Mathematical Biosciences and Engineering

MBE, 21(6): 6336–6358.
DOI: 10.3934/mbe.2024276
Received: 02 January 2024
Revised: 18 March 2024
Accepted: 16 May 2024
Published: 17 June 2024

http://www.aimspress.com/journal/MBE

Research article

Research on MEC computing offload strategy for joint optimization of delay and energy consumption

Mingchang Ni[1], Guo Zhang[1,*], Qi Yang[2,*] and Liqiong Yin[2]

[1] Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650504, China
[2] Kunming Iron & Steel Holding Co., Ltd. Kunming 650302, China

* **Correspondence:** Email: 12309021@kust.edu.cn, kisco_yq@sina.cn.

Abstract: The decision-making process for computational offloading is a critical aspect of mobile edge computing, and various offloading decision strategies are strongly linked to the calculated latency and energy consumption of the mobile edge computing system. This paper proposes an offloading scheme based on an enhanced sine-cosine optimization algorithm (SCAGA) designed for the "edge-end" architecture scenario within edge computing. The research presented in this paper covers the following aspects: (1) Establishment of computational resource allocation models and computational cost models for edge computing scenarios; (2) Introduction of an enhanced sine and cosine optimization algorithm built upon the principles of Levy flight strategy sine and cosine optimization algorithms, incorporating concepts from roulette wheel selection and gene mutation commonly found in genetic algorithms; (3) Execution of simulation experiments to evaluate the SCAGA-based offloading scheme, demonstrating its ability to effectively reduce system latency and optimize offloading utility. Comparative experiments also highlight improvements in system latency, mobile user energy consumption, and offloading utility when compared to alternative offloading schemes.

Keywords: mobil edge computing; computing offloading; computational resource allocation; sine and cosine optimization algorithms

1. Introduction

Owing to constraints related to battery power and computing resources, intelligent mobile terminal equipment (IMTE) faces challenges in handling intricate problems, including massive computing tasks and delay-sensitive tasks arising in the context of intelligent manufacturing processes.

In intelligent manufacturing environments, IMTE is generally used to collect data from sensors, extract meaningful information, and make corresponding decisions based on changes in the data. However, the emergence of new technologies requires IMTE to have high performance and strict real-time response, such as object detection based on computer vision, automatic route planning, perception, and end-to-end decision-making and so on. IMTE is difficult to support the growing amount of computing due to limitations in energy supply and unit size. With the emergence of Mobile Edge Computing (MEC) technology, IMTE can offload computing tasks to edge servers, which to some extent improves the computational efficiency of IMTE.

Computing offloading decision-making, as one of the primary research focuses in MEC, has continuously attracted the attention of numerous scholars. Considering the allocation of computing resources, Huang et al. [1] proposed a computing offloading algorithm based on a greedy approach, which effectively reduced system costs. Tran et al. [2] used convex optimization and quasi convex optimization to solve resource allocation problems, and proposed a new heuristic algorithm to solve the task offloading subproblem, whose solution is close to the optimal solution. Tang et al. [3] proposed a task offloading and resource allocation algorithm based on Lyapunov optimization theory, which maximizes the average time benefit of MEC systems. The algorithm exhibits good performance in terms of latency, reliability, and system benefits. Wu et al. [4] proposed a distributed offloading algorithm based on game theory, which minimizes the system cost of energy consumption and delay for users, and proves that the algorithm can obtain a global optimal solution. Hu et al. [5] proposed a task offloading algorithm based on minority game theory for task offloading in heterogeneous environments with incomplete information, and proved that the proposed offloading algorithm can obtain suboptimal solutions close to the optimal solution. Zheng et al. [6] proposed a computational offloading algorithm and resource allocation strategy based on the dual auction algorithm, which can effectively reduce system losses. Based on the non-cooperative game interaction between wireless characteristics and mobile users, Zang et al. [7] developed an iterative mechanism to jointly determine the computational offloading scheme. The offloading algorithm demonstrates fast convergence speed and excellent energy efficiency performance.

With the continuous advancement of deep learning, intelligent computing offloading solutions based on online learning have emerged as a prominent research focus in recent years. Meng et al. [8] proposed an optimal offloading strategy for random task generation using enhanced Q-learning and deep learning, resulting in reduced energy consumption and a 38.1% reduction in delay weighted sum. Qiu et al. [9] introduced a novel online computation offloading algorithm based on model-free deep reinforcement learning; experimental results demonstrated the algorithm's rapid convergence and its ability to obtain favorable suboptimal solutions. Zhu et al. [10] proposed an enhanced particle swarm optimization algorithm that incorporates a genetic algorithm, surpassing both genetic algorithm and particle swarm algorithm in terms of latency and energy consumption. Cong et al. [11] proposed a task offloading strategy with two-stage heuristic characteristics suitable for vehicular networks, the algorithm achieves optimal resource allocation during the offloading process and can

improve task offloading efficiency. Yang et al. [12] designed a computational offloading method (HFOA) based on a hybrid fruit fly algorithm, and its experiments showed that the algorithm has certain improvements in system computing latency, computing energy consumption, convergence, and other aspects.

Based on a comprehensive analysis of the above literature, computing offloading decision has attracted extensive attention and sustained research in the field of MEC. Various algorithms and strategies have been proposed to solve problems such as resource allocation, energy consumption, and latency, and have achieved significant performance improvements. In addition, computing offloading scheme based on swarm intelligence optimization has also become one of the main research directions in the field of edge computing, by searching different regions in the space and utilizing multiple random and adaptive variables, these algorithms effectively avoid getting stuck in local optima and converge towards global optima.

In recent years, the sine and cosine optimization algorithm, as a novel intelligent optimization algorithm, has found successful applications across various domains. Because the sine and cosine optimization algorithm is a stochastic optimization approach with high adaptability, it can readily address optimization challenges in diverse areas such as production scheduling, path planning, and complex problem-solving.

A hybrid meta-heuristic algorithm is presented in [13], combining the salp group algorithm and the sine and cosine algorithm (SSCA) to enhance convergence speed and attain optimal accuracy in practical engineering applications. Reference [14] introduces an enhanced sine and cosine algorithm known as Hierarchical Multi-leader SCA (HMLSCA), which addresses the balance issue within SCA by employing an efficient hierarchical multi-leader search mechanism. The results show that HMLSCA outperforms other algorithms in various tests and has achieved remarkable achievements in support vector machine parameters and COVID-19 diagnosis. In [15], C-CHOA-SC algorithm is proposed, which combines chimpanzee optimization algorithm and sine and cosine algorithm, aiming at solving complex multi-objective optimization problems. The analysis shows that the performance of the proposed algorithm is obviously better than other methods, and the overall performance is better than the competition algorithm. In [16], a hybrid Harris Hawk Optimization-Sine-cosine algorithm (hHHO-SCA) is proposed to develop a home energy management system based on meta-heuristics, which is used to optimize the scheduling of intelligent devices and reduce the cost of energy use. The experimental results show that hHHO-SCA is relatively effective in reducing cost and peaking ratio, and can be applied to multi-family residential areas. Reference [17] proposes a simplified sine-cosine algorithm (SSCA) to solve the optimal reactive power scheduling (ORPD) problem by estimating control variables. The algorithm uses the sine and cosine functions to generate several random solutions, and seeks the best solution through fluctuations. SSCA is used for ORPD problems to find the best control variable for minimum power loss and maximum net savings. MA-SCA algorithm is proposed in [18], which combines multi-agent system and sine-cosine algorithm, and has been successfully applied to optimize the deployment of distributed energy and shunt capacitor distribution networks. A hybrid intelligence method (ISCA-BP) based on improved sine and cosine algorithm and BP neural network is proposed in [19], which effectively improves the accuracy of transformer fault diagnosis. Reference [20] introduces the Enhanced Sine and Cosine Algorithm (ESCA) aiming to address multi-objective power flow functions encompassing power plant generation cost, loss, emissions, etc., and to enhance voltage stability. The experimental results show that ESCA is superior to other techniques in both

convergence speed and global optimal solution. The improved sine-cosine algorithm has certain advantages in the application of various fields, but it is rarely applied in the field of computing offloading problem under edge computing environment. In this paper, based on the structure of the solution of computing offloading problem, a hybrid sine-cosine genetic optimization algorithm (SCAGA) is proposed by combining the elite strategy with the mutation strategy of genetic algorithm and sine-cosine algorithm.

