "Backhaul-aware user association and resource allocation for energy-constrained HetNets," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 580–593, Jan. 2017.

[7] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An edge-computing based architecture for mobile augmented reality," *IEEE Netw.*, vol. 33, no. 4, pp. 162–169, Jul./Aug. 2019.

[8] Y. Xie *et al.*,"A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment," *Future Gener. Comput. Syst.*, vol. 97, pp. 361–378, Aug. 2019.

[9] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[10] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 726–738, Sep./Oct. 2019.

[11] M. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[12] G. Xu *et al.*, "Particle swarm optimization based on dimensional learning strategy," *Swarm Evol. Comput.*, vol. 45, pp. 33–51, Mar. 2019.

[13] K. Guo, "Research on location selection model of distribution network with constrained line constraints based on genetic algorithm," *Neural Comput. Appl.*, vol. 32, no. 6, pp. 1679–1689, May 2019.

[14] Z. Ning *et al.*,"Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1060–1072, Dec. 2019.

[15] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet of Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[16] Z. Ning *et al.*, "Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2020.3025116.

[17] Y. Wu, B. Shi, L. P. Qian, F. Hou, J. Cai, and X. S. Shen, "Energy-efficient multi-task multi-access computation offloading via NOMA transmission for IoTs," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4811–4822, Jul. 2020.

[18] K. Xie *et al.*, "Distributed multi-dimensional pricing for efficient application offloading in mobile cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 6, pp. 925–940, Nov./Dec. 2019.

[19] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 726–738, Sep./Oct. 2019.

[20] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[21] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.

[22] T. D. T. Nguyen, V. Nguyen, V.-N. Pham, L. N. T. Huynh, M. D. Hossain, and E.-N. Huh, "Modeling data redundancy and cost-aware task allocation in MEC-enabled Internet-of-Vehicles applications," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1687–1701, Feb. 2021.

[23] Z. Ali, S. Khaf, Z. H. Abbas, G. Abbas, F. Muhammad, and S. Kim, "A deep learning approach for mobility-aware and energy-efficient resource allocation in MEC," *IEEE Access*, vol. 8, pp. 179530–179546, Oct. 2020.

[24] W. Na, S. Jang, Y. Lee, L. Park, N. Dao, and S. Cho, "Frequency resource allocation and interference management in mobile edge computing for an Internet of Things system," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4910–4920, Jun. 2019.

[25] P. Wang, C. Yao, Z. Zheng, G. Sun, and L. Song, "Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2872–2884, Apr. 2019.

[26] Y. Liu, Y. Li, Y. Niu, and D. Jin, "Joint optimization of path planning and resource allocation in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 9, pp. 2129–2144, Sep. 2020.

[27] C. Zhan, H. Hu, X. Sui, Z. Liu, and D. Niyato, "Completion time and energy optimization in the UAV-enabled mobile-edge computing system," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7808–7822, Aug. 2020.

[28] X. Huang, D. Ye, R. Yu, and L. Shu, "Securing parked vehicle assisted fog computing with blockchain and optimal smart contract design," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 2, pp. 426–441, Mar. 2020.

