2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 1-4, 2023, Oahu, Hawaii, USA

Cost-Minimized Partial Computation Offloading in Cloud-Assisted Mobile Edge Computing Systems

Jing Bi[1], Ziqi Wang[1], Haitao Yuan[2] and Jia Zhang[3]

[1]Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[2]School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
[3]Dept. of Computer Science in Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA

2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 979-8-3503-3702-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/SMC53992.2023.10393954

***Abstract*—Nowadays, smart mobile devices (SMDs) support various computation-intensive and delay-sensitive applications, *e.g.*, online games, and figure compression. However, SMDs have limited computing resources and battery energy and cannot execute all tasks of the above applications in a real-time manner. Cloud computing provides enormous computing resources and energy that can easily execute tasks offloaded from SMDs. However, could data centers (CDCs) are often located in remote sites, which leads to long transmission time. Small base stations (SBSs) offer high-bandwidth and low-latency services for SMDs, which solves the problem of cloud computing. However, it becomes a challenge to achieve the lowest cost in such a heterogeneous architecture including multiple SMDs, SBSs, and the CDC while meeting delay requirements of tasks. This work proposes a cost-minimized computation offloading strategy to minimize the total cost of the system. A constrained optimization problem is first formulated based on the hybrid architecture. Afterward, a two-stage optimization algorithm called a Lévy flights and Simulated Annealing-based Grey wolf optimizer (LSAG) is developed to optimize the total cost of the system. In the first stage, the optimal edge selection policy is determined given multiple available SBSs. In the second stage, task offloading and resource allocation among SMDs, SBSs, and the cloud are determined. Experiments with real-life tasks prove that LSAG significantly achieves lower cost with faster convergence speed than state-of-the-art peers.**

***Index Terms*—Mobile edge computing, computation offloading, cloud computing, Lévy flights, intelligent optimization algorithms**

I. INTRODUCTION

Over the past few years, there has been a significant increase in the usage of smart mobile devices (SMDs) and wireless communications technologies. Therefore, various applications including mobile games, online meetings in SMDs greatly facilitate and enrich people's daily lives [1]. However, some applications require enormous computing resources and the energy of SMDs. Due to the limited computing resources and battery energy of SMDs, local execution of these applications becomes strained [2]. Moreover, a large amount of energy consumption depletes the battery performance and reduces the lifetime of SMDs. On the other hand, cloud computing provides enormous computing resources to solve this problem. However, cloud servers are often deployed in remote areas at great distances [3]. Therefore, stringent delay requirements of various applications in SMDs cannot be met by the centralized cloud computing paradigm due to the inevitable long distances between SMDs and servers in the cloud.

Edge computing provides a solution to avoid large transmission latency in cloud computing. Small base stations (SBSs) are more flexible to be deployed in crowded areas [4]. In this way, SBSs provide SMDs with close-proximity, high-bandwidth, and low-latency services [5]. However, computing resources in the edge are usually not as sufficient as that in the cloud. As a result, SBSs and remote could data centers (CDCs) are usually connected with fiber links with low latency [6]. Thus, a cloud-assisted mobile edge computing (MEC) architecture is constructed to process mobile applications that require plenty of computing resources and have strict latency requirements.

However, there are still three issues that demand attention. The first one is the latency in communication. Specifically, the extra process of task offloading from SMDs to SBSs, and from SBSs to the cloud unavoidably causes additional communication latency [7]. The second one is resource allocation. With a large number of SMDs offloading tasks to different SBSs that may further offload tasks to the cloud, it becomes an issue of how to efficiently allocate resources in each SMD and SBS [8]. Finally, the total cost of the cloud-assisted MEC system comprises of costs involving SMDs, SBSs, and the cloud. Therefore, it has been recognized critical to realize cost minimization for the system in such a complex and heterogeneous environment.

Motivated by the above analysis, this work proposes a cost-minimized computation offloading technique. As the first step, we propose a fundamental unit architecture comprising of multiple SMDs, SBSs, and the cloud. Then, a constrained optimization problem is formulated based on this architecture. Moreover, we propose a two-stage optimization algorithm called Lévy flights and Simulated Annealing-based Grey wolf optimizer (LSAG). It incorporates Lévy flights strategy and a metropolis acceptance criterion of Simulated Annealing (SA) into Grey Wolf Optimizer. LSAG simultaneously optimizes edge selection, energy consumption, and resource allocation in this cloud-assisted MEC system. Experiments with real-life tasks from Google data centers reveal that LSAG realizes cost-efficient computation offloading in a cloud-assisted MEC architecture.