In this study, we investigated the potential application of the sine and cosine optimization algorithm in making computing offloading decisions. We also introduced an enhanced sine and cosine optimization algorithm (SCAGA), which incorporates the concepts of roulette wheel selection and gene mutation from genetic algorithms, aiming to enhance the efficiency and performance of MEC systems. SCAGA is employed to address the challenges of computing resource allocation and computing offloading in MEC systems, with the following main functions:

1) Construct a multi-user concurrent edge computing system model, considering factors such as edge computing latency, mobile user energy consumption, and edge server computing resources as constraints. The weighted value function, which incorporates edge computing latency and energy consumption, is utilized to assess the calculation offloading strategy. Consequently, the edge computing model's offloading strategy, subject to multiple constraints, is reformulated as an optimization problem of the weighted value function under multi-constraint conditions.

2) Within the "edge-end" model framework, develop a hybrid optimization algorithm that integrates the adaptive sine and cosine optimization algorithm with the genetic algorithm. The adaptive sine and cosine optimization algorithm can yield relatively excellent suboptimal solutions within the established model. Given that the computing offloading strategy matrix comprises {0,1} elements, this format aligns with the initial solution structure of the genetic algorithm. To further improve the solving capability and precision of the hybrid algorithm, a roulette wheel selection genetic algorithm is employed.

3) Apply experiments to verify the optimization effect of the computing offloading strategy based on SCAGA, and compare and analyze it with other classic algorithm in terms of system latency, energy consumption, and other indicators.

2. System model and problem description

The edge computing system composed of edge servers and users is shown in Figure 1, assuming that its wireless communication uses the 5G standard. Each micro workspace has a micro base station, which mainly provides communication resources and does not provide computing resources, MEC servers are deployed in the micro base station.

Assuming the IMTE representation of the micro workspace is $N = \{1,2,3\ldots,N\}$, each IMTE is assigned a delay-sensitive and indivisible computing task, and IMTE utilizes a fixed channel approach to access the network. Within the edge computing system, there are multiple micro workspaces, and the system assigns a dedicated wireless channel to each IMTE requiring computation and offloading.

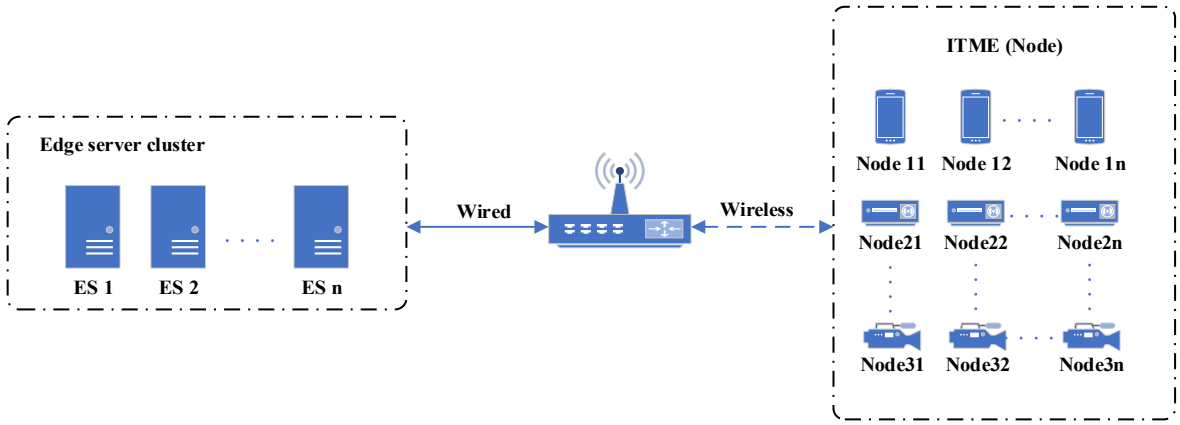


Figure 1. Edge computing architecture.

2.1. Local computing model

Assuming that the i-th user in the micro workspace has a huge amount of data and a latency sensitive computing task to complete, the task is: $task_i = \{M_i, D_i,\}$, Where D_i is the data size of the task (including program code and input parameters, etc.), M_i is the CPU cycle required to process this computing task. IMTE has a certain ability to process computing tasks. When the edge computing network is severely congested, the computing task can be executed locally, and when the edge computing network channel conditions are good, IMTE can choose to unload the computing task to the edge server with rich computing resources.

The latency and energy consumption for executing computing tasks locally are:

$$T_i^{local} = \frac{M_i}{f_i^{local}} \tag{1}$$

$$E_i^{local} = k(f_i^{local})^2 M_i \tag{2}$$

where k is the energy coefficient of the CPU of this IMTE, and it depends on the CPU structure of IMTE, the general value is $k = 10^{-25}$.

f_i^{local} represent the computing power of IMTE, measured in cycles per second of the CPU.

2.2. Computing offloading model

The MEC system is deployed within the micro base station, where edge servers can provision IMTE with limited resources like computing and storage. IMTE uploads computing tasks to micro base stations via the Radio Access Network (RAN). Due to the wired connection between MEC servers and base stations, the delay and energy consumption between MEC and base stations may not be factored in. Every IMTE will be allocated a specific wireless subchannel to access the edge computing network. If p_i is the transmission power of the device, g_i is the wireless channel gain, and σ^2 is the noise power. The transmission rate at which IMTE uploads computing tasks to edge servers is:

$$R_i = W\ log_2(1 + \frac{p_i g_i}{\sigma^2})\ \text{[bit/s]} \tag{3}$$

where W is the transmission bandwidth of each IMTE.

The total delay of IMTE offloading computing tasks to edge servers includes two parts: upload delay T_i^{up} and processing delay T_i^{exe}, respectively represented as:

$$T_i^{up} = \frac{D_i}{R_i} \tag{4}$$

$$T_i^{exe} = \frac{M_i}{f_i^{edge}} \tag{5}$$

$$T_i^{edge} = T_i^{up} + T_i^{exe} = \frac{D_i}{R_i} + \frac{M_i}{f_i^{edge}} \tag{6}$$

where T_i^{edge} is the total latency for IMTE to offload computing tasks to the edge server, f_{edge}^{max} is the computing power of the edge server (in cycles per second of the CPU), f_i^{edge} is to offload the computing resources obtained by the user from the edge server the energy consumption for offloading computing tasks is:

$$E_i^{edge} = \sum_{i=1}^{i=N} \frac{D_i}{R_i} x_i \tag{7}$$

E_i^{edge} is the transmission energy consumption of IMTE when uploading computing tasks.

Due to the typically smaller output size compared to the input and the higher data transmission rate in the downlink relative to the uplink, the transmission delay of the output is disregarded in the model.

2.3. Computing cost model

The cost of an edge computing system primarily comprises the time delay in completing computing tasks and the energy consumption of IMTE. These factors also influence the quality of service provided by the MEC system. Therefore, according to the above model, the computing offloading cost C_i^{total} of each IMTE is defined as:

$$C_i^{total} = \alpha T_i + \beta E_i \tag{8}$$

T_i is the task completion time of IMTE, and E_i is the energy consumption of IMTE. α and β are the delay parameters and energy consumption parameters of the MEC system ($\alpha + \beta = 1$ and $\alpha,\ \beta \in [0,1]$).

By optimizing and adjusting the delay and energy consumption parameters, the edge computing

system can dynamically adapt to varying business requirements under diverse communication and energy conditions. The total system cost C^{total} is used to evaluate the performance of the entire edge computing system, and C^{total} is defined as:

$$C^{total} = \sum_{i=1}^{i=N} (T_i + E_i) \tag{9}$$

$$C^{total} = \sum_{i=1}^{i=N} (1 - x_i) C_i^{local} + x_i C_i^{edge} \tag{10}$$

where $\mathrm{C_i^{local}}$ is the cost for IMTE to execute offloading decisions locally, and $\mathrm{C_i^{edge}}$ is the cost for IMTE to offload computing tasks to edge servers. $x_i \in \{0, 1\}$ is the parameters of computing offloading decision, x_i = 0 indicates that IMTE's computing tasks will be executed locally, while x_i = 1 indicates that IMTE will offload the computing tasks to the edge server.

