Cost-Minimized Partial Computation Offloading in Heterogeneous Edge-Cloud Systems

Ziqi Wang
Faculty of Information Technology
Beijing Laboratory of Smart Environmental Protection
Beijing University of Technology
Beijing, 100124, China
ziqi_wang@emails.bjut.edu.cn

Jing Bi
Faculty of Information Technology
Beijing Laboratory of Smart Environmental Protection
Beijing University of Technology
Beijing, 100124, China
bijing@bjut.edu.cn

MengChu Zhou
School of Information and Electronic Engineering
Zhejiang Gongshang University
Hangzhou 310018, China
mengchu@gmail.com

2024 7th International Symposium on Autonomous Systems (ISAS) | 979-8-3503-6317-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISAS61044.2024.10552596

***Abstract*—Nowadays, Internet of Things Devices (IOTDs) support numerous applications that require extensive computational resources and are sensitive to delays. Nevertheless, IOTDs are constrained by limited computational power and battery life, preventing them from processing all tasks in real-time. Computation offloading provides a solution to these problems where IOTDs can offload part of their tasks to edge servers for execution. Small Base Stations (SBSs) are located closer to IOTDs, which act as edge servers. However, SBSs have limited computing resources compared with a Cloud Data Center (CDC). Therefore, a heterogeneous edge-cloud system is often deployed in urban areas for computation offloading. In addition, attaining the lowest cost in such a system while satisfying the delay requirements of tasks presents a significant challenge. In this work, a computation offloading strategy aimed at minimizing the overall system cost is proposed. Initially, an optimization problem with real-world constraints is defined, leveraging the hybrid architecture as its basis. Then, a novel swarm optimization algorithm named Grey wolf optimization embedded with Simulated annealing and Genetic learning (GSG) is proposed to solve this optimization problem. GSG optimizes resource allocation and task offloading among IOTDs, SBSs, and CDC. Simulation experiments involving real-life tasks demonstrate that GSG achieves a significantly lower system cost than its existing peers.**

***Index Terms*—Internet of things, edge-cloud systems, swarm intelligent algorithms.**

I. INTRODUCTION

The integration of computer devices into commonplace objects *via* the Internet provides a paradigm known as Internet of Things (IOT). It creates a network that enables seamless data transmission and reception through that Internet. IOT not only facilitates the collection of information but also provides opportunities to optimize the cost through utilizing the acquired data [1]–[3]. In recent years, the development of IOT Devices (IOTDs) has experienced a substantial increase. They include smart mobile devices [4], intelligent vehicles [5], and manufacturing robots [6]. These IOTDs enable the associations of physical and digital domains by utilizing sensors and actuators. Nonetheless, the data collected by them requires storage and processing for effective utilization. Moreover, certain tasks demand substantial computing resources and energy to address the extensive volume of information they entail. However, executing these applications locally becomes challenging given their limited computing resources and battery energy [7]. In addition, excessive energy consumption diminishes battery performance and shortens their lifespan.

Computation offloading provides a solution to solve this problem [8]. It refers to assigning computation-intensive tasks to an edge server that has sufficient computation resources for processing and fetching the completed computation results. It can be categorized into two distinct types [9], *i.e.*, full and partial offloading. The former offloads all tasks to edge servers for processing and the latter offloads a part of tasks only. Small Base Stations (SBSs) can act as edge servers for processing offloaded tasks. They offer flexibility for deployment in densely populated areas and furnish IOTDs with nearby access to high-quality services [10]. However, computation resources available at the edge are not as abundant as those in a Cloud Data Center (CDC). Thus, CDC deals with excessive tasks that SBSs cannot process in an edge-cloud system.

The additional task offloading process from IOTDs to SBSs/CDC unavoidably causes communication latency. In addition, as many IOTDs offload tasks to various SBSs/CDC, the challenge arises of efficiently allocating computational resources in each system component. The edge-cloud system's total cost comprises those of IOTDs, SBSs, and CDC. Therefore, its minimization is important.