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62173013 and 62073005, the Beijing Natural Science Foundation under Grant 4232049, and the Fundamental Research Funds for the Central Universities under Grant YWF-22-L-1203.

II. PROBLEM FORMULATION

To study the cost minimization problem of the cloud-assisted MEC system, we propose a unit architecture as illustrated in Fig. 1. In this system, each SMD is connected to one SBS. This work considers computation-intensive tasks that are independent from each other. For example, virus scan applications can be split into multiple logically independent tasks and each task can be executed in SMDs or SBSs/cloud by using partial computation offloading.

In this proposed architecture, if SMD i $(1\leq i\leq N)$ is connected to SBS j $(1\leq j\leq J)$, $\mu_{ij}=1$; otherwise, $\mu_{ij}=0$. Moreover, each task can be executed on SMDs, SBSs and/or the cloud. P_i^k, P_j^k and P^k denote the proportions of task k executed in SMD i, SBS j and the cloud. As a result, the sum of P_i^k, P_j^k and P^k is one, *i.e.*,

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

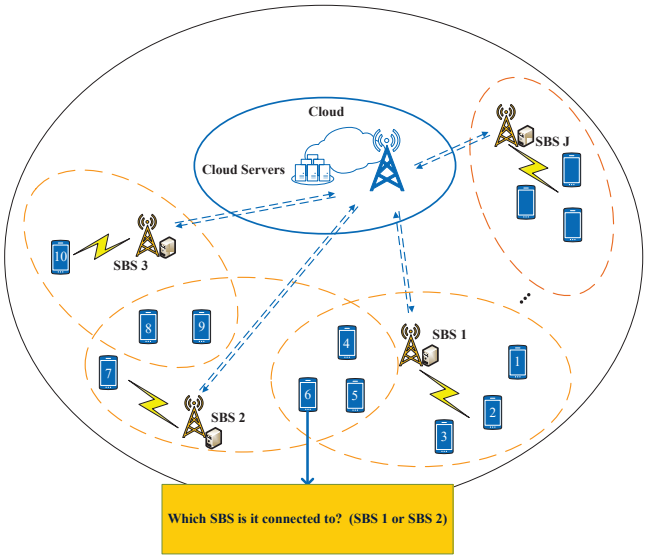

Fig. 1. Architecture of the cloud-assisted MEC system.

A. Modeling of SMDs

T_i^k denotes the average time for executing task k in SMD i and it can be obtained as:

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

where I_i^k denotes the size of input data of task k received by SMD i, α_i^k means the number of CPU cycles required by each bit of input data of task k executed in SMD i, and f_i^k means the CPU running speed for executing task k in SMD i.

For each SMD i, the CPU speed for executing all tasks in SMD i cannot exceed its maximum CPU speed (F_i), *i.e.*,

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

P_i^k denotes the power consumption of executing task k in SMD i. Then, P_i^k is obtained as:

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

where S_i is a constant determined by the chip architecture of SMD i.

Moreover, E_i^1 is the energy consumption of executing all K tasks in SMD i, which is obtained as:

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

For each SMD i, it needs to transmit data to the SBS for computing. P_i^t denotes the transmitting power from SMD i to the SBS. Moreover, it cannot exceed its maximum transmitting power $(\hat{P}_i^t)$, *i.e.*,

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

Therefor, the energy consumption of data transmission from SMD i to its corresponding SBS is E_i^2, which is obtained as:

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

where T_i^t denotes the time of transmitting tasks from SMD i to its corresponding SBS.

The total energy consumed by each SMD i (E_i) consists of two parts. The first part is the energy consumed during local execution (E_i^1) and the second part is the energy required for transmitting data to the SBS (E_i^2), *i.e.*,

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

B. Modeling of SBS and the cloud

d_{ij} denotes the distance from SMD i to SBS j. According to [9], the path loss between SMD i and SBS j is obtained as $(d_{ij})^{-v}$ where v is the path loss parameter. λ_{ij} denotes the bandwidth in SBS j's uplink and downlink channels used for SMD i. Moreover, the total bandwidth of SBS j allocated for all SMDs is one, *i.e.*,