By combining the above calculation models, the overall calculation model of the system can be obtained:

$$\begin{aligned} &P1: min\{ C^{total} = (1 - x_i) C_i^{local} + x_i C_i^{edge} \} \\ &s.t. \\ &C1: x_i \in \{0,1\} \\ &C2: \alpha + \beta = 1; \alpha, \beta \in [0,1] \\ &C3: \sum_{i \in U} f_i^{edge} < f_{edge}^{max} \\ &C4: 0 < f_i^{edge} < f_{edge}^{max} \end{aligned} \tag{11}$$

In the total calculation model given in formula (11), The minimum value of problem P1 will be obtained under multiple constraints of $\mathrm{C1 - C4}$.

$C1$ indicates that the offloading decision parameter, whose value can only be 0 or 1. A value of 0 indicates that the computing task will be executed locally, while a value of 1 indicates that the computing task will be offloaded to an appropriate edge server for execution.

$C2$ represents the delay coefficient and energy consumption coefficient, used for joint optimization of Calculated delay and energy consumption.

$C3$ indicates the computing resources allocated to a single task, which is less than the total resources of the edge server.

$C4$ represents the computing resources occupied by offloading all computing tasks to edge servers, which is less than the total computing resources of all edge servers.

3. Joint optimization of resource allocation and task offloading decisions

In computational offloading research, joint optimization can enhance the efficient utilization of

computational resources and task execution performance. Resource allocation entails assigning computing tasks to suitable nodes or devices to fulfill their requirements and constraints. Task offloading decisions involve transferring selected tasks from IMTE to edge servers to accomplish objectives like load balancing, latency reduction, or energy conservation. Consequently, jointly optimizing resource allocation and task offloading decisions can significantly enhance the overall efficiency of MEC systems.

The joint optimization of task offloading and resource allocation under multiple constraints creates a nonlinear programming problem involving mixed integers. Due to the interdependency of these constraints, the problem becomes challenging to solve directly. As a result, the MINLP problem is divided into two sub-problems: resource allocation and computational task offloading. To solve these sub-problems, we employ a Sine and Cosine Algorithm (SCA) in conjunction with convex function optimization. SCA, a swarm intelligence optimization algorithm, efficiently reaches near-optimal solutions, reducing problem complexity under specific conditions and ensuring rapid convergence.

3.1. Computational resource allocation

The allocation of computing resources in the edge server can reduce the total delay of the system in structure, improve the service quality of the MEC system and the quality of user experience of IMTE, and enhance the queuing efficiency of computing tasks.

Because the MEC server allocates computing resources only for the user tasks that are determined to be unloaded, given U is the unloaded user set, $U = \{i|a_i = 1\}$,and $s = \{s_1, s_2, s_3, \dots, s_N\}$ is the computing resource allocation set.

$$C_{total}(s) = \sum_{i\in U}(\alpha\frac{D_i}{R_i} + \alpha\frac{M_i}{f_i^{edge}}) + \sum_{i\in U}\beta\frac{p_i D_i}{R_i}$$

$$\begin{aligned} &s.t. \\ &C1: x_i \in \{0,1\} \\ &C2: \alpha + \beta = 1; \alpha, \beta \in [0,1] \\ &C3: \sum_{i\in U} f_i^{edge} < f_{edge}^{max} \\ &C4: 0 < f_i^{edge} < f_{edge}^{max} \end{aligned} \tag{12}$$

Calculate the second derivative of the function to obtain:

$$\frac{\partial^2 C_{total}(f)}{\partial f_i \partial f_j} = \begin{cases} \frac{2\alpha M_i}{(f_i^{edge})^3} \geq 0, i = j \\ 0, i \neq j \end{cases} \tag{13}$$

Since the positive parameters in the Hesse matrix, the Hessian matrix of objective function $C_{total}(s)$ is positively definite, $C_{total}(s)$ is a convex function, and the related optimizations belong to convex optimizations. Therefore, the objective function is solved using the Lagrange multiplier method. Introducing Lagrange multipliers λ, P1 can be rewritten as:

$$L(f_i^{edge}, \lambda) = C_{total}(s) + \lambda(\sum_{i \in U} f_i^{edge} - f_{edge}^{max}) \tag{14}$$

Through KKT conditions, it can be concluded that:

$$\frac{\partial L(f_i^{edge}, \lambda)}{\partial f_i^{edge}} = \frac{\alpha M_i}{f_i^{edge}} + \lambda f_i^{edge} = 0 \tag{15}$$

$$\frac{\partial L(f_i^{edge}, \lambda)}{\partial \lambda} = \sum_{i \in U} f_i^{edge} - f_{edge}^{max} \tag{16}$$

Solve formulas (15) and (16) to obtain the optimal solution $f_i^{edge^*}$:

$$f_i^{edge^*} = \frac{\sqrt{\alpha M_i}}{\sum_{i \in U} \sqrt{\alpha M_i}} f_{edge}^{max} \tag{17}$$

Substitute the optimal solution $f_i^{edge^*}$ into problem P1 in formula (12), it can be concluded that:

$$\begin{aligned} C^{total}(x) &= (1 - x_i)[(\alpha \frac{M_i}{f_i^{local}}) + \beta k (f_i^{local})^2 M_i] \\ &+ x_i[\alpha \frac{D_i}{R_i} + \alpha \frac{M_i}{f_i^{edge^*}} + \beta \frac{p_i D_i}{R_i}] \end{aligned} \tag{18}$$

3.2. Joint Optimization of offloading Decision for Computing Tasks

After completing the allocation of computing resources on edge servers, the original problem (11) is transformed into a task offloading decision problem (19), and the objective model of optimal offloading strategy is obtained under the constraints of the highest tolerance delay for delay sensitive tasks and total computing resources:

$$\begin{aligned} &P2: min\, C^{total}(x) = min \sum_{i=1}^{i=N} (1 - x_i)[(\alpha \frac{M_i}{f_i^{\text{local}}}) + \beta k (f_i^{local})^2] \\ &+ x_i[\alpha \frac{D_i}{R_i} + \alpha \frac{M_i}{f_i^{edge^*}} + \beta \frac{p_i D_i}{R_i}] \\ &s.t. \\ &C1: x_{i \in}[0,1], i \in \{1,2,\ldots,N\} \\ &C2: \sum_{i \in U} f_i^{edge} < f_{edge}^{max} \end{aligned} \tag{19}$$

The model described above represents a 0–1 programming model, Attaining the optimal solution for this problem becomes challenging, especially with concurrent large-scale tasks. Consequently, in meeting multi-objective constraints, a mixed sine-cosine optimization algorithm can be deployed to identify suboptimal solutions for the problem.

4. Computing offloading decision based on SCAGA

4.1. Sine and Cosine Optimization Algorithm (SCA) and its improvement

The Sine and Cosine Optimization Algorithm (SCA) is a stochastic optimization method suitable for addressing the optimization challenges inherent in computing offloading decisions within MEC systems. The optimization process in SCA entails two key stages: during the exploration stage, the algorithm rapidly explores feasible regions within the solution space by incorporating specific random solutions; in the exploitation stage, the random solutions undergo gradual changes, with a slower rate of change compared to the exploration phase.

In the Sine and Cosine Algorithm (SCA), the initial candidate solution is randomized. Then, the value of the current solution in each dimension is updated through a combination of the sine or cosine function along with random factors. The update equation is shown below:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 * sin(r_2) * |r_3 P_i^t - X_i^t|, r_4 < 0.5 \\ X_i^t + r_1 * cos(r_2) * |r_3 P_i^t - X_i^t|, r_4 > 0.5 \end{cases} \tag{20}$$

In the formula (20), X_i^t represents an individual with the i-th dimension and the t-th iteration; r_2 is a random number between 0 and 2π; r_3 is a random number between 0 and 2; r_4 is a random number between 0 and 1; P_i^t represents an optimal individual with the i-th dimension and the t-th iteration。r_1 can be obtained from the following equation:

$$r_1 = a - (t\frac{a}{T})^2 \tag{21}$$

In Eq (21), a smaller value of A will help enhance the local development ability of algorithm, while a larger value of A will help improve the global exploration ability of algorithm. In recent years, SCA algorithm has been widely used in various engineering fields because of its excellent performance in exploration ability. On the basis of formula (21), this paper rewrites r_1 as:

$$r_1(t) = a_{start} - (a_{start} - a_{end})^2 * \ln(1 + \frac{(e-1)*t}{T}) \tag{22}$$

$a_{start} = 1$ and $a_{end} = 0$ are the initial and final values of control parameter a ($a_{start} > a_{end} \geq 0$). It can be seen from formula (22) that $r_1(t)$ changes nonlinearly with the increase of iterations, which can effectively balance the exploration and development capabilities of SCA.