Motivated by the aforementioned analysis, this work designs a computation offloading method in an edge-cloud system to minimize the system's total cost. First, an edge-cloud architecture comprising multiple IOTDs, SBSs, and CDC is

This work was supported by the Beijing Natural Science Foundation under Grants L233005 and 4232049, the National Natural Science Foundation of China under Grants 62173013 and 62073005, and the Fundamental Research Funds for the Central Universities under Grant YWF-23-03-QB-015.

constructed. Then, considering the properties of each device and user requirements, a cost-minimization problem is formulated. Finally, a novel swarm intelligent algorithm named Grey wolf optimization embedded with Simulated annealing and Genetic learning (GSG) is designed to solve it. It incorporates the Metropolis acceptance rule of Simulated Annealing (SA) and genetic learning operations into Grey Wolf Optimization (GWO). GSG optimizes energy consumption and resource allocation while meeting the latency requirement of IOTDs. Experimental results demonstrate that GSG achieves economic efficiency in the edge-cloud system.

II. Problem Presentation

An architecture of an edge-cloud system is illustrated in Fig. 1 to investigate the cost minimization problem. Each IOTD in this system is connected to one Access Point (AP), and each IOTD establishes a connection with an associated AP. It establishes uplink and downlink communication channels among IOTDs, SBSs, and/or CDC. In addition, APs possess limited computing resources and do not operate applications independently. Moreover, two types of tasks are considered in this work, *i.e.*, static offloading and dynamic offloading [11]. As illustrated in Fig. 2, in the static offloading manner, the entire task is offloaded to an SBS or CDC for computing. Conversely, task partitioning occurs at runtime in dynamic offloading, and its determination relies on the accessibility of networks and the current state of SBSs/CDC. To represent task types of IOTD i associated with AP j, a binary variable μ_{ij} is introduced. If a task of IOTD i supports dynamic offloading, μ_{ij}=0; otherwise, μ_{ij}=1 if a task of IOTD i only supports static offloading. In the case of dynamic offloading, tasks can be offloaded to SBSs/CDC in any proportion. σ_{ij} denotes the proportion of a task of IOTD i offloaded to the edge through AP j and it can not exceed one, *i.e.*,

$$\sum_{j=1}^{J} \sigma_{ij} \leq 1, i \in [1, 2, ..., N] \tag{1}$$

where J denotes the number of APs, and N denotes that of IOTDs.

If a task of IOTD i is offloaded to SBS k through AP j, α_{ij}^{k}=1. Similarly, if a task of IOTD i is offloaded to CDC through AP j, α_{ij}^{c}=1. It is worth noting that a task can only be offloaded to at most one SBS or CDC at any given time, *i.e.*,

$$\sum_{j=1}^{J}\sum_{k=1}^{K} \alpha_{ij}^{k} + \sum_{j=1}^{J} \alpha_{ij}^{c} \leq 1, i \in [1, 2, ..., N] \tag{2}$$

where K denotes the number of SBSs.

In the case of static offloading, the task of IOTD i must be offloaded to one SBS or CDC. Thus, when μ_{ij}=1, we have:

$$\sum_{j=1}^{J}\sum_{k=1}^{K} \alpha_{ij}^{k} + \sum_{j=1}^{J} \alpha_{ij}^{c} = 1, i \in [1, 2, ..., N] \tag{3}$$

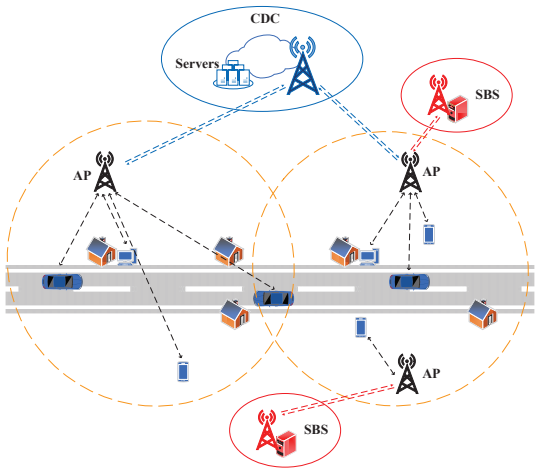

Fig. 1. Architecture of the edge-cloud system.