[29] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[30] O. Munoz, A. Pascual-Iserte , and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[31] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398–401, Jun. 2017.
[32] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12313–12325, Dec. 2018.
[33] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.
[34] G. S. Park and H. Song, "Cooperative base station caching and X2 link traffic offloading system for video streaming over SDN-enabled 5G networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 2005–2019, Sep. 2019.
[35] J. Bi, H. Yuan, S. Duanmu, M. C. 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.
[36] A. Rahmati, V. Shah-Mansouri , and M. Safari, "Price-based resource allocation for self-backhauled small cell networks," *Comput. Commun.*, vol. 97, pp. 72–80, Jan. 2017.
[37] W. Wen, Y. Cui, T. Q. S. Quek, F. Zheng, and S. Jin, "Joint optimal software caching, computation offloading and communications resource allocation for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7879–7894, Jul. 2020.
[38] 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.
[39] A. Jayswal, "An exact L1 penalty function method for multi-dimensional first-order PDE constrained control optimization problem," *Eur. J. Control*, vol. 52, pp. 34–41, Mar. 2020.
[40] G. D'Angelo, M. Tipaldi, F. Palmieri, and L. Glielmo, "A data-driven approximate dynamic programming approach based on association rule learning: Spacecraft autonomy as a case study," *Inf. Sci.*, vol. 504, pp. 501–519, Dec. 2019.
[41] Y. Wei, M. M. Zhao, M. Hong, M. J. Zhao, and M. Lei, "Learned conjugate gradient descent network for massive MIMO detection," *IEEE Trans. Signal Process.*, vol. 68, pp. 6336–6349, 2020.
[42] Q. Zhang, H. Li, and H. Li, "An improved least squares SVM with adaptive PSO for the prediction of coal spontaneous combustion," *Math. Biosci. Eng.*, vol. 16, no. 4, pp. 3169–3182, Apr. 2019.
[43] 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.
[44] 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.
[45] F. Javidrad and M. Nazari, "A new hybrid particle swarm and simulated annealing stochastic optimization method," *Appl. Soft Comput.*, vol. 60, pp. 634–654, Nov. 2017.
[46] P. M. Rekha and M. Dakshayini, "Efficient task allocation approach using genetic algorithm for cloud environment," *Cluster Comput.*, vol. 22, no. 4, pp. 1241–1251, Jan. 2019.
[47] C. W. Tsai, C. H. Hsia, S. J. Yang, S. J. Liu, and Z. Y. Fang, "Optimizing hyperparameters of deep learning in predicting bus passengers based on simulated annealing," *Appl. Soft Comput.*, vol. 88, pp. 1–9, Mar. 2020.
[48] X. Lin, J. Wu, S. Mumtaz, S. Garg, J. Li, and M. Guizani, "Blockchain-based on-demand computing resource trading in IoV-assisted smart city," *IEEE Trans. Emerg. Top. Comput.*, vol. 9, no. 3, pp. 1373–1385, Third Quarter 2021.
[49] L. Silva *et al.*, "Computing paradigms in emerging vehicular environments: A review," in *Proc. IEEE/CAA J. Automatica Sinica*, vol. 8, no. 3, Mar. 2021, pp. 491–511.
[50] G. Fortino, C. Savaglio, G. Spezzano, and M. Zhou, "Internet of things as system of systems: A review of methodologies, frameworks, platforms and tools," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 223–236, Jan. 2021.

JING BI (Senior Member, IEEE) received the BS and PhD degrees in computer science from Northeastern University, Shenyang, China, in 2003 and 2011, respectively. She is currently an associate professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. She was a visiting research scholar with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, New Jersey. She has more than 80 publications including journal and conference papers. Her research interests include Distributed Computing, Cloud Computing, Large-Scale Data Analysis, Machine Learning and Performance Optimization. She was the recipient of the IBM Fellowship Award, the Best Poster Prize in the 3rd IFAC Workshop on Cyber-Physical & Human Systems (CPHS 2020), the recipient of the Best Paper Award-Finalist in the 16th IEEE International Conference on Networking, Sensing and Control (ICNSC), and the recipient of the Best Paper Award in the 17th ICNSC. She is now an associate editor of the *IEEE Access*.

HAITAO YUAN (Senior Member, IEEE) received the PhD degree in modeling simulation theory and technology from Beihang University, Beijing, China, in 2016, and the PhD degree in computer engineering from the New Jersey Institute of Technology (NJIT), Newark, New Jersey, in 2020. He is currently an associate professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. 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, the 2011 Google Excellence Scholarship, the Best Poster Prize in the 3rd IFAC Workshop on Cyber-Physical & Human Systems (CPHS 2020), the recipient of the Best Paper Award-Finalist in the 16th IEEE International Conference on Networking, Sensing and Control (ICNSC), and the recipient of the Best Paper Award in the 17th ICNSC.

KAIYI ZHANG received the BS degree in software engineering form the Beijing University of Technology, China, in 2018. He is currently working toward the master's degree in the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. His research interests include Cloud Computing, Data Center, Task Scheduling, Computation Offloading, Intelligent Optimization Algorithms, Machine Learning, and Reinforcement Learning.

MENGCHU ZHOU (Fellow, IEEE) received the BS degree in control engineering from Nanjing University of Science and Technology, Nanjing, China in 1983, MS degree in automatic control from Beijing Institute of Technology, Beijing, China in 1986, and Phd degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY in 1990. He then joined New Jersey Institute of Technology (NJIT), Newark, NJ in 1990, and is a Distinguished professor of Electrical and Computer Engineering. His interests are in Petri nets, intelligent automation, Internet of Things, and Big Data. He has more than 900 publications including 12 books, more than 600 journal papers (more than 500 in IEEE transactions), 29 patents and 29 book-chapters. His recently co-authored/edited books include *Supervisory Control and Scheduling of Resource Allocation Systems: Reachability Graph Perspective*, IEEE Press/Wiley, Hoboken, Hoboken, New Jersey, 2020 (with B. Huang). He is the founding Editor of IEEE Press Book Series on Systems Science and Engineering and Editor-in-Chief of IEEE/CAA Journal of Automatica Sinica. 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).