In order to increase SCA's global collection and search ability and escape local optimal ability, Levy flight strategy is introduced in this paper, which makes the algorithm more randomness in the optimization process and avoids the algorithm falling into local optimal.

$$\delta_u = [\frac{\Gamma(1+\tau)\sin\left(\frac{\pi\tau}{2}\right)}{\Gamma\left(\frac{1+\tau}{2}\right)\tau*2^{\frac{\tau-1}{2}}}]^{\frac{1}{\tau}}, \delta_v = 1 \tag{23}$$

$$levy(\tau) = \frac{u}{|v|^{-\tau}} \tag{24}$$

In formula (24), $u \sim N(0, \delta_u), v \sim N(0, \delta_v)$, take $\tau = 1.5$.

In this paper, Levy flight strategy is added on the basis of formula (20), so formula (20) is rewritten as:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1(\mathrm{t}) * sin(r_2) * |r_3 P_i^t - X_i^t| + 0.1 * levy(\tau), r_4 < 0.5 \\ X_i^t + r_1(\mathrm{t}) * cos(r_2) * |r_3 P_i^t - X_i^t| + 0.1 * levy(\tau), r_4 > 0.5 \end{cases} \tag{25}$$

4.2. SCAGA algorithm

The solution of the computing offloading model proposed in this article consists of a binary structure composed of {0, 1} elements. Due to its alignment with the genetic algorithms' gene concept, this enables genetic algorithms to bolster local development capabilities and enhance algorithm accuracy. Following the acquisition of an excellent suboptimal solution group in SCA, diverse solutions are chosen as the genetic algorithm's genome, and various operations, including selection and mutation, are employed to ultimately yield an outstanding binary solution.

Since the decision structure for computing offloading is: $A = \{0, \ldots\ldots, 1\}$, which is highly similar to the structure of genes in genetic algorithms. In the mutation step of the genetic algorithm, each decision element undergoes a certain probability of mutation (mutation probability $\varsigma = 0.05$). Since the coupling relationship between each decision variable, the mutation of each gene locus will have a certain impact on the offloading utility. The specific steps of genetic mutation are shown in the following Figure 2.

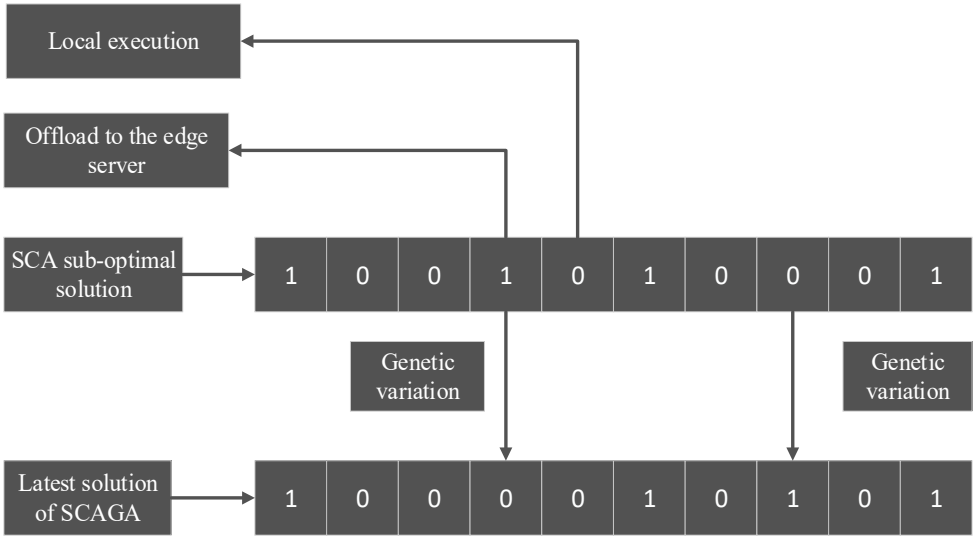


Figure 2. Genetic variation diagram.

4.3. SCAGA architecture

The SCAGA algorithm integrates the gene mutation of the genetic algorithm and roulette wheel selection method on the basis of the SCA optimization algorithm, and structurally optimizes the offloading decision based on the SCA optimization algorithm. The specific process is shown in Figure 3.

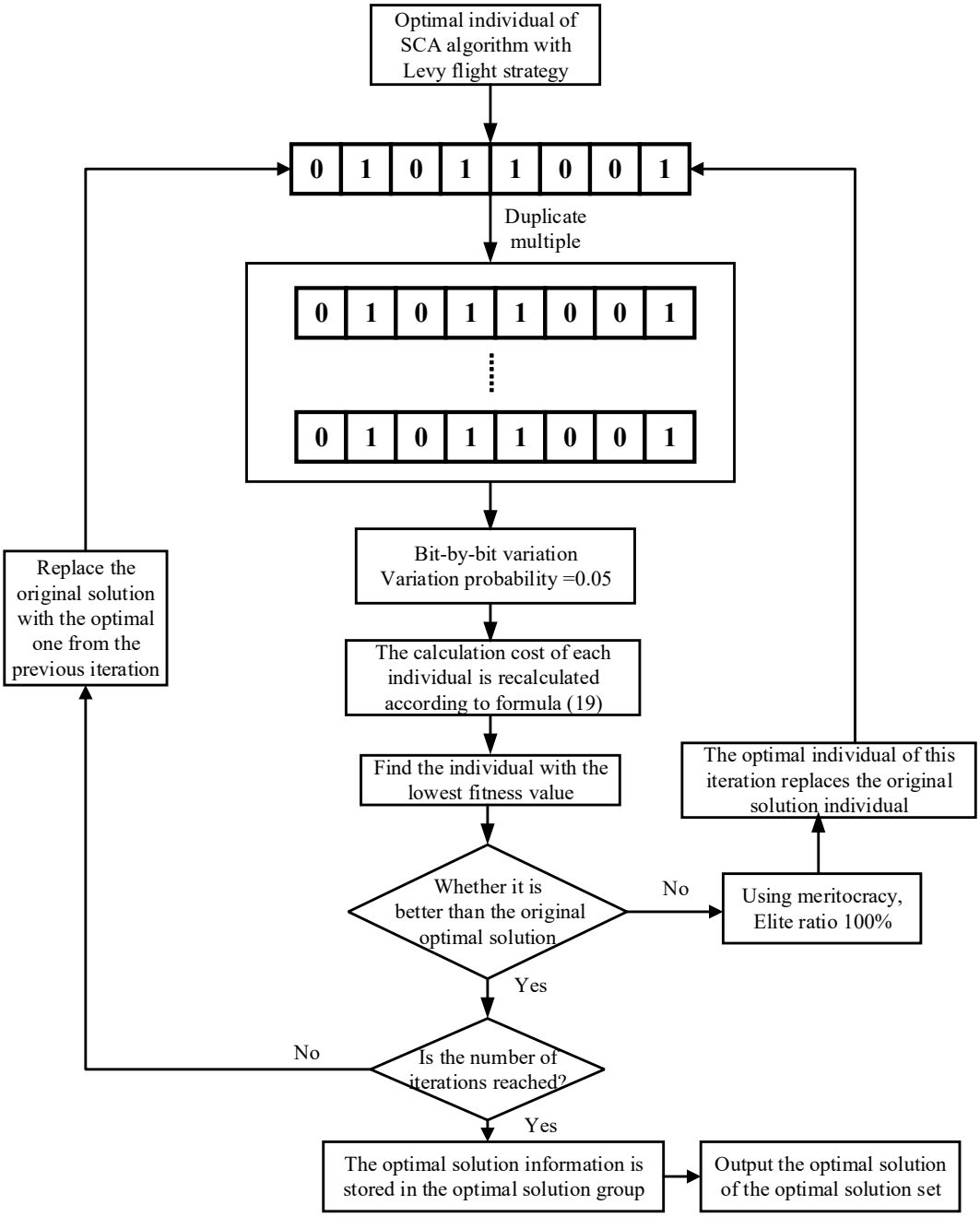


Figure 3. Genetic variation diagram.