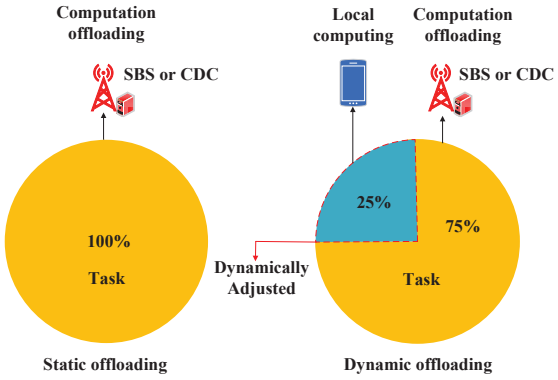

Fig. 2. Static offloading and dynamic offloading.

A. Latency Model

The latency of this edge-cloud system consists of three parts related to IOTDs, SBSs and CDC.

1) Latency of IOTDs: Let $\dot{T}_{ij}$ denote the local execution latency of IOTD i's tasks associated with AP j. Then,

$$\dot{T}_{ij} = (1 - \sigma_{ij}) \frac{\theta_i}{f_i} \tag{4}$$

where θ_i represents the quantity of computing resources needed by IOTD i and f_i denotes its CPU running speed.

Let $\ddot{T}_{ij}$ denote the latency of transmission between IOTD i and AP j. Based on [12], we have:

$$\ddot{T}_{ij} = \sigma_{ij} \frac{I_{ij}}{\hat{R}_{ij}} + \sigma_{ij} \frac{O_{ij}}{\check{R}_{ij}} \tag{5}$$

where the first term is the uplink transmission latency and the second one is the downlink transmission latency. I_{ij} (O_{ij}) denotes the bit size of the input (output) data of IOTD i transmitted *via* AP j. $\hat{R}_{ij}$ ($\check{R}_{ij}$) represents the transmission rates for uplink (downlink) channels between IOTD i and its

associated AP j. Based on the Shannon's theorem, $\hat{R}_{ij}$ and $\check{R}_{ij}$ are obtained as:

$$\hat{R}_{ij}=\lambda_{ij}\hat{B}_j\log_2(1+\frac{\hat{P}_{ij}(d_{ij})^{-v}|o_1|^2}{{\omega_0}^2}) \tag{6}$$

$$\check{R}_{ij}=\lambda_{ij}\check{B}_j\log_2(1+\frac{\check{P}_{ij}(d_{ij})^{-v}|o_2|^2}{{\omega_0}^2}) \tag{7}$$

where λ_{ij} is the bandwidth proportion of uplink and downlink channels linking IOTD i with AP j. $\hat{B}_j$ ($\check{B}_j$) is the bandwidth of uplink (downlink) channels for AP j. $\hat{P}_{ij}$ ($\check{P}_{ij}$) represents the transmission power of uplink (downlink) channels linking IOTD i with AP j. d_{ij} denotes the distance between IOTD i and AP j. o_1 (o_2) denotes fading coefficients of uplink (downlink) channels between each IOTD and each AP. ω_0 is the noise parameter and v denotes a coefficient.

The aggregate allocation of bandwidth proportions across all connected IOTDs to AP j equals one, *i.e.*,

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\lambda_{ij}\alpha_{ij}^k+\sum_{i=1}^{N}\lambda_{ij}\alpha_{ij}^c=1, j\in[1,2,...,J] \tag{8}$$

The time consumption related to IOTDs is calculated as:

$$\overline{T}_{ij}=\dot{T}_{ij}+\ddot{T}_{ij} \tag{9}$$

2) Latency of SBSs: $\dot{T}_{ij}^k$ represents the execution latency of SBS k when processing the offloaded tasks from IOTD i through AP j. f_k denotes the CPU processing speed of SBS k. We have:

$$\dot{T}_{ij}^k=\frac{\alpha_{ij}^k\sigma_{ij}\theta_i}{f_k} \tag{10}$$

The total latency of SBS k is:

$$T_{ij}^k=\dot{T}_{ij}^k+\ddot{T}_{ij}^k \tag{11}$$

where $\ddot{T}_{ij}^k$ denotes the transmission time between SBS k and AP j associated with IOTD i.

3) Time modeling in CDC: $\dot{T}_{ij}^c$ denotes the execution time of CDC when processing the offloaded tasks from IOTD i through AP j, and f_c denotes the CPU processing speed of CDC. Therefore, similar to SBS, we have:

$$\dot{T}_{ij}^c=\frac{\alpha_{ij}^c\sigma_{ij}\theta_i}{f_c} \tag{12}$$

The total time related to CDC is $\widetilde{T}_{ij}^c$, and it is obtained as:

$$\widetilde{T}_{ij}^c=\dot{T}_{ij}^c+\ddot{T}_{ij}^c \tag{13}$$

where $\ddot{T}_{ij}^c$ denotes the transmission time between CDC and AP j associated with IOTD i.

