Cost-Minimized Computation Offloading and User Association in Hybrid Cloud and Edge Computing

Jing Bi, *Senior Member, IEEE*, Ziqi Wang, *Student Member, IEEE*, Haitao Yuan, *Senior Member, IEEE*, Jia Zhang, *Senior Member, IEEE*, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Smart mobile devices (SMDs) are integral for running advanced applications that demand significant computing resources and quick response time, e.g., immersive gaming and advanced image editing. However, SMDs often face constraints in computational capacity and battery duration, restricting their ability to process these tasks instantaneously. Cloud computing can circumvent these limitations by computation offloading, but cloud data centers (CDCs) are often deployed at long distances from users, which results in longer computational latency. To address the latency issue, the incorporation of small base stations (SBSs) in the vicinity of the user provides services with high bandwidth and low latency. The primary challenge lies in balancing the economics of the system consisting of different SMDs, SBSs, and a CDC, i.e., minimizing cost while still meeting the latency requirements of applications. In this work, a cost-minimized computation offloading framework is formulated and solved by a two-stage optimization algorithm named Lévy flight and simulated annealing-based grey wolf optimizer (LSAG). The optimal edge selection strategy is defined in the first stage for dealing with the case of several available SBSs. The second stage coordinates task scheduling and optimizes the allocation of resources among SMDs, SBSs, and CDC. LSAG integrates the extended search property of Lévy flight and the individual selection strategy of simulated annealing in the grey wolf optimizer, which reduces the risk of falling into local optima and finds the global optimum. Experimental results of executing real-life tasks show that LSAG outperforms its state-of-the-art peers in terms of cost and speed of convergence.**

***Index Terms*—Cloud computing, computation offloading, edge computing, grey wolf optimizer (GWO), swarm intelligence algorithms.**

Manuscript received 31 July 2023; revised 12 December 2023; accepted 12 January 2024. Date of publication 16 January 2024; date of current version 25 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62173013 and Grant 62073005; in part by the Beijing Natural Science Foundation under Grant 4232049; and in part by the Fundamental Research Funds for the Central Universities under Grant YWF-23-03-QB-015. This article was presented in part at the 2023 IEEE Conference on Systems, Man, and Cybernetics, Honolulu, HI, USA. *(Corresponding author: Haitao Yuan.)*

Jing Bi and Ziqi Wang are with the School of Software Engineering, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn; ziqi_wang@emails.bjut.edu.cn).

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

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.2024.3354348

I. Introduction

OVER recent years, the widespread use of smart mobile devices (SMDs) and advancements in wireless communication has brought a large number of applications that significantly enrich our daily routines, e.g., mobile gaming and online conferencing [1]. However, these applications often demand extensive computing resources, e.g., CPU time and memory, and consume significant amount of battery power. Given the limited computational capabilities and battery endurance of SMDs, running such demanding applications on the devices themselves poses a substantial challenge. Furthermore, the high energy demand also leads to rapid battery drain, ultimately reducing the lifespan of SMDs. Cloud computing offers an expansive supply of computational resources, enabling SMDs to offload intensive tasks for remote processing that alleviates these issues. It enables SMDs to offload their computation-intensive tasks to cloud data centers (CDCs) through wireless connections. However, the remote placement of CDCs introduces latency issues because the data must travel considerable distances to and from servers in CDCs, which is particularly problematic for applications that require immediate response.

In response to these concerns, edge computing emerges as a solution to counteract the latency concern by leveraging small base stations (SBSs) positioned closer to the users [2]. SBSs have more computational resources than SMDs and provide local processing for offloaded tasks, thus delivering services with high bandwidth and minimal latency. However, it is worth noting that the computational resources at SBSs are not as abundant as those in CDCs. Thus, low-latency fiber links connect SBSs to CDCs. When SBSs encounter overloads or tasks beyond their processing capacities, they can still delegate those tasks to CDCs. It also needs to ensure that the processing results can be returned promptly to SMDs. This collaboration establishes a cloud-assisted mobile-edge computing (CMEC) architecture especially suited for mobile applications that require large amounts of computational resources and have stringent latency requirements.

However, there are still three issues that demand our resolution. The first one is the latency issue in communications. Specifically, the extra processes of task offloaded from SMDs to SBSs, and from SBSs to the CDCs unavoidably cause additional communication latency [3]. However, some applications are delay-sensitive and require results to be returned within a specified time limit defined by the users [4]. In that case, the total system latency must be within the

acceptable range for the applications. The second one is the resource allocation issue. With a large number of SMDs offloading tasks to different SBSs that may further offload tasks to the CDCs, it becomes an issue of how to efficiently allocate resources in each SMD and SBS [5]. Finally, the total cost of a CMEC system comprises costs involving SMDs, SBSs, and the CDC and it is directly related to the system's energy consumption. In addition, lower system cost indicates a more sustainable and environmentally friendly system [6]. In dealing with the above problems, some of the studies [7], [8], [9] consider minimizing the system delay, and others [10], [11], [12] consider minimizing the system energy consumption. However, it is difficult for existing computation offloading strategies to balance resource allocation and energy consumption while satisfying SMDs' latency requirements in the CMEC architecture.

This work designs a partial computation offloading technique, which aims to realize cost minimization in such a complex and heterogeneous environment while meeting the delay requirements of mobile applications. As the first step, without losing generality, we propose a fundamental unit architecture comprising multiple SMDs, SBSs, and a CDC. This work adopts triple queueing models to analyze its overall cost and performance. To be specific, $M/M/1$, $M/M/c$, and $M/M/\infty$ are adopted to model and monitor SMDs, SBSs and the CDC, respectively. Moreover, this work adopts an $M/M/1$ queueing model in the transmission channel. Based on this architecture, a constrained optimization problem of cost minimization for the CMEC system is formulated. Moreover, a novel two-stage algorithm, named Lévy flight and simulated annealing-based grey wolf optimizer (LSAG) is designed for solving the above problems. LSAG incorporates Lévy flight and a Metropolis acceptance criterion of simulated annealing (SA) into a grey wolf optimizer (GWO). It is worth noting that LSAG takes the resource allocation parameters to be optimized as decision variables, and the computational capacity and delay requirements of the CMEC system as constraints. LSAG aims to optimize the system cost, which is a single-objective optimization problem. Experiments using real-world application tasks from Google's CDCs reveal that LSAG realizes cost-efficient computation offloading in the CMEC architecture.

This work intends to make new contributions to CMEC given as follows.

1) This work constructs a fundamental unit architecture for studying the CMEC system, including SMDs, SBSs, and CDC, which are characterized by heterogeneous triple queueing models to analyze its overall cost and performance.
2) A multiconstraint cost optimization problem is constructed according to the CMEC system, supported by a novel two-stage optimization algorithm called LSAG.
3) Experiments and the comparison with several typical algorithms with realistic trace data demonstrate that the proposed LSAG significantly achieves lower cost and faster convergence speed compared with its state-of-the-art peers.

In addition, main differences between the current work and our previous one [13] are listed as follows.

1) Different from [13], this work further considers different queueing models of SMDs, SBSs, and the CDCs to analyze the overall cost and performance of the CMEC system.
2) Different from [13], this work explores queueing models of transmission channels between SMDs and SBSs, which makes the CMEC system closer to the actual system scenario.
3) Different from [13], this work further investigates the impact of user association in the first stage of LSAG on the cost of the CMEC system and gives the experimental results.

The remainder of this work is given as follows. Section II gives the related work. Section III formulates a cost minimization problem. Section IV gives the details of the implementation of LSAG. Section V discusses the simulation results. Section VI concludes this work.

II. Related Work

This section discusses the related work from two aspects, i.e., resource allocation and energy efficiency in mobile-edge computing (MEC).

A. Resource Allocation in MEC

In a system with multiple SMDs, SBSs, and the CDC, the computational offloading spreads the execution of tasks across them. As a result, a rational allocation of computational resources between edge and the CDC is required for users because of their limited computational resources. A proper resource allocation strategy can satisfy the requirements of each SMD while conserving server resources. Moreover, since different services in SMDs require different requirements of Quality of Service (QoS), each service may utilize different resource allocation strategies to satisfy its own QoS need. Several studies have focused on resource allocation in CDC or edge systems recently.