The specific algorithm flowchart is as follows:

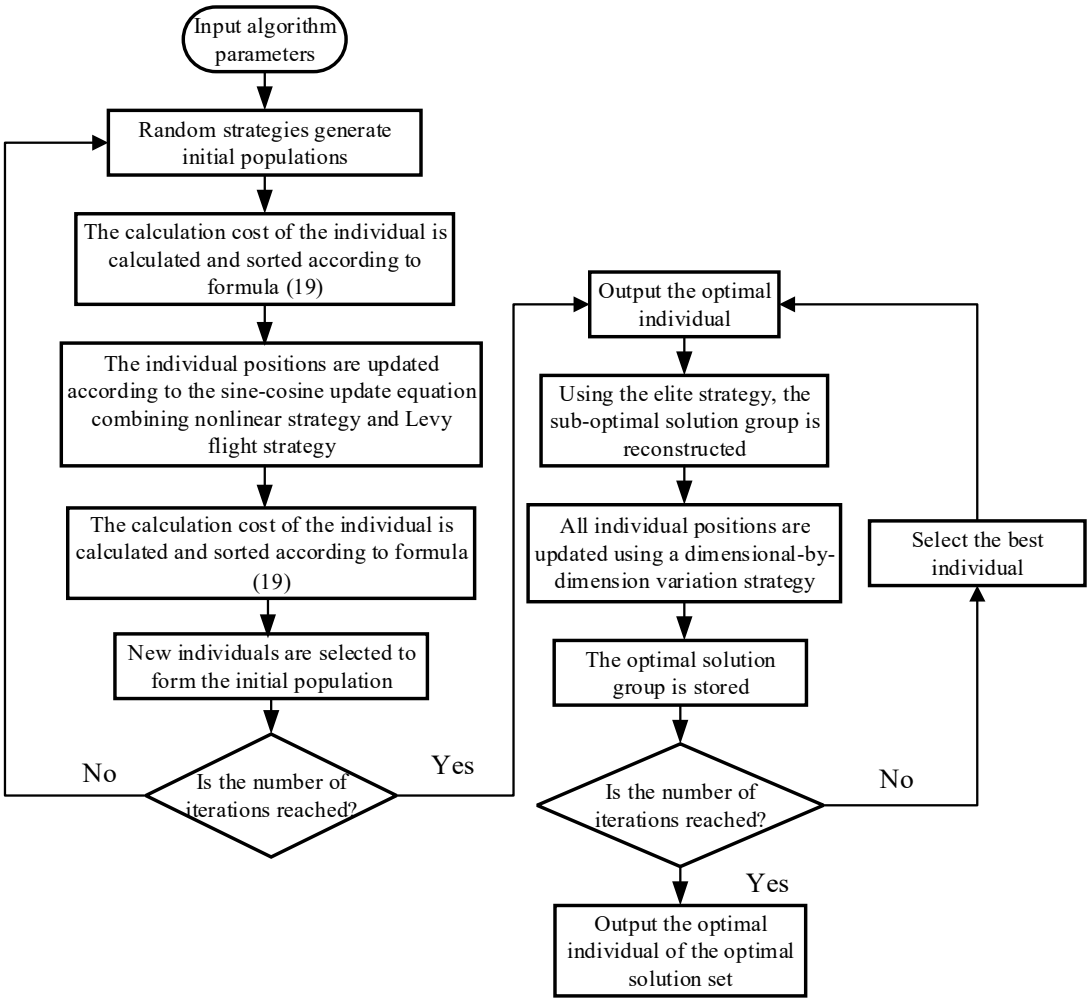


Figure 4. Flow chart of SCAGA algorithm.

5. Experiment and analysis

In order to verify the performance of SCAGA, local computation offloading algorithm (LA), random offloading algorithm (RA), computation offloading algorithm based on sine and cosine optimization algorithm (SCA), computation offloading algorithm based on genetic algorithm (GA), computation offloading algorithm based on COSCA [21], and complete offloading algorithm (AA) were selected. Under the same experimental conditions and environment, comparisons were made on indicators such as system latency, user energy consumption, and computational costs.

All experiments were implemented using Python programming. The operating system of experimental environment is Windows 10 64 bit, and the hardware configuration is AMD Ryzen R5 5600 CPU with 16GB of memory.

5.1. Experimental parameter settings

The simulation scenario for SCAGA is depicted in Figure 1, and all simulation parameters are randomly generated within the specified range, as illustrated in Table 2. Each decision-making

method underwent 10 simulations, and the results were then averaged across these 10 experiments. The communication parameters adhere to the specifications of the third-generation partnership project [22].

Table1. Experimental parameter settings.

Parameters	Values
Computing power of mobile device (GHz)	1–1.2
Computing power of edge server (GHz)	40
Data volume of task (Mb)	0.1–1.2
Computing intensity of task (Hz)	600–1200
Transmission power of mobile device (W)	0.5
Transmission channel gain (dB)	137 + 30*$log_{10}\,d$ [22]
Transmission distance (m)	0–1000
Population size	20
Number of iterations	60
Gaussian channel noise (W)	2×10^{-13}

5.2. Impact of IMTE quantity on Calculated cost

The purpose of this experiment is to compare the offloading efficiency of various algorithms under different task quantities, including the system latency and the energy consumption of mobile user.

This experiment sets the computing power of MEC to 40GHz, $\alpha = 0.8$, $\beta = 0.2$. Test the Calculated cost generated when the number of mobile users is 20, 25, 30, 35, 40, 45, and 50. The experimental results are shown in Figure 5 and Table 2.

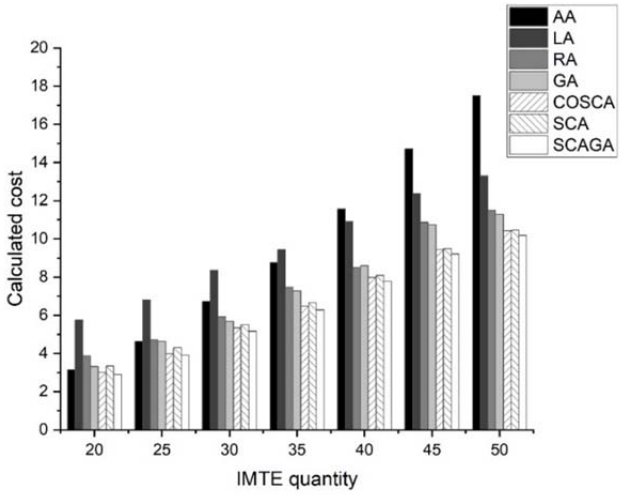


Figure 5. IMTE quantity - Calculated cost of different algorithms.

Table2. IMTE quantity - Calculated cost relationship table.

	Calculated cost						
IMTE quantity	20	25	30	35	40	45	50
AA	3.14	4.63	6.73	8.76	11.56	14.72	17.51
LA	5.76	6.81	8.36	9.43	10.90	12.37	13.29
RA	3.88	4.72	5.94	7.46	8.49	10.87	11.50
GA	3.31	4.63	5.68	7.28	8.59	10.73	11.27
COSCA	3.00	3.99	5.34	6.48	7.97	9.43	10.43
SCA	3.34	4.30	5.51	6.65	8.09	9.49	10.46
SCAGA	2.91	3.90	5.16	6.27	7.77	9.20	10.18

In mobile edge computing, the decision-making performance of the full offload algorithm (AA) gradually declines as the number of mobile users increases. Initially, the local offload algorithm (LA) demonstrates the poorest performance with a small number of mobile users; however, as the user count grows, its calculated cost gradually improves. This shift occurs due to the finite nature of resources available on edge servers—a scenario in which, if all users opt for offloading their computing tasks, the edge servers allocate fewer computing resources compared to local computing resources.

Other optimization algorithms such as random offloading algorithm (RA), GA algorithm, COSCA algorithm, and SCA algorithm are also superior to AA algorithm and LA algorithms. The offloading scheme based on SCAGA algorithm has the best performance, compared to the offloading schemes of AA algorithm, LA algorithm, RA algorithm, GA algorithm, COSCA algorithm, and SCA algorithm. When the number of IMTEs is 35, their Calculated cost decreases by 32.76, 33.51, 15.95, 13.87, 3.24 and 5.71%, respectively. Therefore, in the unloading problem of edge computing environment, the unloading scheme based on SCAGA algorithm has a great advantage. And during the process of increasing the number of IMTEs from 20 to 50, the SCAGA based offloading scheme still outperforms the comparison algorithms listed in the article. As the number of IMTEs continues to increase, the benefits of SCAGA based offloading schemes become more apparent compared to other comparative algorithms.