Finally, the total time of completing tasks in IOTD i ($\mathcal{T}_i$) is the maximum value of local execution and offloaded one, *i.e.*,

$$\mathcal{T}_i=max\left(\sum_{j=1}^{J}\overline{T}_{ij},\sum_{j=1}^{J}\sum_{k=1}^{K}T_{ij}^k,\sum_{j=1}^{J}\widetilde{T}_{ij}^c\right) \tag{14}$$

Moreover, $\mathcal{T}_i$ has to be less than the time limit defined by the user ($\hat{\mathcal{T}}_i$), *i.e.*,

$$\mathcal{T}_i\le\hat{\mathcal{T}}_i \tag{15}$$

B. Total Cost Modeling

The total system cost correlates closely with its energy consumption, which consists of three parts related to IOTDs, SBSs, and CDC.

1) Energy consumption related to IOTDs: $\dot{E}_{ij}$ denotes the local execution energy in IOTD i associated with AP j, *i.e.*,

$$\dot{E}_{ij}=(1-\sigma_{ij})\theta_i S_i(f_i)^2 \tag{16}$$

where S_i is decided by the chip architecture of IOTD i. Moreover, f_i must not exceed its maximum limit ($\hat{f}_i$), *i.e.*,

$$f_i\le\hat{f}_i \tag{17}$$

$\ddot{E}_{ij}$ denotes the transmission energy between IOTD i and AP j and it is obtained as:

$$\ddot{E}_{ij}=\hat{P}_{ij}\hat{T}_{ij}+\check{P}_{ij}\check{T}_{ij} \tag{18}$$

where $\hat{T}_{ij}$ and $\check{T}_{ij}$ denote the time of data transmission between IOTD i to AP j, and that between AP j to IOTD i, respectively.

We have the total energy consumption related to IOTD i, *i.e.*,

$$\overline{E}_{ij}=\dot{E}_{ij}+\ddot{E}_{ij} \tag{19}$$

2) Energy consumption modeling of SBSs: Let $\dot{E}_{ij}^k$ denote the execution energy of processing IOTD i's offloaded tasks in SBS k through AP j. It is calculated as:

$$\dot{E}_{ij}^k=\alpha_{ij}^k\sigma_{ij}\theta_i S_k(f_k)^2 \tag{20}$$

where S_k is decided by the chip architecture of SBS k. In addition, the CPU processing speed in SBS must not exceed its maximum limit ($\hat{f}_k$), *i.e.*,

$$\sum_{i=1}^{N}\sum_{j=1}^{J}\alpha_{ij}^k f_k\le\hat{f}_k \tag{21}$$

$\ddot{E}_{ij}^k$ denotes the transmission energy between SBS k and AP j associated with IOTD i, *i.e.*,

$$\ddot{E}_{ij}^k=2\alpha_{ij}^k\overline{P}_{jk}d_{jk} \tag{22}$$

where $\overline{P}_{jk}$ denotes the power of data transmission between SBS k and AP j. d_{jk} represents the distance between AP j and SBS k.

Therefore, the total energy consumption of SBS k is:

$$E_{ij}^k=\dot{E}_{ij}^k+\ddot{E}_{ij}^k \tag{23}$$

3) Energy consumption modeling of CDC: $\dot{E}_{ij}^c$ denotes the execution energy of processing IOTD i's offloaded tasks in CDC through AP j, *i.e.*,

$$\dot{E}_{ij}^c=\alpha_{ij}^c\sigma_{ij}\theta_i e_c \tag{24}$$

where e_c represents the energy consumption associated with each CPU cycle in the CDC.

$\ddot{E}_{ij}^c$ denotes the transmission energy between CDC and AP j associated with IOTD i, which is given as:

$$\ddot{E}_{ij}^c=2\alpha_{ij}^k\widetilde{P}_{jc}d_{jc} \tag{25}$$

where $\widetilde{P}_{jc}$ denotes the transmission power between AP j and CDC, and d_{jc} represents the distance between AP j and CDC.