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

B_j^U and B_j^D denote the bandwidth of uplink and downlink channels for SBS j. R_{ij}^U and R_{ij}^D denote uplink and downlink rates between SMD i and SBS j. Based on Shannon's theorem [10], they are obtained as:

$$R_{ij}^U=\lambda_{ij}B_j^U log_2(1+\frac{P_i^t(d_{ij})^{-v}|f_1|^2}{\omega_0}) \tag{10}$$

$$R_{ij}^D=\lambda_{ij}B_j^D log_2(1+\frac{P_j^t(d_{ij})^{-v}|f_2|^2}{\omega_0}) \tag{11}$$

where P_j^t means the transmitting power of SBS j to each SMD, f_1 and f_2 denote fading coefficients of uplink and downlink channels, and ω_0 denotes the power parameter of white Guassian noise.

Moreover, f_{ij}^k denotes the running speed for executing task k from SMD i in SBS j and the running speed of all tasks in SBS j cannot exceed its maximum 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 required by tasks executed in SBS j need to be less or equal to its limit $\hat{C}_S^j$, *i.e.*,

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

Moreover, the number of memories required by tasks executed in SBS j cannot exceed its limit $\hat{M}_S^j$, *i.e.*,

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

where χ_i^k denotes the amount of memory for executing task k in SMD i.

Similarly, the number of CPU cycles and memories required by tasks executed in the cloud cannot exceed its limit, *i.e.*,

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

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

where $\hat{C}_{CDC}$ and $\hat{M}_{CDC}$ denote maximum CPU cycles and memory in the cloud, respectively.

C. Latency model

The cloud-assisted MEC system consists of multiple time consumption parts including computation and data transmission among SMDs, SBSs, and the cloud.

T_{ij}^k denotes the total time of executing task k of SMD i in SBS j and the cloud. It is obtained as:

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

where $\widetilde{T}_{ij}^k$ is the total time of data uploading, downloading, and executing for task k of SMD i in SBS j and $\overline{T}_{ij}^k$ denotes the total time of data uploading, downloading between SBS j and the cloud. Moreover, it also contains the time of processing task k of SMD i in the cloud. $\widetilde{T}_{ij}^k$ is obtained as:

$$\widetilde{T}_{ij}^k = \frac{O_1(P_j^k + P^k)I_i^k}{R_{ij}^U} + \frac{P_j^k I_i^k P_i^k}{f_{ij}^k} + \frac{O_2(P_j^k + P^k)I_i^k}{R_{ij}^D} \tag{18}$$

where O_1 and O_2 are the overhead of transmitting data in each uplink and downlink channel from each SMD to each SBS and from each SBS to each SMD, respectively.

Moreover, tasks in SMDs are offloaded to an SBS and it can be further offloaded to the cloud. Therefore, $\overline{T}_{ij}^k$ is obtained as:

$$\overline{T}_{ij}^k = \frac{O_3 P^k I_i^k}{r_t} + \frac{P^k I_i^k \alpha_i^k}{f_C} + \frac{O_4 P^k I_i^k}{r_t} \tag{19}$$

where r_t denotes the transmission rate between SBSs and the cloud. O_3 and O_4 denote the overhead of transmitting data from each SBS to the cloud, and that from the cloud to each SBS, respectively. Moreover, f_C denotes the computational speed of the cloud.

The total time (T_i) by executing all K tasks in the system is the maximum value of local computation and edge computation, which is obtained as:

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

Moreover, the total computational time cannot exceed the upper limit required by the user ($\hat{T}_i$), *i.e.*,

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

D. Total cost model

E_{ij} denotes the energy consumption of executing SMD i's offloaded tasks in SBS j and it is obtained as:

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

where S_j is a constant determined by SBS j's chip architecture.

Moreover, E_{i0} is the energy consumption of executing SMD i's offloaded tasks in the cloud and it is obtained as:

$$E_{i0} = \sum_{k=1}^{K}\left(\frac{P_S^C O_3 P^k I_i^k}{r_t} + \frac{P_C^S O_4 P^k I_i^k}{r_t} + I_i^k P^k \alpha_i^k e_0\right) \tag{23}$$

where P_S^C and P_C^S denote data transmission power from SBS to the cloud, and that from the cloud to SBS, respectively. e_0 means the energy consumption of each CPU cycle in the cloud.