5.3. Impact of IMTE quantity on system latency

Table 3 and Figure 6 show the influence of the number of IMTE on the calculation delay. Choosing midpoint 35 [20, 50] as the value of IMTE helps to observe the sensitivity of the system to this variable and to some extent represents the entire range. The experimental results show that in terms of handling system latency, the offloading scheme based on SCAGA algorithm reduces system latency by 33.69, 34.55, 10.39, 8.86, 3.17 and 3.83% respectively compared to AA, LA, RA, GA, COSCA and SCA, demonstrating certain advantages.

Regarding the impact of the increase in the number of IMTEs on computation latency, it can be observed that as the number of IMTEs increases, the computation latency of each algorithm also shows a corresponding increasing trend. However, with an increase in the number of IMTEs, the offloading scheme based on the SCAGA still maintains a competitive advantage over other algorithms. This can be attributed to the fact that the SCAGA is more effectively adapted to the

characteristics of large-scale tasks, thus exhibiting better robustness and effectiveness in handling system delays. Therefore, the unloading scheme based on SCAGA shows obvious potential advantages in edge computing environment, especially in processing system delay. These results provide important research reference value for task scheduling and optimization of SCAGA in edge computing environment.

The relevant experimental results are shown in Figure 6 and Table 3.

Table 3. Calculated delay-IMTE quantity relationship table.

	Calculated delay /s						
IMTE quantity	20	25	30	35	40	45	50
AA	4.03	6.94	8.98	12.88	15.39	19.34	24.66
LA	6.44	9.02	9.81	12.03	12.72	14.22	16.36
RA	4.25	6.36	7.88	9.53	10.24	11.87	14.61
GA	5.47	7.36	7.62	9.37	10.68	11.65	14.12
COSCA	3.61	5.73	6.69	8.82	9.69	11.25	13.38
SCA	3.71	5.9	6.79	8.88	9.67	11.18	13.35
SCAGA	3.43	5.6	6.51	8.54	9.71	10.83	12.97

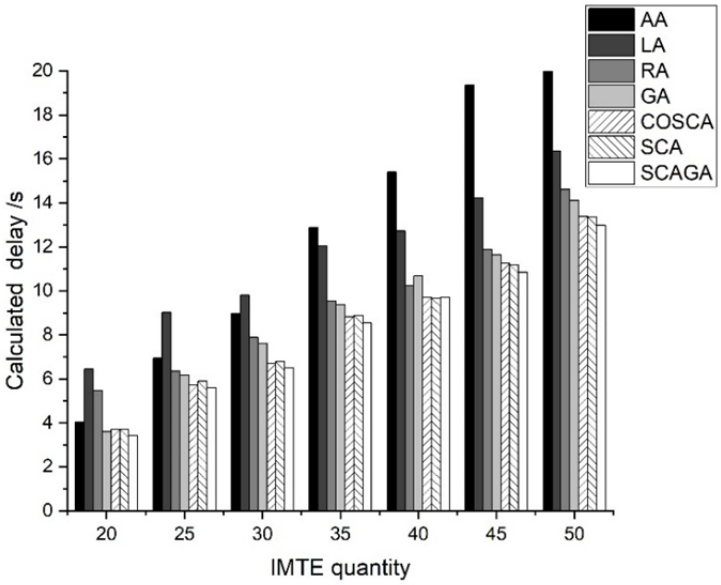


Figure 6. Calculated delay-IMTE quantity relationship diagram.

5.4. Impact of computing resources in edge server on Offload effect

As shown in Table 4 and Figure 7 (The experimental parameters are set to $\alpha = 0.8$, $\beta = 0.2$), overall, with the increase of MEC computing resources, the computational cost based on AA continues to decrease and gradually surpasses that of LA. This experiment proves that with the increase of computing resources, mobile users will obtain less computing latency and energy

consumption when offloading computing tasks to MEC. With the increase of MEC computing resources, the computing costs of offloading schemes based on AA, LA, RA, GA, COSCA, SCA, and SCAGA are gradually decreasing. In the process of linear increase of MEC computing resources, the computing costs of offloading schemes based on SCAGA algorithm are always lower than those based on AA, LA, RA, GA, COSCA, and SCA. Compared to GA based offloading solutions, the computational cost of SCA based offloading solutions will continue to increase. The computational cost of COSCA based offloading schemes is much lower when MEC resources are limited compared to SCAGA based offloading schemes. As MEC resources continue to increase, the computational cost of COSCA based offloading schemes gradually approaches that of SCAGA based offloading schemes. This indicates that the offloading scheme based on SCAGA has strong robustness and optimization in the process of constantly changing MEC resources.

Table 4. Table of the impact of computing resources on Calculated cost.

	Calculated cost						
computing resource/GHz	10	20	30	40	50	60	70
AA	34.15	17.19	11.94	8.84	7.20	6.44	5.34
LA	9.56	9.58	9.52	9.57	9.59	9.51	9.59
RA	12.84	9.67	8.82	6.97	6.97	6.59	6.34
GA	10.62	8.31	7.71	7.15	6.81	6.55	6.83
COSCA	9.99	8.37	7.66	6.59	6.14	5.75	5.11
SCA	8.74	7.90	7.48	6.73	6.32	6.02	5.81
SCAGA	8.66	7.81	7.30	6.37	5.94	5.59	5.00

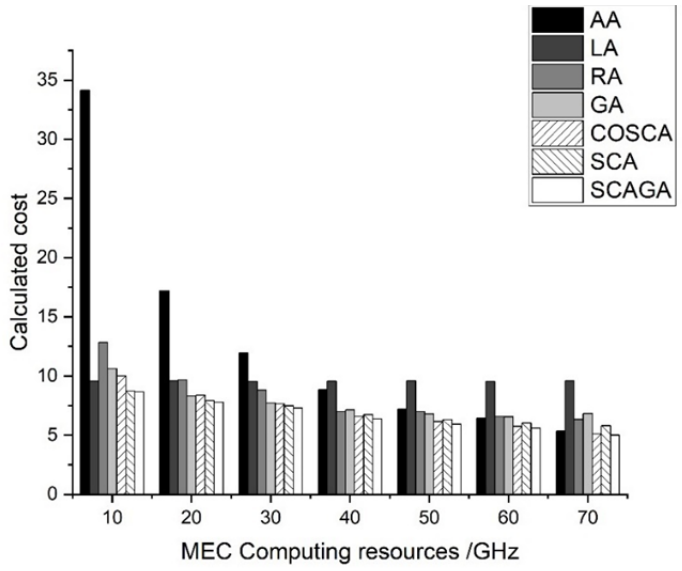


Figure 7. Diagram of the impact of computing resources on Calculated cost.

5.5. Impact of different computational intensities on offloading utility Sub-subheading

As shown in Table 5 and Figure 8, when the number of IMTEs is 35, $\alpha = 0.8$, $\beta = 0.2$ Under experimental conditions with MEC computing resources of 40GHz, simulation results show that with the increase of computing task intensity (computing cycle required for computing tasks), the computing cost of GA, COSCA, SCA based offloading schemes will not change dramatically with changes in computing tasks. This indicates that offloading schemes based on GA, COSCA, and SCA have certain robustness. At the same time, it can be seen that the computational cost of the SCAGA based uninstallation scheme is lower than that of the AA, LA, RA, GA, COSCA, and SCA uninstallation programs. Compared with the offloading schemes based on AA, LA, RA, GA, COSCA, and SCA, the computational cost of the proposed offloading scheme in this paper is roughly reduced by 27.3, 32.2, 12.2, 10.5, 4.1 and 4.6%, respectively.

Table 5. Table of the impact of computing intensity on Calculated.