The total energy consumption of the CDC is $\widetilde{E}^c_{ij}$, *i.e.*,

$$\widetilde{E}^c_{ij}=\dot{E}^c_{ij}+\ddot{E}^c_{ij} \tag{26}$$

F represents the total cost of the edge-cloud system, consisting of three distinct components: the cost of IOTDs (F_I), SBSs (F_S), and CDC (F_C). Finally, we have:

$$F=F_I+F_S+F_C \tag{27}$$

$$F_I=\sum_{i=1}^{N}\sum_{j=1}^{J} r_i\overline{E}_{ij} \tag{28}$$

$$F_S=\sum_{i=1}^{N}\sum_{j=1}^{J}\sum_{k=1}^{K} r_k\alpha^k_{ij}E^k_{ij} \tag{29}$$

$$F_C=\sum_{i=1}^{N}\sum_{j=1}^{J} r_c\alpha^c_{ij}\widetilde{E}^c_{ij} \tag{30}$$

where r_i, r_k, and r_c denote prices ($/KWH) of energy in IOTD i, SBS k, and CDC, respectively.

C. Constrained Optimization Problem

Thus, decision variables include α^k_{ij}, σ_{ij}, α^c_{ij}, λ_{ij}, f_i, f_k. We aim to optimize F, *i.e.*,

$$\underset{\zeta}{\textbf{Min}}\ F$$

where ζ represents a vector containing decision variables. Moreover, they are subject to (1), (2), (3), (8), (15), (17) and (21). To address the aforementioned constraints, we employ a penalty function method [13] to convert all constraints into penalties, thus transforming the problem into an unconstrained optimization one.

III. Grey wolf optimization embedded with Simulated annealing and Genetic learning (GSG)

GSG is proposed to solve the above optimization problem (F). First, chaotic mapping [14] is utilized to initialize the population, ensuring comprehensive coverage of the search space. M represents the population size, and D represents the number of decision variables. Then, the population (X) is initialized as:

$$\begin{aligned} Z_i&=4\times Z_{i-1}\times(1-Z_{i-1})\\ X_i&=\check{b}_d+(\check{b}_d-\hat{b}_d)\times Z_i \end{aligned} \tag{31}$$

where Z represents a zero matrix with $M\times D$ elements. $\check{b}_d$ and $\hat{b}_d$ denote lower and upper limits of each dimension d.

In the searching process, the rest of grey wolves (ω) search under the guidance of the first three best wolves (α, β and δ). D^α, D^β and D^δ represent the distance between ω and α, and that between β and δ, respectively. They are updated as:

$$D^\alpha=|C_1\times\alpha_d-X^i_d| \tag{32}$$

$$D^\beta=|C_2\times\beta_d-X^i_d| \tag{33}$$

$$D^\delta=|C_3\times\delta_d-X^i_d| \tag{34}$$

where C_1, C_2 and C_3 denote updating coefficients. α_d, β_d and δ_d denote the elements of dimension d of α, β and δ. X^i_d denotes the element of dimension d of individual i.

Moreover, X_1, X_2 and X_3 denote the positions of ω that need to be adjusted to reflect the influence of α, β and δ, respectively. They are updated as:

$$X_1=|\alpha_d-A_1\times D^\alpha| \tag{35}$$

$$X_2=|\beta_d-A_2\times D^\beta| \tag{36}$$

$$X_3=|\delta_d-A_3\times D^\delta| \tag{37}$$

where A_1, A_2 and A_3 denote updating coefficients.

Finally, the positions in the next iteration are the average values of X_1, X_2 and X_3, *i.e.*,

$$X^i_d(t+1)=\frac{X_1+X_2+X_3}{3} \tag{38}$$

where $X^i_d(t+1)$ denotes the element of dimension d of individual i in iteration $t+1$.

A crossover strategy from the Genetic Algorithm (GA) is integrated to sustain the diversity of the population. Specifically, p_c controls the possibility of a crossover. If the updated individual ($X^i(t+1)$) outperforms a random one ($X^k(t+1)$) of the population, each dimension of $X^i(t+1)$ is updated with (39); otherwise, it is adjusted with (40).

$$X^i_d(t+1)=X^i_d(t+1) \tag{39}$$

$$X^i_d(t+1)=r\cdot\alpha_d(t+1)+(1-r_1-r_2)\cdot X^k_d(t+1), k\in[1,...,M] \tag{40}$$

where $\alpha_d(t+1)$ denotes the element of dimension d of the best individual in iteration $t+1$. $X^k_d(t+1)$ denotes the element of dimension d of individual k in iteration $t+1$. r_1 and r_2 denote two random values in [-1,1].