Xiao et al. [14] pointed out that MEC has difficulty in properly scheduling policies that ensure fairness of loads among MEC servers while maintaining a high level of resource utilization. To address it, they design an MEC service migration method based on coalition game and location-aware mechanisms for SMDs. It divides MEC servers into coalitions according to their Euclidean distances. Furthermore, it discovers hotspots in each coalition region and migrates tasks to proper edge servers to realize high resource utilization. However, this work ignores the energy consumption during service migration and it may cause significant energy consumption in this system. Ma et al. [15] showed that workloads of SBSs are usually unbalanced, which causes high response time and resource costs. Therefore, a dynamic task scheduling strategy is proposed to optimize the average edge response time while satisfying the limits of resources. However, this strategy does not consider the capabilities of

SMDs. Wang et al. [16] considered the imperfect channel-state information in MEC and proposed a two-stage algorithm to decide the resource allocation strategy. Specifically, the first stage reduces the original problem into a simplified one and provides offloading priorities for all devices. The second stage obtains an offloading decision. However, although the algorithm has low complexity, some practical problems do not have specific structures for the first stage. Li et al. [17] considered numerous idle resources in vehicles in an urban area, and therefore, vehicles are used as edge servers to construct an MEC system. Moreover, they formulated this model as a multistage Stackelberg game for dealing with the resource allocation problem. Nevertheless, the workloads of different vehicles are unbalanced and cause unbalanced issues. Chen et al. [18] divided an MEC problem into two separate optimization problems and the resource allocation strategy considers both problems. The first one aims to optimize the local execution cost and the second one aims to optimize the offloading execution cost. They also considered the instability of the connections between SMDs and the cloud because of their mobility. In that case, the proposed offloading strategy is embedded with failure recovery mechanisms. However, the method has high time complexity, which leads to difficulties in solving practical problems. Since different types of tasks have different resource requirements, Yun et al. [19] considered that different services in SMDs require different QoS levels and some services may be urgent. In that case, they put different types of tasks in SMDs into different service queues according to their types, and each queue is managed dynamically to effectively reduce the queuing delay of urgent tasks. Moreover, the total workload of all queues is monitored for dealing with insufficient resources in SBSs. However, some low-priority queues may be blocked for a long time because of other high-priority queues taking up resources and this method does not consider the usage of cloud computing.

Different from the aforementioned studies, this work constructs a CMEC system to determine the allocation of bandwidth resources in transmission networks and hardware resources (computing and storage) in SBSs and the cloud. We consider the generality and low complexity of the proposed method, which allows it to better solve more practical problems.

B. Energy Efficiency in MEC

Energy consumption and cost are important in a multilayer heterogeneous MEC system because they affect the sustainability of the system. Less energy consumption can slow down the aging of the hardware and increase the economic efficiency of the system. There are many studies on energy consumption in an MEC system.

Merluzzi et al. [20] proposed an energy-efficient strategy for reducing the total energy consumption in an MEC-aided 5G network system. They propose a sleep operation for SBSs that shift edge servers from always-on to always-available manner for reducing the energy consumption. However, this strategy does not consider the resource allocation in this system. Shi et al. [21] proposed a nonlinear energy model for SMDs that considers both resource requirements and energy consumption. Moreover, they turn this problem into a nonlinear program and propose a Dinkelbach-based iterative algorithm to solve it. Nevertheless, this method does not consider the battery levels of edge users. In addition, some studies consider multiobjective optimization and one of the objectives is to minimize energy consumption, Song et al. [22] investigated the problem of route planning and resource allocation in an unmanned aerial vehicle (UAV)-aided MEC architecture. UAVs are considered edge base stations in this architecture, and its trajectory optimization aims to make the UAV receive more users' offloaded tasks and minimize the system energy consumption while reducing the overall computational delay of the system. They propose an evolutionary-based reinforcement learning approach that aims to find a balance between the three objectives. Huang et al. [23] and Guo et al. [24] solved a multiobjective disassembly and resource-constrained optimization problem. The objectives of the problem include minimizing energy consumption, minimizing disassembly time, and maximizing profitability. They proposed a lexicographic multiobjective scatter search (LMSS) to solve the problem. Specifically, each individual means an objective solution, and the population is used to find the global optimum of the problem. In addition, LMSS employs linear weight scatter search to improve the search efficiency and accuracy of the algorithm. Finally, Guim et al. [25] considered the autonomous lifecycle management for an MEC system. They proposed a strategy that allows the efficient usage of resources while guaranteeing QoS of various services. Moreover, the energy consumption of the entire system is monitored in a real-time manner to achieve energy minimization. However, it has high complexity, making it challenging to solve real-world problems.

In summary, different from these aforementioned studies, we design an enhanced computation offloading approach for minimizing the cost of the CMEC system. We consider the energy consumption of SMDs, SBSs, and the CDC. Moreover, we focus on delay-sensitive applications in such systems and consider resource allocation among them. Finally, a novel two-stage optimization algorithm is proposed to minimize the system cost while satisfying time constraints.

III. Problem Formulation

This section formulates a cost minimization problem based on the CMEC systems. Fig. 1 shows the architecture of the partial computation offloading and main notations are summarized in Table I. It comprises N SMDs, J SBSs, and a CDC. Each SMD is associated with one SBS, and each SBS is linked to the CDC by fiber optics. This work considers tasks that can be partitioned into several small tasks and each of them can be processed independently and in parallel. For instance, antivirus scanning software can be decomposed into several discrete tasks, and each of them can be processed in an SMD, SBS, or the CDC.

To analyze and monitor a CMEC system in partial computation offloading scenarios, queuing models are adopted to evaluate the performance of each component [26]. Each

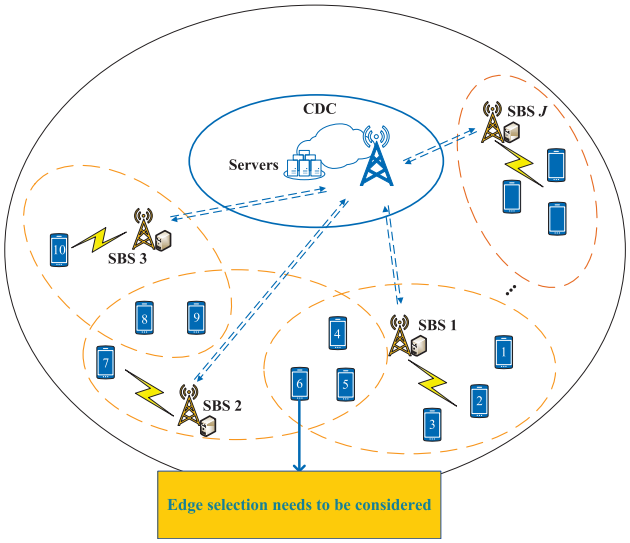


Fig. 1. Partial computation offloading architecture of CMEC systems.

SMD supports a specific type of applications and yields tasks continuously. Thus, an $M/M/1$ queue is adopted to simulate an SMD. Taking into account their different roles in the partial computation offloading scenarios, each SBS is analyzed as an $M/M/c$ queue model, and the CDC is analyzed as an $M/M/\infty$ queue model. In addition, this work also considers the queue model of shared channels between SMDs and SBSs. Following [27], an $M/M/1$ queue is adopted in the uplink and downlink of shared channels. Fig. 2 illustrates queueing models of shared channels in this CMEC system.

For the given SMD i ($1 \le i \le N$), if there is a connection to SBS j ($1 \le j \le J$), $\mu_{ij} = 1$; otherwise, $\mu_{ij} = 0$. P_i^k, P_j^k, and P^k represent the fractions of task k that are processed in SMD i, SBS j, and the CDC, respectively. They satisfy that

$$P_i^k + P_j^k + P^k = 1. \tag{1}$$

In the next sections, we first give the modeling of SMDs, SBSs, and the CDC, and formulate a latency model and a cost one for the unit architecture. Finally, a multiconstraint cost optimization problem is formulated for the CMEC system.