	Calculated cost					
computing intensity/GHz	0.2	0.4	0.6	0.8	1.0	1.2
AA	8.82	8.76	8.64	9.10	8.54	9.22
LA	9.51	9.41	9.20	9.75	9.13	9.92
RA	7.34	7.00	7.15	7.54	7.59	7.34
GA	7.21	6.89	7.05	7.42	7.12	7.38
COSCA	6.58	6.56	6.37	6.78	6.38	7.03
SCA	6.71	6.62	6.47	6.88	6.58	7.21
SCAGA	6.57	6.35	6.17	6.52	6.18	6.82

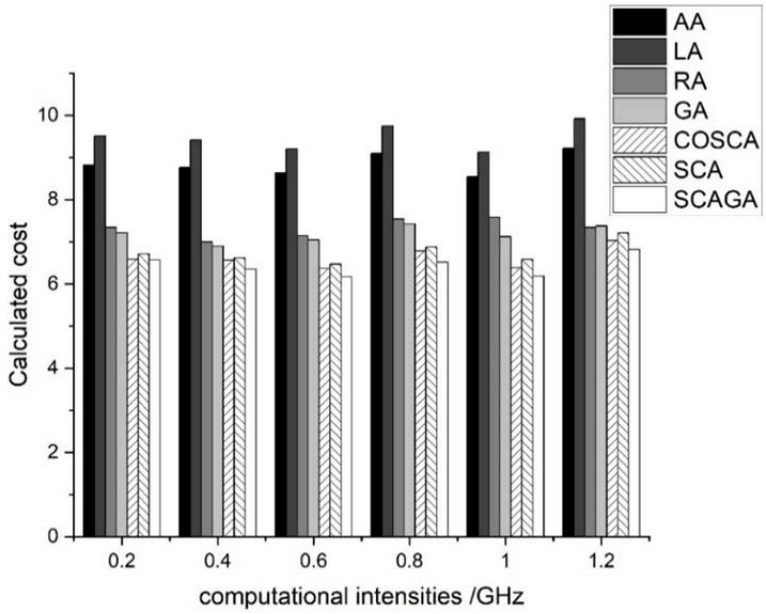


Figure 8. Computing intensity-offloading utility diagram.

5.6. The impact of delay parameter (α) and energy consumption parameter (β) on the Calculated cost

This section of the experiment is a simulation experiment conducted in an environment with MEC computing resources of 40GHz and 35 tasks. In order to explore the effects of latency and energy consumption parameters on computational costs in the offloading scheme based on the SCAGA algorithm, this section set up three sets of experiments (Experiment 1: $\alpha = 0.8$, $\beta = 0.2$, Experiment 2: $\alpha = 0.5$, $\beta = 0.5$, Experiment 3:$\alpha = 0.2$, $\beta = 0.8$). The specific experimental results are shown in Figure 9. From Figure 9, it can be concluded that under the same experimental environment, the larger the delay parameter, the higher its corresponding computational cost. Conversely, the smaller the delay parameter, the lower its corresponding computational cost. Experimental parameters and energy consumption parameters do not require optimization algorithms for optimization. The delay parameters and energy consumption parameters are considered by mobile users based on the network environment and their own battery level. If the delay sensitivity of the task is strong, the delay parameters can be adjusted higher. If the battery level of the task is low or the IMTE needs to work for a long time and consumes more power, the high energy consumption parameters can be adjusted.

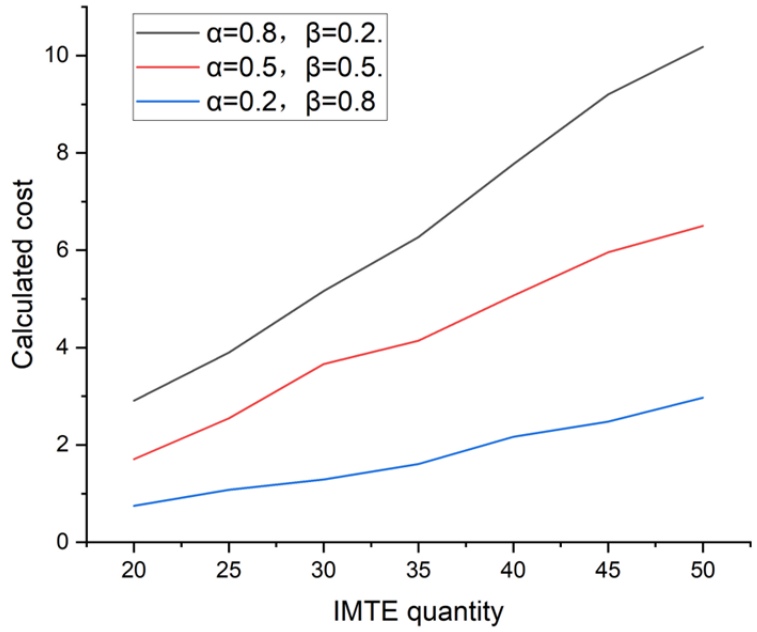


Figure 9. The impact of weight changes on computational costs.

5.7. Comparison of convergence of various algorithms

This section provides a thorough analysis of the convergence and accuracy of the proposed SCAGA algorithm. In this section of the simulation experiment, the experimental parameters are set to a. In this section, we will analyze the convergence of the algorithm in detail through two

simulation experiments, $\alpha = 0.8$, $\beta = 0.2$ and The number of IMTEs is set to 35. Through in-depth comparison of the iterative graphs of the algorithms, it was found that the SCAGA algorithm exhibited good convergence. Specifically, the convergence of the SCAGA algorithm far exceeds that of traditional genetic algorithms (GA) and is slightly better than the COSCA algorithm. This indicates the improvement of our proposed algorithm compared to traditional algorithms, and it has a faster convergence speed when solving problems.

In addition, SCAGA also demonstrates outstanding performance in terms of accuracy. The experimental results show that the accuracy of SCAGA algorithm is much higher than SCA and GA algorithms, and slightly better than COSCA algorithm. This means that our algorithm can more reliably find the optimal solution or approach the optimal solution, providing more reliable and effective support for solving practical problems.

These findings not only emphasize the potential of the SCAGA algorithm in solving optimization problems, but also demonstrate that our method has achieved better performance in the optimization process compared to traditional methods and other improved algorithms. The algorithm iteration diagrams of each comparative algorithm are shown in Figures 10, respectively

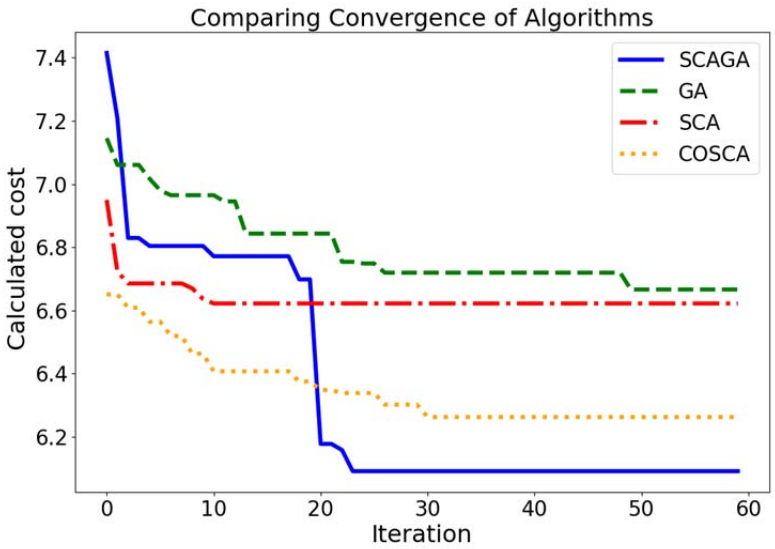
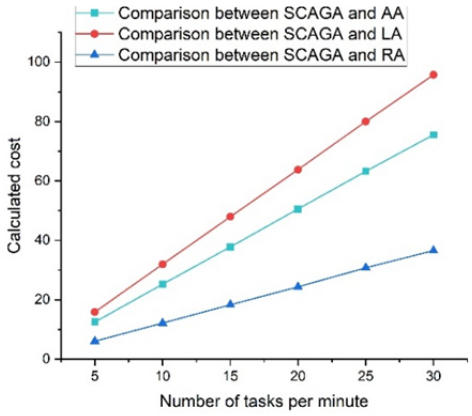


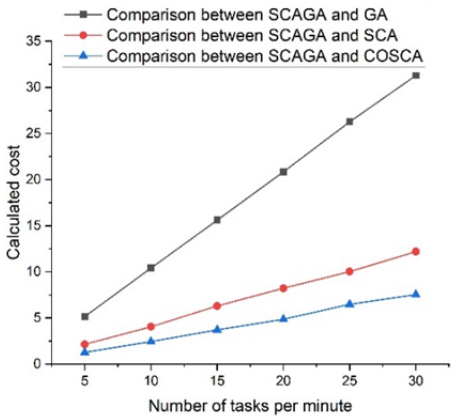
Figure 10. Algorithm iteration diagram.