GSG adopts a mutation operation to avoid its trapping into local optima. Specifically, an individual undergoes mutation, and the probability of mutation is governed by p_m. If the individual is mutated, the value of a random dimension of the individual is reinitialized in the decision space. Finally, GSG employs the Metropolis acceptance rule to determine the selection of individuals for the next iteration. It facilitates escaping from local optima and effectively locating global optima by appropriately setting the initial temperature (T). The acceptance probability (p_a) is defined as:

$$p_a=e^{\frac{T}{\psi}} \tag{41}$$

where ψ denotes the difference in fitness values before and after each iteration.

Finally, the process of GSG is given in Algorithm 1.

TABLE I
PARAMETER SETTING

N	K	J	d_{ij}	$\hat{f}_i$	$\check{f}_k$	$\hat{P}_{ij}$ ($\check{P}_{ij}$)	$\overline{P}_{jk}$ ($\tilde{P}_{jc}$)	ω_0	S_i	S_k	$\hat{B}_j$ ($\check{B}_j$)	r_i	r_k	r_c
10	2	3	[0.2,2] km	[1,10]$\times 10^5$ Hz	6×10^{10} Hz	20 dBm	15 dBm	-56 dBm	10^{-23}	10^{-31}	5 MHz	[0.01,0.07] \$/KWH	0.01 \$/KWH	0.03 \$/KWH

Algorithm 1 GSG

Input: $\hat{t}$, $\check{b}_d$, $\hat{b}_d$, T, p_c, and p_m
Output: X

Initialize X with (31)
for t=1:$\hat{t}$ **do**
  Choose α, β and δ
  Update a with $2-\frac{2t}{\hat{t}}$
  for i=1:M **do**
    for d=1:D **do**
      Update A_1, A_2 and A_3 with $2\times a\times r_1-a$
      Update C_1, C_2 and C_3 with $2\times r_2-a$
      Calculate D^{α} with (32)
      Update X_1 with (35)
      Calculate D^{β} with (33)
      Update X_2 with (36)
      Calculate D^{δ} with (34)
      Update X_3 with (37)
      Update $X_d^i(t+1)$ with (38)
    end for
  end for
  for i=1:M **do**
    if $r<p_c$ **then**
      for d=1:D **do**
        Select k in [1,M]
        if $f(X^i(t+1)) < f(X^k(t+1))$ **then**
          Update $X_d^i(t+1)$ with (39)
        else
          Update $X_d^i(t+1)$ with (40)
        end if
      end for
    end if
    if $r<p_m$ **then**
      Select d as a random number in [1,D]
      $X_d^i(t+1)=\check{b}_d+(\hat{b}_d-\check{b}_d)$
    end if
    Calculate the p_a with (41)
    Adopt Metropolis acceptance rule to update the individual
  end for
end for

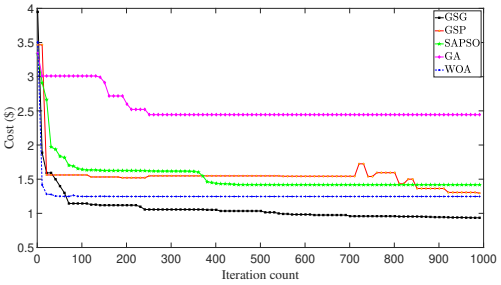

Fig. 3. Cost in each iteration for each algorithm.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

Experiments are carried out to simulate the real situation of an edge-cloud system. The parameter setting of the system is shown in Table I. The parameters of GSG are set as: $p_c = 0.5$,

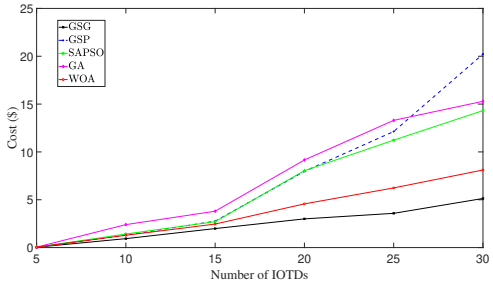

Fig. 4. Cost v.s. d for each algorithm.