TABLE I
MAIN PARAMETERS

Notation	Definition
N	Number of SMDs
J	Number of SBSs
T_i^k	Execution time of task k in SMD i
α_i^k	Quantity of CPU cycles for processing each bit of tasks k in SMD i
χ_i^k	Quantity of memory for processing each bit of task k in SMD i
P_i^k	Power consumption involved in performing a specific part of task k in SMD i
$\hat{P}_i^t$	Maximum transmission power of SMD i
S_i	Constant determined by the chip architecture in SMD i
E_i	Total energy used to perform all K tasks in SMD i
P_i^t	Power consumption during data transmission between SMD i and its associated SBS
d_{ij}	Distance from SMD i to SBS j
$\hat{R}_{ij}$ and $\check{R}_{ij}$	Transmission rate for both uplink and downlink between SMD i and SBS j
$\hat{B}_j$ and $\check{B}_j$	Uplink and downlink bandwidth of SBS j
P_j^t	Transmission power of SBS j
λ_{ij}	Channel bandwidth employed by SMD i in SBS j's channels
f_C	Computation speed of the CDC
$\widetilde{T}_{ij}^k$	Overall time required to perform data uploading, downloading and processing for task k in SMD i when using SBS j
$\overline{T}_{ij}^k$	Overall time required to perform data uploading, downloading and processing for task k in SMD i when using the CDC
T_{ij}^k	Time required to perform task k of SMD i in SBS j and the CDC
r_t	Transmission rate between SBSs and the CDC
T_i	Overall time required to perform all K tasks when executing them in SMDs, SBSs, and the CDC
$\hat{T}_i$	Upper limit of T_i
E_{ij}	Energy consumption involved in performing an offloaded task of SMD i when executing it in SBS j
E_{i0}	Energy consumption involved in performing an offloaded task of SMD i when executing it in the CDC
S_j	Constant determined by the chip architecture of SBS j
P^S	Power consumption for transmitting data from SBSs to the CDC
P^C	Power consumption for transmitting data from the CDC to SBSs
e_c	Energy consumption of each CPU cycle in the CDC
$\hat{E}_j$	Maximum available energy in SBS j
$\hat{E}_c$	Maximum available energy in the CDC
$\mathcal{F}_1$	Total cost of all SMDs
$\mathcal{F}_2$	Total cost of all SBSs
$\mathcal{F}_3$	Total cost of the CDC

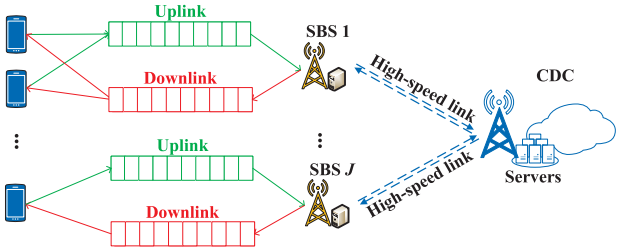


Fig. 2. Queueing models of shared channels in the CMEC system.

A. Modeling of SMDs

This section constructs the working model of SMDs and constraints based on their runtime performance. In this work, we assume that tasks from each SMD i arrive in a Poisson manner [28]. T_i^k is the time to run task k in SMD i, i.e.,

$$T_i^k = \frac{I_i^k P_i^k \alpha_i^k}{f_i^k} \tag{2}$$

where I_i^k represents the volume of input data for task k acquired by SMD i, α_i^k denotes the quantity of CPU cycles for processing each bit of task k in SMD i, and f_i^k denotes the computational speed for executing task k in SMD i.

For each SMD i, the CPU speed utilized for performing all tasks must not surpass its maximum allowable CPU speed (F_i), i.e.,

$$\sum_{k=1}^{K} f_i^k \le F_i. \tag{3}$$

P_i^k denotes the power consumption involved in performing a specific part of task k in SMD i, i.e.,

$$P_i^k = S_i (f_i^k)^3 \tag{4}$$

where S_i is a fixed value established by the architecture of SMD i's chip.

E_i^1 is the total amount of energy used to complete all K tasks in SMD i, and it is obtained as

$$E_i^1 = \sum_{k=1}^{K} S_i I_i^k P_i^k \alpha_i^k (f_i^k)^2. \tag{5}$$

SMDs may offload tasks over the wireless channels to SBSs for processing. P_i^t represents the power consumption during data transmission between SMD i and its associated SBS. In addition, it must not surpass its peak transmission power ($\hat{P}_i^t$), i.e.,

$$0 \le P_i^t \le \hat{P}_i^t. \tag{6}$$

Accordingly, the energy consumption for transmitting data from SMD i to its associated SBS is denoted as E_i^2, i.e.,

$$E_i^2 = P_i^t T_i^t \tag{7}$$

where T_i^t denotes the transmission time between SMD i to its associated SBS.

The amount of data returned by its SBS is considerably smaller than the size of its original transmitted data [29]. Consequently, the energy consumption required for SMDs to receive these results can be ignored. Accordingly, the energy consumption of SMD i is denoted as E_i and it includes two parts: local execution energy (E_i^1) and data transmission energy (E_i^2), i.e.,

$$E_i = E_i^1 + E_i^2. \tag{8}$$

B. Modeling of SBSs and the CDC

This section constructs the working model of SBSs and the CDC as well as constraints based on their runtime performance. The distance between SMD i and SBS j is represented as d_{ij}. Based on [30], the path loss between them is $(d_{ij})^{-v}$, where v is a parameter of the path loss. λ_{ij} denotes the channel bandwidth employed by SMD i in channels of SBS j, and the bandwidth of SBS j assigned to all SMDs is one, i.e.,

$$\sum_{i=1}^{N} \mu_{ij}\lambda_{ij} = 1. \tag{9}$$

$\hat{B}_j$ and $\check{B}_j$ refer to uplink and downlink bandwidth of SBS j, respectively. $\hat{R}_{ij}$ and $\check{R}_{ij}$ denote the upstream and downstream transmission rates from SMD i to SBS j. According to Shannon's theorem [31], we have

$$\hat{R}_{ij} = \lambda_{ij}\hat{B}_j \log_2\left(1+\frac{P_i^t (d_{ij})^{-v}|f_1|^2}{\omega_0}\right) \tag{10}$$

$$\check{R}_{ij} = \lambda_{ij}\check{B}_j \log_2\left(1+\frac{P_j^t (d_{ij})^{-v}|f_2|^2}{\omega_0}\right) \tag{11}$$

where P_j^t means the power of transmitting data from SBS j to each SMD, and f_1 and f_2 represent the wireless channel fading parameters, which reflect the variations in signal strength that occur as radio waves propagate through the environment. ω_0 is the noise power of white Gaussian noise.

f_{ij}^k represents the running speed at which task k from SMD i is executed by SBS j and the computational speed of running all tasks in SBS j must not surpass its upper bound limit ($\hat{F}_j$), i.e.,

$$\sum_{i=1}^{N}\sum_{k=1}^{K} \mu_{ij} f_{ij}^k \le \hat{F}_j. \tag{12}$$

The number of CPU cycles used to execute tasks in SBS j cannot exceed its predetermined limit ($\hat{C}_S^j$), i.e.,

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\left(\mu_{ij} I_i^k P_j^k \alpha_i^k\right) \le \hat{C}_S^j. \tag{13}$$

Moreover, memory required by tasks executed in SBS j must not surpass its upper bound limit ($\hat{M}_S^j$), i.e.,

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\left(\mu_{ij} I_i^k P_j^k \chi_i^k\right) \le \hat{M}_S^j \tag{14}$$

where χ_i^k denotes the memory requirement for carrying out a single bit of task k in SMD i.

Moreover, the number of CPU cycles and memories consumed by tasks executing in the CDC must not surpass their predetermined limits, i.e.,

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\left(I_i^k P^k \alpha_i^k\right) \le \hat{C}_{CDC} \tag{15}$$

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\left(I_i^k P^k \chi_i^k\right) \le \hat{M}_{CDC} \tag{16}$$

where $\hat{C}_{CDC}$ and $\hat{M}_{CDC}$ represent the upper bound limits of CPU cycles and memories in CDC, respectively.

C. Latency Modeling

Many factors in the CMEC system cause latency, e.g., computation time at the SMDs, SBSs, and the CDC, and the transmission latency among them because of the wireless channel communication. The final task completion time needs to satisfy users' requirements. Accordingly, the latency model is constructed in this section.

The duration required for SMDs to receive results from the SBS can be neglected in this work according to [32]. T_{ij}^k represents the time required for SMD i to complete task k in both SBS j and the CDC, i.e.,

$$T_{ij}^k = \widetilde{T}_{ij}^k + \overline{T}_{ij}^k \tag{17}$$

where $\widetilde{T}_{ij}^k$ is the overall time required for task k in SMD i to upload, download, and process when using SBS j, and $\overline{T}_{ij}^k$ is the overall time required for task k in SMD i to upload, download, and process when using the CDC. $\widetilde{T}_{ij}^k$ is obtained as

$$\widetilde{T}_{ij}^k = \frac{\Theta_1\left(P_j^k + P^k\right) I_i^k}{\hat{R}_{ij}} + \frac{P_j^k I_i^k P_i^k}{f_{ij}^k} + \frac{\Theta_2\left(P_j^k + P^k\right) I_i^k}{\check{R}_{ij}}. \tag{18}$$

The first term in (18) represents the duration needed for task k to be uploaded from SMD i to SBS j while the second term denotes the processing time for task k to be completed by SMD i in SBS j. Besides, the third term represents the duration needed for task k to be downloaded from SBS j to SMD i. Additionally, Θ_1 and Θ_2 denote the uplink and downlink data transmission overhead between each SMD and each SBS, respectively.