Moreover, the total energy consumption in SBS and the cloud cannot exceed their corresponding limit, *i.e.*,

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

where $\hat{E}_j$ means the maximum energy of SBS j.

$$\sum_{i=1}^{N}E_{i0} \le \hat{E}_0 \tag{25}$$

where $\hat{E}_0$ means the maximum energy in the cloud.

Finally, F denotes the total cost of the cloud-assisted MEC system, which consists of three parts, *i.e.*, the cost of the local SMDs (F_L), the cost of all SBSs (F_S) and the cost of the cloud (F_C). Then,

$$F = F_L + F_S + F_C \tag{26}$$

$$F_L = r_M \sum_{i=1}^{N} E_i \tag{27}$$

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

$$F_C = r_C \sum_{i=1}^{N} E_{i0} \tag{29}$$

where r_M, r_S, and r_C are prices (\$/KWH) of energy in SMDs, SBSs, and the cloud, respectively.

III. Lévy flights and Simulated Annealing-based Grey wolf optimizer (LSAG)

Given the constrained optimization problem (F), our proposed LSAG is given as follows. F is nonlinear with respect to decision variables including μ_{ij}, P_i^k, P_j^k, $P^k, f_i^k, P_i^t, \lambda_{ij}, f_{ij}^k$. To handle the above constraints, a penalty function method [11] is adopted to transform all constraints into the penalty and convert this problem into an unconstrained optimization one. Moreover, LSAG comprises of two stages. The first stage is used to determine the edge selection (μ_{ij}) and the second one is used to determine other decision variables.

Low-requirement and low-capacity-first (LLF) principles are proposed in the first stage of LSAG to decide the association between each SBS and each SMD. For example, LLF decides which SBS is used to serve SMD 6 in Fig. 1. In the proposed LLF, users with lower resource requirements are allocated to SBSs with fewer resources. In this way, SBSs with more resources have a higher probability of idling. Therefore, users with more resource requirements have more possibility to be directly served by a SBS with more resources instead of the cloud. As a result, LLF can reduce the number of tasks allocated to the cloud for reducing latency and cost. However, the computing resource requirement of an SMD is represented as a two-dimensional vector including CPU and memory. As different resource types have varying scales, computing resource requirements of SMDs are first normalized. Thus, the Euclidean norm for each resource type is used to represent resource requirements of an SMD.

The second stage of LSAG uses chaotic mapping to initialize the population. M denotes the number of population and D denotes the dimension of each individual. The population (X) is initialized as:

$$\begin{aligned} Z_i &= 4 \times Z_{i-1} \times (1 - Z_{i-1}), i \in [2, 3, ..., M] \\ X_i &= \check{b}_d + (\check{b}_d - \hat{b}_d) \times Z_i, i \in [1, 2, ..., M] \end{aligned} \tag{30}$$

where Z denotes a $M \times D$ zero matrix. $\check{b}_d$ and $\hat{b}_d$ represent lower and upper bounds of each dimension d.

The value of attenuation factor a controls the balance of exploration and exploitation during the optimization process. However, a fixed proportion of exploration and exploitation is difficult to adapt. Therefore, we propose an adaptive attenuation factor that is controlled by γ. t_1 denotes the current iteration number and $\hat{t}_1$ denotes the maximum number of iterations. Therefore, when $\frac{t_1}{\hat{t}_1} < \gamma$, a is updated with (31); otherwise, it is updated with (32).

$$a = -0.1 \times \frac{t_1}{\hat{t}_1} + 0.5 \tag{31}$$

$$a = 2 - (-0.1 \times \frac{t_1}{\hat{t}_1} + 0.5) \tag{32}$$

To improve the exploration ability of LSAG for complex problems, a strategy of Lévy flights is used for enhancing global exploration ability. Lévy flights refer to the random walk with the probability distribution of step length and heavy tail distribution [12]. In this way, gray wolfs have a higher possibility to jump out of local optima. In LSGA, the distance between other wolfs and the third best wolf δ is D_δ, which is decided by that between other wolfs and the two best wolfs (D_α and D_β).

$$\sigma_u = \frac{\Gamma(1+\zeta) sin(\frac{\pi\zeta}{2})^{\frac{1}{\zeta}}}{\Gamma\frac{1+\zeta}{2} \zeta \times 2^{\frac{\zeta-1}{2}}} \tag{33}$$

$$\sigma_v = 1 \tag{34}$$

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

$$D_\delta = \frac{1}{2} \left[\frac{u}{|v|^{-\zeta}} ({X_i}^d - \alpha^d) + \frac{u}{|v|^{-\zeta}} ({X_i}^d - \beta^d) \right] \tag{35}$$

where α^d and β^d denotes the dimension d of the first two best wolfs α and β. Finally, new population X' is updated as:

$${X'_i}^d = \frac{1}{2} \left[\alpha^d - A_1 \times D_\alpha + \beta^d - A_2 \times D_\beta \right] + D_\delta \tag{36}$$

where A_1 and A_2 denote coefficient vectors.