TABLE II
PENALTY OF EACH METHOD FOR DIFFERENT NUMBERS OF IOTDS

Methods \ N	5	10	15	20	25	30
GSG	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
GSP	**0.00**	**0.00**	**0.00**	**0.00**	2.11	2.72
SAPSO	**0.00**	0.01	0.04	0.21	0.52	0.36
GA	**0.00**	**0.00**	**0.00**	3.26	1884.23	9978.36
WOA	**0.00**	**0.00**	**0.00**	**0.00**	4.82	20.11

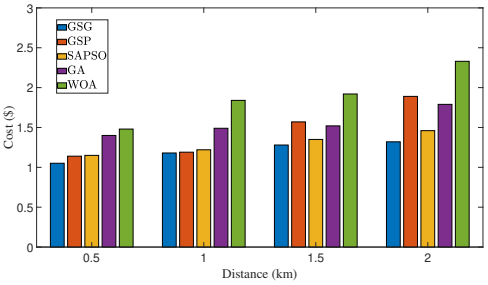

Fig. 5. Cost v.s. d_{ij} for each algorithm.

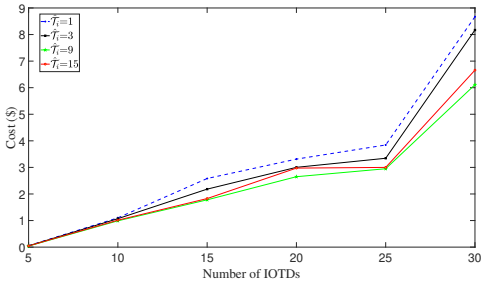

Fig. 6. Cost v.s. $\hat{\mathcal{T}}_i$ for each algorithm.

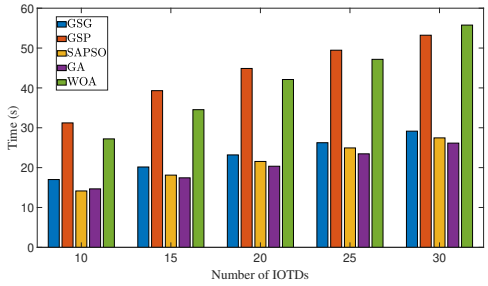

Fig. 7. Time of each algorithm for 1000 iterations.

$p_m = 0.01$, and $T = 10^{10}$. GSG is compared with four state-of-the-art peers, including GA [15], Genetic SA-based Particle swarm optimization (GSP) [16], SA-based Particle Swarm Optimization (SAPSO) [17], and Whale Optimization Algorithm (WOA) [18].

Fig. 3 shows the system's cost of GSG, GSP, SAPSO, GA, and WOA. The number of IOTDs is set to 10. The results demonstrate that GSG achieves the lowest cost (0.92 \$) after its iterations. Fig. 4 illustrates the system's total cost of five algorithms given different numbers of IOTDs. It is demonstrated that GSG achieves the lowest cost when N varies from 0 to 30. Furthermore, Table II presents the penalty of each algorithm for different values of N. The results demonstrate that the penalty of GSG consistently remains zero across varying values of N, affirming that GSG consistently finds valid solutions. However, its peers cannot find solutions that meet all constraints when N is large.

The total system cost for various distances between APs and IOTDs is shown in Fig. 5. It demonstrates that the system cost increases for all algorithms as d_{ij} grows, with GSG consistently achieving the lowest cost across all d_{ij}. Fig. 6 shows the cost with different values of $\hat{\mathcal{T}}_i$ and the number of IOTDs for GSG. It demonstrates that GSG obtains satisfactory solutions under different latency constraints. Finally, Fig. 7 shows the time consumption of each method with 1000 iterations given 10 IOTDs in the system. It shows that the time consumption of GSG is slightly longer than GA and SAPSO. However, both of them cannot generate valid solutions for large N.