SBSs and the CDC are connected through high-speed fibers, and tasks in SMDs are offloaded to SBSs and they can be further offloaded to the CDC. Hence

$$\overline{T}_{ij}^{k} = \frac{\Theta_3 P^k I_i^k}{r_t} + \frac{P^k I_i^k \alpha_i^k}{f_C} + \frac{\Theta_4 P^k I_i^k}{r_t}. \tag{19}$$

The first term in (19) represents the duration needed for task k to be uploaded from SBS j to the CDC, while the second term represents the processing time for task k to be completed by SMD i in the CDC. Finally, the third term represents the duration needed for task k to be downloaded from the CDC to SBS j. r_t is defined as the transmission rate of backhaul between SBSs and the CDC. Additionally, Θ_3 and Θ_4 denote the uplink and downlink data transmission overhead between each SBS and the CDC, respectively. In addition, f_C represents the computation speed of the CDC.

It should be mentioned that the computation conducted in SMDs and that at SBSs and the CDC works in parallel. Accordingly, the total time (T_i) required to execute all K tasks in the system is the maximum value of T_i^k and $\sum_{j=1}^{J} \mu_{ij} T_{ij}^k$, i.e.,

$$T_i = \sum_{k=1}^{K} \max\left(T_i^k, \sum_{j=1}^{J} \mu_{ij} T_{ij}^k \right). \tag{20}$$

Finally, the computation time must not surpass the predetermined time required by SMD i ($\hat{T}_i$), i.e.,

$$T_i \le \hat{T}_i. \tag{21}$$

D. Total Cost Model

The total cost of the CMEC system includes the cost of all SMDs, SBSs, and the CDC. Moreover, the final optimization goal is to minimize the system cost. In that case, the energy consumption model and the cost model of the CMEC system are formulated in this section.

E_{ij} represents the energy consumed by executing SMD i's offloaded tasks in SBS j, i.e.,

$$E_{ij} = \sum_{k=1}^{K} S_j I_i^k P_j^k \alpha_i^k (f_{ij}^k)^2 \tag{22}$$

where S_j is a fixed value established by the chip architecture of SBS j.

E_{i0} represents the energy consumed in executing SMD i's offloaded tasks in the CDC, i.e.,

$$E_{i0} = \sum_{k=1}^{K} \left(\frac{P^S \Theta_3 P^k I_i^k}{r_t} + \frac{P^C \Theta_4 P^k I_i^k}{r_t} + I_i^k P^k \alpha_i^k e_c \right) \tag{23}$$

where P^S and P^C represent the power of transmitting data from the uplink and downlink channels between an SBS and the CDC, respectively. e_c denotes the energy consumption per CPU cycle in the CDC.

Moreover, the energy consumption in both SBSs and the CDC must not surpass their predetermined limits, i.e.,

$$\sum_{i=1}^{N} \mu_{ij} E_{ij} \le \hat{E}_j \tag{24}$$

TABLE II
DECISION VARIABLES

Notation	Definition
μ_{ij}	Connection of SMD i to SBS j
P_i^k	Proportion of task k executed in SMD i
P_j^k	Proportion of task k executed in SBS j
P^k	Proportion of task k executed in the CDC
f_i^k	Running speed for executing task k in SMD i
P_i^t	Transmission power of SMD i
λ_{ij}	Channel bandwidth employed by SMD i in SBS j's channels
f_{ij}^k	Running speed at which task k from SMD i is executed by SBS j

where $\hat{E}_j$ denotes the maximum available energy of SBS j

$$\sum_{i=1}^{N} E_{i0} \le \hat{E}_c \tag{25}$$

where $\hat{E}_c$ denotes the maximum available energy of the CDC.

The total cost of the CMEC system ($\mathcal{F}$) includes three distinct parts, i.e., the cost of local computing ($\mathcal{F}_1$), the cost of edge computing ($\mathcal{F}_2$), and the cost of CDC ($\mathcal{F}_3$). In this case

$$\mathcal{F} = \mathcal{F}_1 + \mathcal{F}_2 + \mathcal{F}_3 \tag{26}$$

$$\mathcal{F}_1 = r_M \sum_{i=1}^{N} E_i \tag{27}$$

$$\mathcal{F}_2 = r_S \sum_{i=1}^{N} \sum_{j=1}^{J} E_{ij} \tag{28}$$

$$\mathcal{F}_3 = r_C \sum_{i=1}^{N} E_{i0} \tag{29}$$

where r_M, r_S, and r_C denote costs per unit of energy ($/kWh) in SMDs, SBSs, and the CDC, respectively.

E. Optimization Problem

In conclusion, Table II shows the decision variables of the system and our goal is to optimize $\mathcal{F}$, i.e.,

$$\underset{\chi}{\textbf{Min}} \quad \mathcal{F}$$

where χ represents a set of decision variables and it is subject to (1), (3), (6), (9), (12)–(16), (21), (24), and (25).

IV. Lévy Flights and Simulated Annealing-Based Grey Wolf Optimizer

This section explains the implementation details of LSAG. The objective function and its constraints are designed based on the offloading scenario discussed in Section III. We aim to find the optimal values of decision variables to yield our offloading strategy. The proposed offloading strategy aims to optimize the cost of the system while meeting time requirements of tasks.

Since $\mathcal{F}$ is nonlinear with respect to χ, it is a nonlinear constrained optimization problem. In this case, a penalty function approach is used to tackle the constraints. Specifically, it turns constraints into penalties and transforms the multiconstraint

problem into an unconstrained optimization problem. In this approach, each constraint is converted to a penalty with a nonnegative value. For instance, when the total penalty is zero, it indicates that all constraints are satisfied without any violation

$$\underset{\chi}{\mathbf{Min}} \left\{ \widetilde{\phi} = \overset{\infty}{\mathcal{N}} \mho + \phi \right\}. \tag{30}$$

In (30), $\widetilde{\phi}$ is an augmented objective function and $\overset{\infty}{\mathcal{N}}$ denotes a big positive number. $\mho$ denotes the sum of all penalties of all constraints, which is obtained as

$$\mho = \sum_{p=1}^{\mathbb{N}^{\neq}} (\max\{0, -g_p(\boldsymbol{x})\})^{\overset{0}{\gamma}_1} + \sum_{q=1}^{\mathbb{N}^{=}} \left| h_q(\boldsymbol{x}) \right|^{\overset{0}{\gamma}_2}. \tag{31}$$

In (31), $\mathbb{N}^{=}$ and $\mathbb{N}^{\neq}$ are numbers of equality and inequality constraints. $\overset{0}{\gamma}_1$ and $\overset{0}{\gamma}_2$ denote two positive numbers. An inequality constraint p ($1 \le p \le \mathbb{N}^{\neq}$) is then turned into $g_p(\boldsymbol{x}) \ge 0$. The penalty of p is $(\max\{0, -g_p(\boldsymbol{x})\})^{\overset{0}{\gamma}_1}$ if it is not satisfied, and it is zero otherwise. Similarly, an equality constraint q ($1 \le q \le \mathbb{N}^{=}$) is turned into $h_q(\boldsymbol{x}) = 0$. The penalty of q is $|h_q(\boldsymbol{x})|^{\overset{0}{\gamma}_2}$ if it is not satisfied, and 0 otherwise.

Some typical methods can solve unconstrained problems, e.g., nonlinear least squares and Levenberg–Marquardt methods. However, they require that optimization problems have certain mathematical structures. For instance, they demand the first or second-order derivatives [33]. To avoid such aforementioned disadvantages, many studies adopt evolutionary algorithms and swarm intelligence algorithms because they can easily be implemented to solve unconstrained problems and the results obtained are of high robustness. However, each optimization algorithm may bear some disadvantages. For example, although the convergence of GWO is fast, it is prone to fall into local optima when solving high-dimensional problems [34]. The genetic algorithm (GA) has great exploration ability but fails to balance its exploitation ability with poor search accuracy.