Moreover, LSAG adopts SA's Metropolis acceptance rule when selecting individuals for the next iteration. The Metropolis acceptance rule allows directions worsening objective function values, which is able to jump out of local optima and successfully find global one by setting the cooling rate of temperature [13]. The possibility of acceptance is given as:

$$p = e^{-\frac{\psi}{T}} \tag{37}$$

where ψ is the difference of fitness values before and after the update. T is the initial temperature in SA and p is the possibility of acceptance.

The details of LSAG are shown in algorithm 1. In addition, we discuss the time complexity of LSAG. The computation overhead is mainly brought by the **for** loop, which terminates when the number of iterations reaches $\hat{t}_1$. Thus, the time complexity in each iteration is $\mathcal{O}(DN)$. As a result, the overall time complexity of LSAG is $\mathcal{O}(\hat{t}_1 DN)$.

IV. Experimental Results and Discussion

Experiments are carried out by using data from Google data centers for one day. Parameter setting is shown in Tables I and II. Parameters of LSAG are set as follows. $\gamma = 0.5$, $\zeta = 1.5$, and $T = 1000$. We compare LSAG with three benchmark algorithms including genetic algorithm (GA) [14], genetic learning particle swarm optimization (GLPSO) [15], and grey wolf optimizer (GWO) [16].

Figs. 3 and 4 illustrate the cost and the penalty of LSAG, GA, GWO, and GLPSO in each iteration, respectively when there are 10 SMDs in the system. It is shown that LSAG has the lowest cost (0.048 \$) after 1000 iterations. Moreover,