V. CONCLUSIONS

The wide use of Internet of Things Devices (IOTD) greatly facilitates people's lives due to their easy access, convenient monitoring, and fast speed. However, they face constraints such as limited battery energy and computational resources, hindering their ability to process all tasks demanded by users. This work constructs an edge-cloud architecture to facilitate computation offloading for IOTDs. A constrained cost minimization problem is defined and resolved utilizing a novel swarm intelligent algorithm named Grey wolf optimization embedded with Simulated annealing and Genetic learning (GSG). It optimizes resource allocation and task offloading among IOTDs, small base stations, and a cloud data center to minimize the overall system cost. Experiments based on real-life data are conducted and the results demonstrate that GSG attains the lowest cost in fewer iterations than its peers. Our future work intends to extend GSG by incorporating the optimal edge selection strategies and more efficient search mechanisms with deep learning.

REFERENCES

[1] A. Tripathi, K. Sharma, M. Bala, *et al.,* "A Parallel Military-Dog-Based Algorithm for Clustering Big Data in Cognitive Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2134–2142, Mar. 2021.

[2] J. Hwang, L. Nkenyereye, N. Sung, *et al.,* "IoT Service Slicing and Task Offloading for Edge Computing," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11526–11547, Jul. 2021.

[3] H. Yuan, Q. Hu, J. Bi, *et al.,* "Profit-optimized Computation Offloading with Autoencoder-assisted Evolution in Large-scale Mobile Edge Computing," *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2024.3354251.

[4] M. Ghahramani, M. Zhou, and G. Wang, "Urban Sensing based on Mobile Phone Data: Approaches, Applications, and Challenges," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 627–637, May 2020.

[5] Z. Wang, X. Li, J. Wang, *et al.,* "Energy-minimized Partial Computation Offloading in Cloud-assisted Vehicular Edge Computing Systems," *2023 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pp. 1–6, Marseille, France, 2023.

[6] F. Chervinskii, S. Zobov, A. Rybnikov, *et al.,* "Auto-Assembly: A Framework for Automated Robotic Assembly Directly from CAD," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11294–11300, London, United Kingdom, 2023.

[7] Z. Sharif, L. T. Jung, I. Razzak, *et al.,* "Adaptive and Priority-Based Resource Allocation for Efficient Resources Utilization in Mobile-Edge Computing," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3079–3093, Feb. 2023.

[8] W. Lu and X. Zhang, "Computation Offloading for Partitionable Applications in Dense Networks: An Evolutionary Game Approach," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 20985–20996, Nov. 2022.

[9] C. Ren, G. Zhang, X. Gu, *et al.,* "Computing Offloading in Vehicular Edge Computing Networks: Full or Partial Offloading?," *2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pp. 693–698, Chongqing, China, 2022.

[10] Z. Wang and J. Zheng, "Performance Analysis of Location-based Base Station Cooperation for Cellular-Connected UAV Networks," *IEEE Transactions on Vehicular Technology*, pp. 1–14, May 2023.

[11] H. Yuan, J. Bi, Z. Wang, *et al.,* "Partial and Cost-minimized Computation Offloading in Hybrid Edge and Cloud Systems," *Expert Systems with Applications*, vol. 250, pp. 1–13, Sept. 2024.

[12] J. Bi, Z. Wang, H. Yuan, *et al.,* "Cost-Minimized Partial Computation Offloading in Cloud-Assisted Mobile Edge Computing Systems," *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 5052–5057, Honolulu, Oahu, HI, USA, 2023.

[13] J. Bi, Z. Wang, H. Yuan, *et al.,* "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.

[14] J. Cui, J. Yu, H. Zhong, *et al.,* "Chaotic Map-Based Authentication Scheme Using Physical Unclonable Function for Internet of Autonomous Vehicle," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 3, pp. 3167-3181, Mar. 2023.

[15] C. Peng, X. Wu, W. Yuan, *et al.,* "MGRFE: Multilayer Recursive Feature Elimination Based on an Embedded Genetic Algorithm for Cancer Classification," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 2, pp. 621–632, Mar. 2021.

[16] J. Bi, Z. Wang, H. Yuan, *et al.,* "Cost-Minimized Computation Offloading and User Association in Hybrid Cloud and Edge Computing," *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2024.3354348.

[17] F. Javidrad and M. Nazari, "A New Hybrid Particle Swarm and Simulated Annealing Stochastic Optimization Method," *Appl. Soft Comput.* vol. 60, pp. 2866-2880, Nov. 2017.

[18] Q. Pham, S. Mirjalili, N. Kumar, *et al.,* "Whale Optimization Algorithm With Applications to Resource Allocation in Wireless Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4285–4297, Apr. 2020.