To avoid such drawbacks, this work proposes a novel two-stage optimization algorithm called LSAG. It comprises two stages. The first stage is used to decide edge selection (μ_{ij}) and the second one is used to decide other decision variables, including P_i^k, P_j^k, P^k, f_i^k, P_i^t, λ_{ij}, and f_{ij}^k.

The principle of low-requirement and low-capacity-first (LLF) is applied in the first stage of LSAG to establish the linkage between each SBS and each SMD. In other words, LLF is proposed to decide the connection between an SBS and an SMD. If an SMD (X_i) is only within the coverage of one SBS (S_j), $\mu_{ij} = 1$, e.g., SMD 1 is served by SBS 1 in Fig. 1 and thus $\mu_{11} = 1$. Otherwise, when LLF is employed, SMDs that require fewer resources are paired with SBSs that have fewer resource capacities. This leads to a situation where SBSs in a larger resource pool remain underutilized. In that case, SMDs with higher resource requirements are inclined to receive services directly from these underutilized and resource-abundant SBSs rather than depending on the CDC. The reduced reliance on CDC processing can lead to lower latency and cost. However, the resource requirement of an SMD is represented as a 2-D vector, encompassing CPU and memories, where each axis is a resource type. To compare different resource types uniformly, the needs across all types are normalized with the maximum norm. The overall resource requirements are then determined with the Euclidean norm.

In the second stage, LSAG optimizes the rest of decision variables. It adopts Latin hypercube sampling (LHS) to initialize the population for well covering the search space. Moreover, the attenuation factor a is pivotal in balancing exploration and exploitation during the optimization. The process is exploration when $a > 1$. Each grey wolf hunts with a probability of $(1/a)$, and searches with a probability of $1-(1/a)$. However, a fixed proportion of exploration and exploitation is difficult to adapt to the actual situation. Therefore, an adaptive attenuation factor that is controlled by γ is proposed in this work. It is assumed that t_1 is the current iteration count and $\hat{t}_1$ is the maximum one. Therefore, when $(t_1/\hat{t}_1) < \gamma$, a is updated with (32); otherwise, a is updated with (33)

$$a = -0.1 \times \frac{t_1}{\hat{t}_1} + 0.5 \tag{32}$$

$$a = 2 - \left(-0.1 \times \frac{t_1}{\hat{t}_1} + 0.5 \right). \tag{33}$$

Furthermore, to enhance the exploration capacity in complex scenarios, LSAG adopts the Lévy flight strategy, which supports broader searches in the decision space due to its heavy tail distribution [35]. There is a relatively high probability of taking long strides in the process of random walking. In this way, grey wolves can step out of local optima and take a big step in the search space during an optimization process for improving the global exploration ability. The distance between the third best wolf (δ) and the other wolves is D_δ, which is determined by the distance between the two best wolves and other wolves (D_α and D_β)

$$\sigma_u = \frac{\Gamma(1+\zeta)\sin\left(\frac{\pi\zeta}{2}\right)^{\frac{1}{\zeta}}}{\Gamma\frac{1+\zeta}{2}\zeta \times 2^{\frac{\zeta-1}{2}}} \tag{34}$$

$$\sigma_v = 1 \tag{35}$$

where ζ is a parameter for stabilization of Lévy flight, $u \sim N(0, {\sigma_u}^2)$, and $v \sim N(0, {\sigma_v}^2)$. Then, D_δ is updated as

$$D_\delta = \frac{1}{2}\left[\frac{u}{|v|^{-\zeta}} \left({X_i}^d - \alpha^d \right) + \frac{u}{|v|^{-\zeta}} \left({X_i}^d - \beta^d \right) \right] \tag{36}$$

where α^d and β^d denote the values of dimension d of the first two best wolves including α and β. Then, the new population X' is updated as

$${X'_i}^d = \frac{1}{2}\left[\alpha^d - A_1 \times D_\alpha + \beta^d - A_2 \times D_\beta \right] + D_\delta \tag{37}$$

where A_1 and A_2 denote coefficient vectors.

LSAG employs the Metropolis acceptance criterion from SA to choose candidates for the next iteration. It can accept moves of the population that could worsen the objective function [36]. Therefore, it can increase the population diversity, and individuals have a higher probability of targeting the global minimum more effectively. The acceptance probability is obtained as

$$p = e^{-\frac{\Delta}{T}} \tag{38}$$

Algorithm 1 LSAG

Input: Maximum number of iterations ($\hat{t}_1$), exploration proportion (γ), number of individuals (M), dimension number (D), and initial temperature (T)
Output: Final population

Initialize the population (X)
for each SMD i **do**
  if SMD i can be only served by SBS S_j **then**
    $\mu_{ij} = 1$
  end if
end for
Allocate SMD i to SBS j with the LLF
Choose the best individual α, and the suboptimal individual β
for t_1=1:$\hat{t}_1$ **do**
  for i=1:M **do**
    if $\frac{t_1}{\hat{t}_1} < \gamma$ **then**
      Calculate a with (32)
    else
      Calculate a with (33)
    end if
  end for
  for i=1:M **do**
    for d=1:D **do**
      Update A_1 and A_2 with the equation of $2 \times a \times r_1 - a$
      Update C_1 and C_2 with the equation of $2 \times r_2 - a$
      $D_\alpha = |C_1 \times \alpha^d - X_i{}^d|$
      $D_\beta = |C_2 \times \beta^d - X_i{}^d|$
      Calculate σ_u with (34)
      Calculate D_δ with (36)
      Update X_i' with (37)
      Calculate p with (38)
    end for
    if $\mathbf{f}(X_i') < \mathbf{f}(X_i)$ **then**
      $X_i = X_i'$
    else
      if $p > r_1$ **then**
        $X_i = X_i'$
      else
        $X_i = X_i$
      end if
    end if
  end for
end for
return X

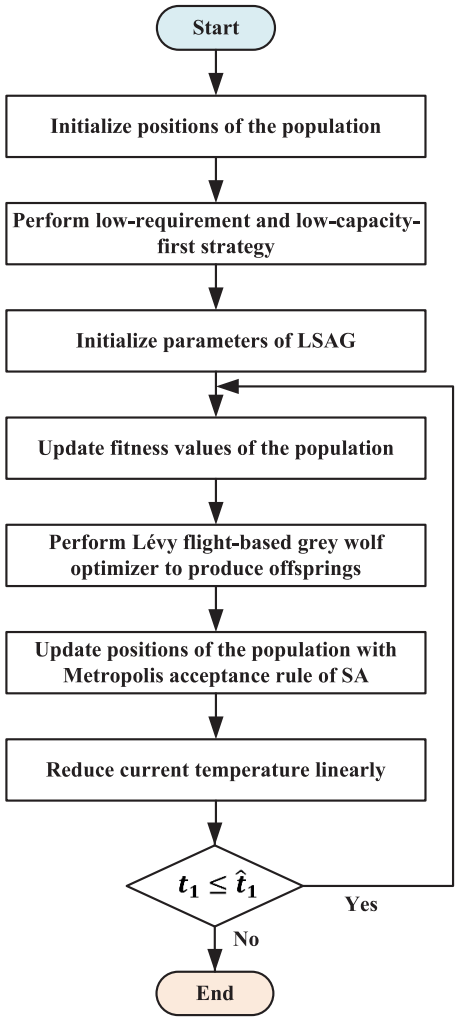


Fig. 3. Flowchart of LSAG.

where p is the acceptability and T is the initial temperature. Δ is the difference of objective function values before and after each iteration.

LSAG is realized in Algorithm 1 and its flowchart is shown in Fig. 3, where $\mathbf{f}(\cdot)$ is the fitness function that calculates the total cost of the system. In addition, we discuss the time complexity of LSAG. The primary source of computation overhead is the **for** loop, which halts when the number of iterations reaches $\hat{t}_1$. Moreover, the time complexity per iteration is $\mathcal{O}(DN)$. Consequently, LSAG has the overall time complexity of $\mathcal{O}(\hat{t}_1 DN)$. It is worth noting that LSAG is executed in SBSs. The reasons are given as follows. First, SBSs have more computational resources than SMDs. Therefore, they have enough computational resources and performance to execute LSAG. Second, SBSs communicate with SMDs directly. In this case, SMDs report their current states like residual energy to SBSs and SBSs can collect performance metrics of SMDs. Moreover, LSAG can be better leveraged by implementing it in high-performance SBSs.