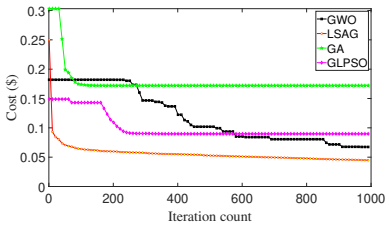

Fig. 3. Cost in each iteration for each algorithm

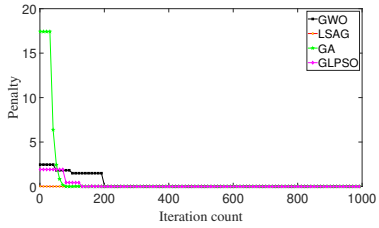

Fig. 4. Penalty for each algorithm

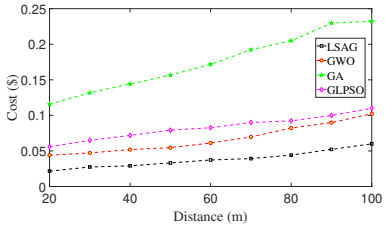

Fig. 5. Cost v.s. distance for each algorithm

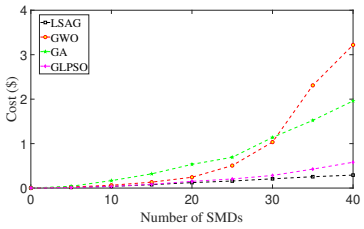

Fig. 6. Cost v.s. SMDs' number for each algorithm

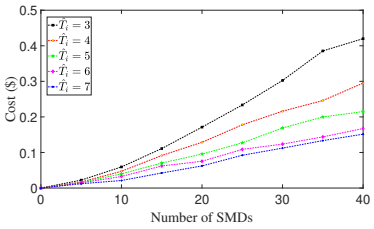

Fig. 7. Cost with $\hat{T}_i$ and SMDs' number for LSAG

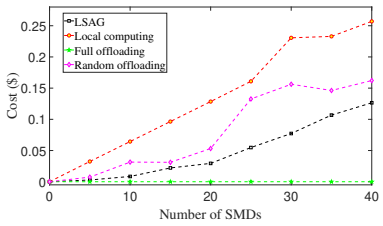

Fig. 8. Cost of SMDs in each iteration

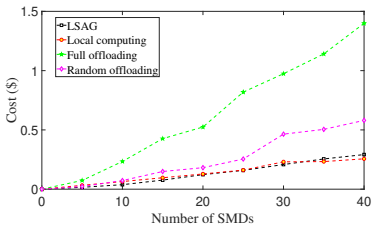

Fig. 9. Cost of system in each iteration

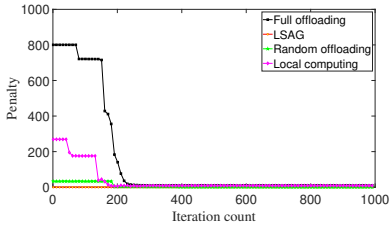

Fig. 10. Penalty of each strategy with 10 SMDs

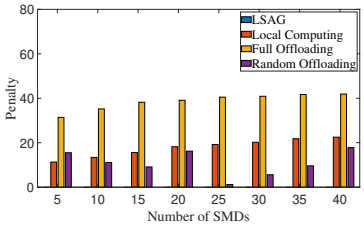

Fig. 11. Penalty of each strategy

TABLE I
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	B_j^U (B_j^D)	P_j^t	λ_{ij}
10	3	[50,100] cycles/bit	[50,100] memory/bit	$[0.5,1.2]\times10^{-26}$	0.1 W	[20,100] m	0.73	1.6×10^{-11}	[5,25] MHz	0.1 W	[10,20] MHz

TABLE II
PARAMETER SETTING-PART II

$\hat{C}_S^j$ ($\hat{M}_S^j$)	$\hat{C}_{CDC}$ ($\hat{M}_{CDC}$)	O_1	O_2	O_3	O_4	$\hat{T}_i$	S_j	e_0	$\hat{E}_j$	$\hat{E}_0$	r_M	r_S	r_C
2×10^{10} Hz (2 GB)	6×10^{10} Hz (6 GB)	1	0.3	1	0.3	[3,7] S	$[0.5,1.2]\times10^{-27}$	1 W/GHz	13 J	45 J	0.04 $/KWH	0.01 $/KWH	0.03 $/KWH

it is shown in Fig. 4 that the penalty of GLPSO and GA cannot be zero at the end. The penalties of LSAG and GWO both decrease to zero at the end of their search processes. This demonstrates that the solutions finally obtained by these algorithms are valid. However, GWO yields a satisfied solution after 200 iterations, which is longer than LSAG. LSAG has a lower penalty at the beginning of iterations and it achieves the penalty of zero after 100 iterations.

Fig. 5 shows the total cost of the system by different distances between SBSs and SMDs. It is shown that the cost of all algorithms increases with the distance because a larger distance between SBSs and SMDs requires more energy consumption in data transmission. Among all the algorithms, LSAG achieves the lowest cost with all different distances. Fig. 6 illustrates the total cost of the system of four algorithms with respect to different numbers of SMDs. It is shown that the cost of LSAG is the smallest when N varies from 0 to 40. In addition, Fig. 7 shows the cost with different values of $\hat{T}_i$ and different numbers of SMDs of LSAG. It is shown that LSAG can yield satisfied solutions under different latency limits.

Furthermore, Figs. 8 and 9 illustrate the cost of SMDs and the total one with respect to different numbers of SMDs. For comparison, we also give the cost of random offloading, local computing, and full offloading. It is shown that LSAG outperforms random and local computing in terms of the

Algorithm 1 LSAG

Input: $\hat{t}_1$, $\check{b}_d$ and $\hat{b}_d$, γ, M, D, T
Output: Final population X

Initialize X with (30)
for each SMD i **do**
  if SMD i can be only served by SBS S_j **then**
    $\mu_{ij} = 1$
  end if
end for
Sort all unallocated SMDs in an ascending order based on the number of required computing resources
Allocate SMD i to SBS S_j with the least available capacity ($\mu_{ij} = 1$)
Calculate the fitness value of all individuals and choose the best one as α, and the suboptimal individual as β
for t_1=1:$\hat{t}_1$ **do**
  for i=1:M **do**
    if $\frac{t_1}{\hat{t}_1}$<γ **then**
      Calculate a with (31)
    else
      Calculate a with (32)
    end if
  end for
  for i=1:M **do**
    for d=1:D **do**
      Update A_1 and A_2 with $2 \times a \times r_1 - a$
      Update C_1 and C_2 with $2 \times r_2 - a$
      $D_\alpha = |C_1 \times \alpha^d - {X_i}^d|$
      $D_\beta = |C_2 \times \beta^d - {X_i}^d|$
      Calculate σ_u with (33)
      Calculate D_δ with (35)
      Update X_i' with (36)
      Calculate the acceptance probability with (37)
    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