5.8. The effect of the number of tasks per minute on the computational cost

All of the above experiments were conducted with one computing task per IMTE at a certain point in time. The experimental conditions in this section were set as IMTE would produce multiple tasks within one minute. In this experiment, 5, 10, 15, 20, 25 and 30 computing tasks were set within one minute, and the number of IMTE was set to 35. $\alpha = 0.8$, $\beta = 0.2$, the total computing resources of the edge server is 40 GHz, and other simulation parameters are set according to Table 1. The following figure shows the computational cost reduction of SCAGA's unloading scheme compared to the unloading scheme of other comparison algorithms. That is, the calculation cost of SCAGA scheme minus the calculation cost of other schemes.



(a)Comparison of SCAGA with baseline algorithms



(b)Comparison of SCAGA and swarm intelligence algorithm

Figure 11. Comparison of SCAGA and other algorithms under multi-task conditions.

The experiment in this section assumes that each IMTE will generate 5–30 computing tasks per minute. According to the above simulation data, the more computing tasks generated per minute, the more the unloading scheme of SCAGA will reduce the computing cost of unloading schemes of other algorithms, especially those based on AA, LA, RA and GA. SCAGA's offloading solution will get better as the number of computing tasks increases. When the number of computing tasks per minute increases, the gap between the computing cost and that of SCAGA is also increasing. When the number of tasks per minute is 30, the gap between the computing cost of SCa-based unloading scheme and that of SCAGA is 12, which indicates that the more computing tasks per minute, the greater the number of computing tasks per minute. The computational cost of SCAGA compared to the computational cost of SCA will become more and more obvious.

6. Conclusions

For the challenge of computing offloading in a Multi-access Edge Computing (MEC) system involving concurrent multitasking, the allocation of computing resources in an edge server is considered. This paper introduces a Computing Offloading Strategy based on the SCAGA algorithm. The SCAGA algorithm initially employs the cosine optimization algorithm to identify suboptimal solutions, and subsequently utilizes the roulette wheel selection method and genetic algorithm's gene mutation approach for continuous iteration and calculation to arrive at the optimal solution. Through addressing the task offloading problem in MEC systems, a superior computing offloading decision is ultimately achieved.

Simulation experiments have verified the feasibility of the SCAGA algorithm in computing offloading decisions. Compared with other offloading algorithms, the SCAGA can effectively reduce system latency, energy consumption of mobile user, and optimize Calculated cost.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. D. Huang, L. Yu, J. Chen, T. Wei, Research on joint computation offloading and resource allocation strategy for mobile edge computing, *J. East China Normal Univ.*, **2021** (2021), 88–99. https://doi.org/ 10.3969/j.issn.1000-5641.2021.06.010
2. T. X. Tran, D. Pompili, Joint task offloading and resource allocation for multi-server mobile-edge computing networks, *IEEE Trans. Veh. Technol.,* **68** (2018), 856–868. https://doi.org/10.1109/TVT.2018.2881191
3. L. Tang, Y. J. Hu, T. Liu, Q. B. Chen, Task offloading and resource allocation algorithm based on Lyapunov in mobile edge computing, *Comput. Eng.*, **47** (2021), 29–36. https://doi.org/10.19678/j.issn.1000-3428.0058268
4. B. Wu, J. Zeng, L. Ge, X. Su, Y. Tang, Energy-latency aware offloading for hierarchical mobile edge computing, *IEEE Access*, **7** (2019), 121982–121997. https://doi.org/10.1109/ACCESS.2019.2938186
5. M. Hu, Z. Xie, D. Wu, Y. Zhou, X. Chen, L. Xiao, Heterogeneous edge offloading with incomplete information: A minority game approach, *IEEE Trans. Parallel Distrib. Syst.*, **31** (2020), 2139**–**2154. https://doi.org/10.1109/TPDS.2020.2988161
6. J. S. Zheng, X. L. Jia, Double-auction-based task offloading and resource allocation strategy for mobile edge computing, *Comput. Syst. Appl.*, **32** (2023), 45–56. https://doi.org/10.15888/j.cnki.csa.009110
7. X. J. Zang, W. G. Wu, C. Zhang, Y. X. Chai, S. Y. Yang, X. Wang, Energy-efficient computing offloading algorithm for mobile edge computing network, *J. Software*, **34** (2023), 849–867. https://doi.org/10.13328/j.cnki.jos.006417
8. H. Meng, R. Huo, Q. Y. Guo, T. Huang, Y. J. Liu, Machine learning-based stochastic task offloading algorithm in mobile-edge computing, *J. Beijing Univ. Posts Telecommun.*, **42** (2019), 25–30. https://doi.org/10.13190/j.jbupt.2018-078
9. X. Qiu, L. Liu, W. Chen, Z. Hong, Z. Zheng, Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing, *IEEE Trans. Veh. Technol.*, **68** (2019), 8050–8062. https://doi.org/10.1109/TVT.2019.2924015
10. S. F. Zhu, M. Y. Zhao, Z. Y. Chai, Computing offloading scheme based on particle swarm optimization algorithm in edge computing scene, *J. Jilin Univ.*, **52** (2022), 2698–2705. https://doi.org/10.13229/j.cnki.jdxbgxb20210328
11. Y. L. Cong, W. X. Sun, K. Xue, Z. H. Qian, M. S. Chen, Research on task offloading strategy of Internet of vehicles based on improved hybrid genetic algorithm, *J. Commun.*, **43** (2022), 77–85. https://doi.org/10.11959/j.issn.1000-436x.2022188

12. Z. X. Yang, W. Z. Zhang, P. Cheng, S. H. Xie, Computation offloading decision strategy based on hybrid fruit fly optimization algorithm, *J. Chinese Comput. Syst.*, **44** (2023), 1290–1296. https://doi.org/10.20009/j.cnki.21-1106/TP.2021-0749
13. S. Chauhan, G. Vashishtha, L. Abualigah, A. Kumar, Boosting salp swarm algorithm by opposition-based learning concept and sine cosine algorithm for engineering design problems, *Soft Comput.*, **2023** (2023), 1–28. https://doi.org/10.1007/s00500-023-09147-z
14. M. Zhong, J. Wen, J. Ma, H. Cui, Q. Zhang, M. K. Parizi, A hierarchical multi-leadership sine cosine algorithm to dissolving global optimization and data classification: The COVID-19 case study, *Comput. Biol. Med.*, **164** (2023), 107212. https://doi.org/10.1016/j.compbiomed.2023.107212
15. S. Raj, C. K. Shiva , B. Vedik, S. Mahapatra, V. Mukherjee, A novel chaotic chimp sine cosine algorithm Part-I: For solving optimization problem, *Chaos Solitons Fractals*, **173** (2023), 113672. https://doi.org/10.1016/j.chaos.2023.113672
16. K. Paul, D. Hati, A novel hybrid Harris hawk optimization and sine cosine algorithm based home energy management system for residential buildings, *Building Serv. Eng. Res. Technol.*, **2023** (2023), 01436244231170387.
17. S. K. Gupta, M. K. Kar, L. Kumar, S. Kumar, A simplified sine cosine algorithm for the solution of optimal reactive power dispatch, *Int. Trans. Electr. Energy Syst.*, **2022** (2022). https://doi.org/10.1155/2022/2165966
18. C. D. Patel, T. K. Tailor, Multi-agent based sine–cosine algorithm for optimal integration of DERs with consideration of existing OLTC in distribution networks, *Appl. Soft Comput.*, **117** (2022), 108387. https://doi.org/10.1016/j.asoc.2021.108387
19. J. Cheng, Z. Feng, Y. Xiong, Transformer fault diagnosis based on an improved sine cosine algorithm and BP neural network, *Recent Adv. Electr. Electron. Eng.*, **15** (2022), 502–510. https://doi.org/10.2174/2352096515666220819141443
20. S. Karimulla, K. Ravi. Solving multi objective power flow problem using enhanced sine cosine algorithm, *Ain Shams Eng. J.*, **12** (2021), 3803–3817. https://doi.org/10.1016/j.asej.2021.02.037
21. W. Y. Guo, Y. Wang, F. Dai, T. Liu, Alternating sine cosine algorithm based on elite chaotic search strategy, *Control Decis.*, **34** (2019), 1654–1662. https://doi.org/10.13195/j.kzyjc.2018.0006
22. 3GPP Technical Specification Group Radio Access Network, Further advancements for E-UTRA physical layer aspects (Release 9), 3GPP TR 36.814 V9.0.0, 2010. Available from: https://documents.pub/document/3gpp-tr-36814-v900-2010-03.html?page=1.