V. Performance Evaluation

This work evaluates LSAG with realistic tasks from Google data centers for one day. To simulate the input data of each SMD, tasks from Google data centers are collected every five minutes. Moreover, each time slot also has a duration of five minutes. LSAG is implemented in MATLAB R2021b, running on a computer with 16-GB RAM and an Intel i7-10700F CPU with 2.90 GHz.

A. Parameter Setting

According to [37], parameters of SMDs, SBSs, and the CDC are set in Tables III and IV. Moreover, parameters of LSAG are set as: $\gamma = 0.5$ for balancing exploration and exploitation, $\zeta = 1.5$ according to [38], and $T = 1000$ according to [39].

B. Experimental Results

We compare LSAG with its three typical benchmark peers, including GA [40], genetic learning particle swarm optimization (GLPSO) [41], and GWO [42]. Moreover, $M =$

TABLE III
PARAMETER SETTING—PART I

N	J	α_i^k	χ_i^k	S_i	$\hat{P}_i^t$	d_{ij}	f_1 (f_2)	ω_0	$\hat{B}_j$ ($\hat{B}_j$)	P_j^t	λ_{ij}
5-40	3	[25,100] cycles/bit	[25,100] memory/bit	[0.2,1.4]$\times 10^{-26}$	0.15 W	[10,150] m	0.73	1.6×10^{-11}	[5,30] MHz	0.1 W	[5,30] MHz

TABLE IV
PARAMETER SETTING—PART II

$\hat{C}_S^j$ ($\hat{M}_S^j$)	$\hat{C}_{CDC}$ ($\hat{M}_{CDC}$)	Θ_1	Θ_2	Θ_3	Θ_4	$\hat{T}_i$	S_j	e_c	$\hat{E}_j$	$\hat{E}_c$	r_M	r_S	r_C
3×10^{10} Hz (3 GB)	9×10^{10} Hz (9 GB)	1	0.3	1	0.3	[3,7] S	[0.3,1.5]$\times 10^{-27}$	1 W/GHz	13 J	45 J	0.04 $/KWH	0.01 $/KWH	0.03 $/KWH

50 and $\hat{t}_1 = 1000$ for all compared algorithms. Their advantages are summarized as follows.

1) GA combines genetic operations, e.g., crossover, selection, and mutation. In that case, its individual diversity is high. As a result, it has good global search ability and can search all solutions. Thus, the comparison between GA and LSAG proves search efficiency of LSAG.
2) GLPSO combines merits of GA and particle swarm optimization (PSO), and therefore, GLPSO inherits the search accuracy of PSO and excellent search efficiency of GA. Thus, comparing GLPSO with LSAG and GLPSO proves the exploration speed and search accuracy of LSAG.
3) GWO is the base optimizer of LSAG. Moreover, it has strong convergence performance. Therefore, the comparison between LSAG and GWO proves LSAG's convergence performance.

Figs. 4 and 5 show the cost and associated penalties for LSAG, GA, GWO, and GLPSO in the CMEC system with ten SMDs after 1000 iterations, it is shown that LSAG outperforms the other algorithms by minimizing the cost ($0.048) after its iteration. Moreover, GWO obtains its best solution after 920 iterations, which is much larger than LSAG. Although the high diversity of the population in GA helps it to find a solution quickly. However, it traps into local optima and has the highest cost after 1000 iterations. Furthermore, it is shown in Fig. 5 that the penalty of GLPSO is 0.0015 after 1000 iterations. In addition, the penalty of GA also fails to achieve zero, which is 0.02 in Fig. 5 after 900 iterations. The result proves that GA and GLPSO cannot generate high-quality solutions meeting all the constraints. The penalties associated with both LSAG and GWO end up reaching zero values after the iterations. It confirms the validity of they yielded solutions. Nevertheless, GWO derives a feasible solution after 200 iterations, yet takes longer time than LSAG. It is worth noting that LSAG starts with a lower penalty and reaches the zero penalty after only 100 iterations, which outperforms other algorithms. As a result, it is hard for benchmark algorithms to well balance search efficiency and accuracy in the formulated problem. Specifically, GLPSO and GA cannot find desired solutions meeting all the constraints after their required iterations. GWO finds valid solutions but it has low search efficiency. Therefore, LSAG has better search efficiency and accuracy than state-of-art algorithms, and it can better solve the offloading problem.

Fig. 6 shows the cost and penalty of LSAG for varying numbers of SMDs. It demonstrates that the penalty of LSAG is always zero, which justifies that LSAG finds solutions that meet all constraints. The total cost of the four algorithms for different numbers of SMDs is presented in Fig. 7, it is illustrated that LSAG attains the lowest cost when N ranges from 0 to 40. Moreover, it is shown that the cost of GWO and GA increases faster after 25 SMDs, which proves that GWO and GA cannot well explore the search space when the

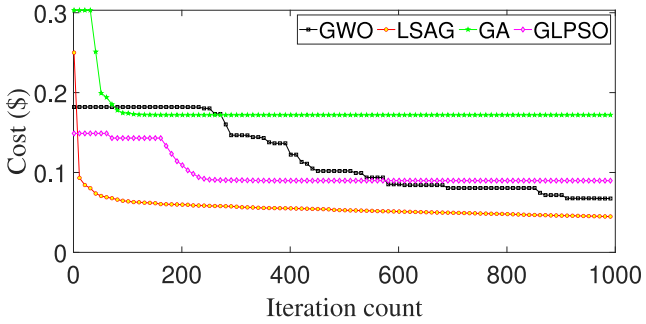


Fig. 4. Iteration curve of the cost given ten SMDs.

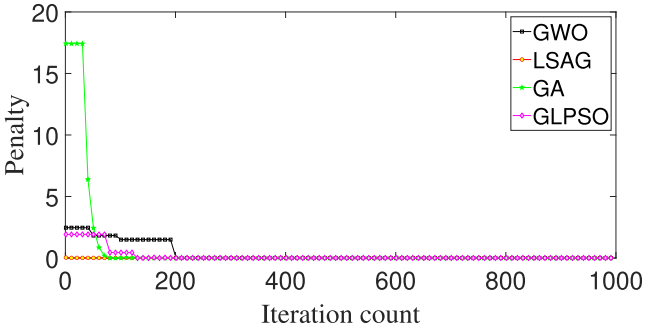


Fig. 5. Iteration curve of the penalty given ten SMDs.

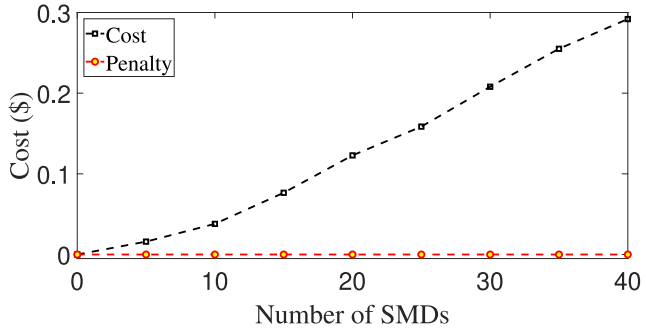


Fig. 6. Total cost and penalty with different N.

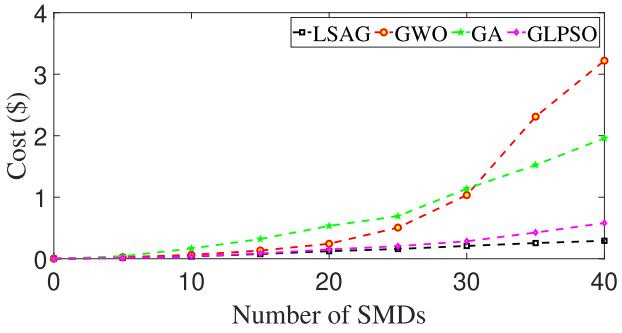


Fig. 7. Total cost versus different N.

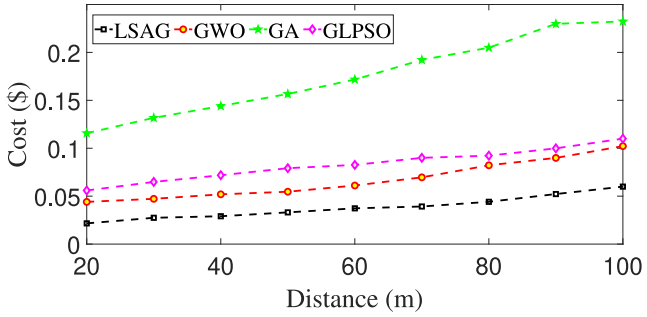


Fig. 8. Total cost versus different d.