cost of SMDs in Fig. 8. Its cost is higher than that of full offloading, which leads to larger transmission latency. Fig. 9 shows that the proposed offloading strategy achieves almost the lowest cost among all the strategies. The penalty of these strategies for 10 SMDs is shown in Fig. 10 and the penalty of each strategy with different numbers of SMDs is shown in Fig. 11. It is shown that all strategies except LSAG do not achieve zero penalties after the iterations.

V. CONCLUSIONS

Nowadays, smart mobile devices (SMDs) play a dominant role in people's lives. However, their limited battery energy and computation resources make it difficult to execute all tasks within the limited time required by users. This work designs the architecture of a cloud-assisted mobile edge computing system for partial computation offloading. Moreover, a constrained cost minimization problem is formulated and solved by a two-stage optimization algorithm called Lévy flights and Simulated Annealing-based Grey wolf optimizer (LSAG). The first stage of LSAG provides the optimal edge selection policy and the second stage jointly optimizes the offloading and resource allocation among SMDs, SBSs, and the cloud for minimizing the total cost of the system. Real data-based experiments prove that LSAG achieves the least cost in fewer iterations than its typical peers. Our next work will extend LSAG by applying it to a more complex environment where locations of SMDs and SBSs move dynamically.

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 of Things Journal*, vol. 9, no. 16, pp. 14521–14529, Aug. 2022.

[2] H. Yuan and M. Zhou, "Profit-Maximized Collaborative Computation Offloading and Resource Allocation in Distributed Cloud and Edge Computing Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1277–1287, Jul. 2021.

[3] J. Bi, H. Yuan, K. Zhang and M. Zhou, "Energy-Minimized Partial Computation Offloading for Delay-Sensitive Applications in Heterogeneous Edge Networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 4, pp. 1941–1954, Oct. 2022.

[4] 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 Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9588–9592, Sept. 2021.

[5] H. Yuan, Q. Hu, J. Bi, J. Lü, J. Zhang and M. Zhou, "Profit-optimized Computation Offloading with Autoencoder-assisted Evolution in Large-scale Mobile Edge Computing," *IEEE Internet of Things Journal*, Feb. 2023.

[6] J. Bi, K. Zhang, H. Yuan and J. Zhang, "Energy-Efficient Computation Offloading for Static and Dynamic Applications in Hybrid Mobile Edge Cloud System," *IEEE Transactions on Sustainable Computing*, Oct. 2022.

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

[8] A. Belgacem, K. Beghdad-Bey and H. Nacer, "Dynamic Resource Allocation Method Based on Symbiotic Organism Search Algorithm in Cloud Computing," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1714–1725, Jul. 2022.

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

[10] M. A. van Wyk, L. Ping and G. Chen, "Multivaluedness in Networks: Shannon's Noisy-Channel Coding Theorem," *IEEE Transactions on Circuits and Systems*, vol. 68, no. 10, pp. 3234–3235, Oct. 2021.

[11] J. Bi, Z. Wang, H. Yuan, J. Zhang, M. Zhou, "Self-adaptive Teaching-learning-based Optimizer with Improved RBF and Sparse Autoencoder for High-dimensional Problems," *Information Sciences*, vol. 630, pp. 463–481, Jun. 2023.

[12] 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 of Things Journal*, vol. 9, no. 9, pp. 6597–6610, May 2022.

[13] X. Zhou, S. Li and Y. Feng, "Quantum Circuit Transformation Based on Simulated Annealing and Heuristic Search," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4683–4694, Dec. 2020.

[14] M. Xu, G. Feng, Y. Ren and X. Zhang, "On Cloud Storage Optimization of Blockchain With a Clustering-Based Genetic Algorithm," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8547–8558, Sept. 2020.

[15] Y. Gong, J. Li, Y. Zhou, Y. Li, H. Chung, Y. Shi and J. Zhang, "Genetic Learning Particle Swarm Optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[16] S. Mirjalili, S. M. Mirjalili, A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, no. 10, pp. 46–61, Mar. 2014.