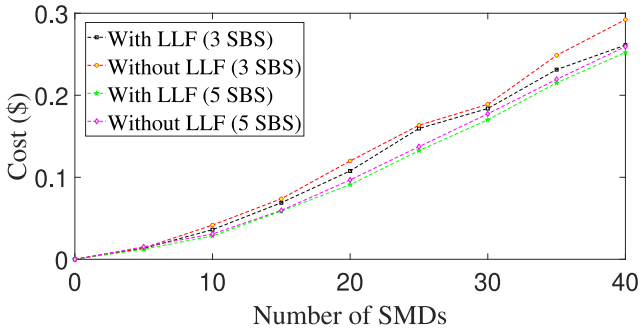


Fig. 9. Cost with and without LLF.

number of SMDs is large. Thus, LSAG achieves higher global exploration ability and yields the best result.

The total cost of four algorithms against various distances between SMDs and SBSs is shown in Fig. 8. The scenario assumes a fixed number of ten SMDs. It is shown that the final cost of all algorithms increases with distance because the increasing distance between SMDs and SBSs results in larger transmission energy. Moreover, LSAG achieves the minimum cost across all distances for all algorithms. Fig. 9 shows the cost with the LLF strategy and without it for varying numbers of SMDs and SBSs. It illustrates that LLF can reduce the system cost and more SBSs reduce the total cost because LSAG can better allocate sufficient resources among SBSs and reduce the cost of the CDC. Moreover, Fig. 10 illustrates the cost of LSAG for different $\hat{T}_i$ and N. It is illustrated that LSAG can be satisfactorily solved the problem under different delay constraints.

Furthermore, Fig. 11(a)–(d) illustrates the cost of SMDs, SBSs, the CDC, and the total one of LSAG with respect to the number of SMDs. We also include three other strategies, including random offloading, local computing, and full

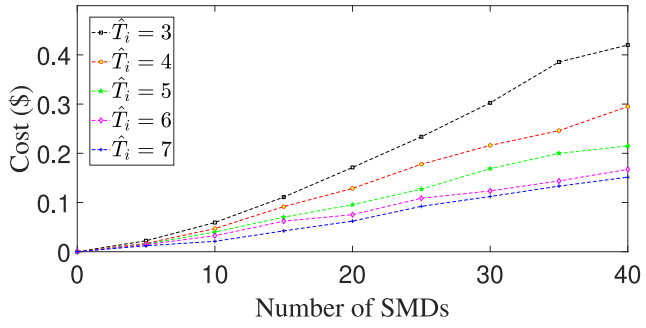


Fig. 10. Cost with different $\hat{T}_i$ and N in LSAG.

offloading [43]. Fig. 11(a) shows that LSAG outperforms random and local computing in terms of SMDs' total cost for different numbers of SMDs. Its cost is higher than full offloading, resulting in larger transmission delays. The offloading strategy aims to minimize energy consumption while meeting the time requirements of tasks. Thus, the latency of full offloading is not acceptable. Moreover, Fig. 11(b) illustrates that the cost of SBS with LSAG is \$0.16 for the case with 40 SMDs. It is lower than random offloading and full offloading. Local computing executes all tasks in SMDs and does not involve SBSs and the CDC. However, it requires large computation time and it does not meet the latency requirements of tasks.

Fig. 11(c) shows that the proposed strategy outperforms the random and full offloading with respect to the cost of the CDC. It is higher than that with local computing that does not involve the CDC. Furthermore, Fig. 11(d) shows the total cost with the proposed offloading strategy for the number of SMDs. It can be noted that LSAG has the lowest cost among almost all strategies. In addition, the iteration curves provided with ten SMDs are shown in Fig. 11(e), and the penalty for each strategy with various numbers of SMDs is shown in Fig. 11(f). It can be observed that all the strategies except LSAG have nonzero penalty values after their total iterations. To sum up, LSAG achieves the best results and meets all the constraints.

VI. Conclusion

Nowadays, SMDs substantially influence our daily lives. However, their confined battery power and computational capabilities make it challenging to complete all tasks within the limited time required by users. MEC is introduced to solve the problem. However, the challenge remains for existing computation offloading strategies to achieve a balance between resource allocation and energy consumption while meeting the latency requirements of SMDs. This work designs a CMEC framework for partial computation offloading. Moreover, a constrained optimization problem targeting cost minimization is established and addressed by a two-stage optimization algorithm, named LSAG. The first stage of LSAG determines the optimal edge selection policy when multiple SBSs are available to an SMD, while the second stage aims to optimize resource allocation in the system, thereby minimizing the system's total cost. LSAG integrates the merits of Lévy flight and SA into a GWO to enhance the global search capabilities and the ability to escape from local optima. Experiments based on real-world data

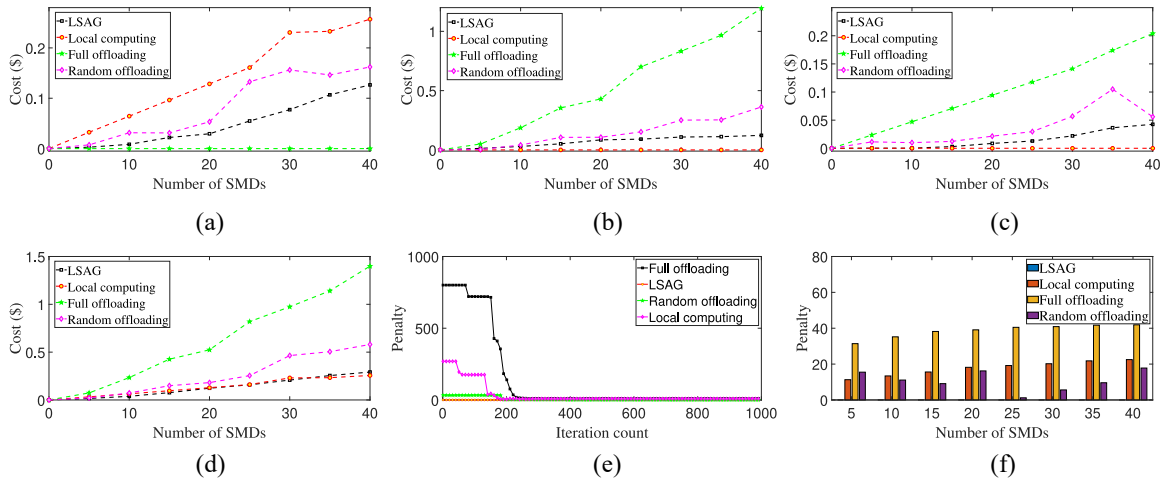


Fig. 11. Results of different strategies. (a) Cost of SMDs versus different N. (b) Cost of SBSs versus different N. (c) Cost of the CDC versus different N. (d) Total cost versus different N. (e) Penalty of ten SMDs. (f) Penalty of each strategy.

demonstrate that LSAG outperforms the compared algorithms by achieving lower system costs in fewer iterations.

In the future, we intend to enhance the LSAG framework to accommodate scenarios involving the mobility of SMDs and SBSs, subsequently conducting a thorough performance analysis in these dynamic settings. In addition, we will also consider applying LSAG to solve multiobjective optimization problems with more objectives, e.g., user experience.

REFERENCES

[1] R. Cong, Z. Zhao, G. Min, C. Feng, and Y. Jiang, "EdgeGO: A mobile resource-sharing framework for 6G edge computing in massive IoT systems," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14521–14529, Aug. 2022.

[2] C. Chen, J. Zhang, X. Chu, and J. Zhang, "On the optimal base-station height in mmWave small-cell networks considering cylindrical blockage effects," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9588–9592, Sep. 2021.

[3] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, Mar. 2021.

[4] H. Seo, H. Oh, J. K. Choi, and S. Park, "Differential pricing-based task offloading for delay-sensitive IoT applications in mobile edge computing system," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 19116–19131, Oct. 2022.

[5] A. Belgacem, K. Beghdad-Bey, and H. Nacer, "Dynamic resource allocation method based on symbiotic organism search algorithm in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1714–1725, Jul.–Sep. 2022.

[6] J. Wang, P. Pinson, S. Chatzivasileiadis, M. Panteli, G. Strbac, and V. Terzija, "On machine learning-based techniques for future sustainable and resilient energy systems," *IEEE Trans. Sustain. Energy*, vol. 14, no. 2, pp. 1230–1243, Apr. 2023.

[7] S. A. Mohamed, S. Sorour, and H. S. Hassanein, "Group delay-aware scalable mobile edge computing using service replication," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 11911–11920, Nov. 2022.

[8] M. Chen, X. Gong, and Y. Cao, "Delay-optimal distributed edge computation offloading with correlated computation and communication workloads," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5846–5857, Oct. 2023.

[9] J. Shi, Y. Zhou, Z. Li, Z. Zhao, Z. Chu, and P. Xiao, "Delay minimization for NOMA-mmW scheme-based MEC offloading," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2285–2296, Feb. 2023.

[10] L. Ai, B. Tan, J. Zhang, R. Wang, and J. Wu, "Dynamic offloading strategy for delay-sensitive task in mobile-edge computing networks," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 526–538, Jan. 2023.

[11] X. Pu et al., "Incentive mechanism and resource allocation for collaborative task offloading in energy-efficient mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 72, no. 10, pp. 13775–13780, Oct. 2023.

[12] M. Bolourian and H. Shah-Mansouri, "Energy-efficient task offloading for three-tier wireless-powered mobile-edge computing," *IEEE Internet Things J.*, vol. 10, no. 12, pp. 10400–10412, Jun. 2023.

[13] J. Bi, Z. Wang, H. Yuan, and J. Zhang, "Cost-minimized partial computation offloading in cloud-assisted mobile edge computing systems," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2023, pp. 1–6.

[14] X. Xiao et al., "Novel workload-aware approach to mobile user reallocation in crowded mobile edge computing environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8846–8856, Jul. 2022.

[15] X. Ma, A. Zhou, S. Zhang, Q. Li, A. X. Liu, and S. Wang, "Dynamic task scheduling in cloud-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2116–2130, Apr. 2023.

[16] J. Wang, D. Feng, S. Zhang, A. Liu, and X.-G. Xia, "Joint computation offloading and resource allocation for MEC-enabled IoT systems with imperfect CSI," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3462–3475, Mar. 2021.

[17] Y. Li, B. Yang, H. Wu, Q. Han, C. Chen, and X. Guan, "Joint offloading decision and resource allocation for vehicular fog-edge computing networks: A contract-stackelberg approach," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15969–15982, Sep. 2022.

[18] M. Chen, S. Guo, K. Liu, X. Liao, and B. Xiao, "Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 2025–2040, May 2021.

[19] J. Yun, Y. Goh, W. Yoo, and J. Chung, "5G Multi-RAT URLLC and eMBB dynamic task offloading with MEC resource allocation using distributed deep reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20733–20749, Oct. 2022.

[20] M. Merluzzi, N. D. Pietro, P. Di Lorenzo, E. C. Strinati, and S. Barbarossa, "Discontinuous computation offloading for energy-efficient mobile edge computing," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 2, pp. 1242–1257, Jun. 2022.

[21] L. Shi, Y. Ye, X. Chu, and G. Lu, "Computation energy efficiency maximization for a NOMA-Based WPT-MEC network," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10731–10744, Jul. 2021.

[22] F. Song et al., "Evolutionary multi-objective reinforcement learning based trajectory control and task offloading in UAV-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 12, pp. 7387–7405, Dec. 2023.

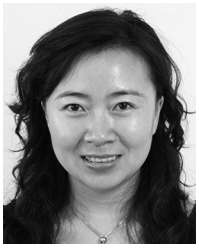
[23] B. Huang, M. Zhou, G. Zhang, A. C. Ammari, A. Alabdulwahab, and A. G. Fayoumi, "Lexicographic multiobjective integer programming for optimal and structurally minimal petri net supervisors of automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 11, pp. 1459–1470, Nov. 2015.

[24] X. Guo, M. Zhou, S. Liu, and L. Qi, "Lexicographic multiobjective scatter search for the optimization of sequence-dependent selective disassembly subject to multiresource constraints," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3307–3317, Jul. 2020.

[25] F. Guim et al., "Autonomous lifecycle management for resource-efficient workload orchestration for green edge computing," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 571–582, Mar. 2022.

[26] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[27] K. Guo, M. Yang, Y. Zhang, and J. Cao, "Joint computation offloading and bandwidth assignment in cloud-assisted edge computing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 451–460, Jan.–Mar. 2022.
[28] V. V. Chetlur and H. S. Dhillon, "On the load distribution of vehicular users modeled by a Poisson line cox process," *IEEE Wireless Commun. Lett.*, vol. 9, no. 12, pp. 2121–2125, Dec. 2020.
[29] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Minimizing the delay and cost of computation offloading for vehicular edge computing," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2897–2909, Sep./Oct. 2022.
[30] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1133–1145, May 2021.
[31] M. A. van Wyk, L. Ping, and G. Chen, "Multivaluedness in networks: Shannon's noisy-channel coding theorem," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 10, pp. 3234–3235, Oct. 2021.
[32] X. Xia et al., "OL-MEDC: An online approach for cost-effective data caching in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1646–1658, Mar. 2023.
[33] A. Chakrabarty, D. K. Jha, G. T. Buzzard, Y. Wang, and K. G. Vamvoudakis, "Safe approximate dynamic programming via kernelized lipschitz estimation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 405–419, Jan. 2021.
[34] J. Bi, Z. Wang, H. Yuan, J. Zhang, and M. Zhou, "Self-adaptive teaching-learning-based optimizer with improved RBF and sparse autoencoder for high-dimensional Problems," *Inf. Sci.*, vol. 630, pp. 463–481, Jun. 2023.
[35] F. Jiang, L. Dong, K. Wang, K. Yang, and C. Pan, "Distributed resource scheduling for large-scale MEC systems: A multiagent ensemble deep reinforcement learning with imitation acceleration," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6597–6610, May 2022.
[36] X. Zhou, S. Li, and Y. Feng, "Quantum circuit transformation based on simulated annealing and heuristic search," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4683–4694, Dec. 2020.
[37] 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. 2022.
[38] H. Liu, C. Li, S. He, W. Shi, Y. Chen, and W. Shi, "Simulated annealing particle swarm optimization for a dual-input broadband GaN doherty like load-modulated balance amplifier design," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 9, pp. 3734–3738, Sep. 2022.
[39] Y. Zhou, W. Xu, Z.-H. Fu, and M. Zhou, "Multi-neighborhood simulated annealing-based iterated local search for colored traveling salesman problems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16072–16082, Sep. 2022.
[40] M. Xu, G. Feng, Y. Ren, and X. Zhang, "On cloud storage optimization of blockchain with a clustering-based genetic algorithm," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8547–8558, Sep. 2020.
[41] Y. Gong, J. Li, Y. Zhou, Y. Li, H. Chung, Y. Shi, and J. Zhang, "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.
[42] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, no. 10, pp. 46–61, Mar. 2014.
[43] J. Choi, "Random-access-based multiuser computation offloading for devices in IoT applications," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 22034–22043, Nov. 2022.



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 2011 to 2013, she was a Research Scientist with the Beijing Research Institute of Electronic Engineering Technology, Beijing. From 2009 to 2010, she was a Research Assistant and participated in research on cloud computing with IBM Research, Beijing. From 2018 to 2019, she was a Visiting Research Scholar with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. She is currently a Professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing. She has over 150 publications in international journals and conference proceedings. Her research interests include distributed computing, cloud and edge computing, large-scale data analytics, machine learning, industrial Internet, and performance optimization.

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

Ziqi Wang (Student Member, IEEE) received the B.E. degree in Internet of Things from Beijing University of Technology, Beijing, China, in 2022, where he is currently pursuing the master's degree with the Faculty of Information Technology, School of Software Engineering.

His research interests include cloud computing, task scheduling, intelligent optimization algorithms, and machine learning.

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 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.

Dr. Yuan received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, and the Best Paper Award in the 17th ICNSC. He serves as an Associate Editor for *Expert Systems with Applications*.

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

She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in 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 graph, 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.

In 1990, he joined New Jersey Institute of Technology, Newark, NJ, USA, where he is currently a Distinguished Professor. He has over 900 publications, including 12 books, 600+ journal papers (450+ in IEEE transactions), 28 patents, and 29 book chapters. His interests are in Petri nets, automation, Internet of Things, and big data.

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