Coordinated Allocation of Mobile Edge Computing Server Resources Based on Improved Aquila Optimizer Algorithm

Yujing Wang, Yuanyuan Wei, Hui Li, Xiaoyu Du, and Wen Zhang

***Abstract*—In ultra-dense network environments, the collaborative allocation of resources among multiple edge computing servers presents a complex global optimization challenge. While Aquila Optimizer (AO) can be applied for resource optimization, it is prone to premature convergence to local optima and exhibits relatively low task completion rates. To address these issues, this paper proposes a Hybrid Strategy Aquila Optimizer (HSAO) with four key innovations. Firstly, a candidate pattern pruning method is introduced to reduce redundant decision variables and lower computational complexity. Furthermore, a Tent chaotic mapping mechanism is adopted for population initialization to enhance randomness. During the AO's expanded exploration phase, a random spiral update mechanism is incorporated to enhance the algorithm's global search ability and robustness. Finally, a hybrid strategy combining stochastic reverse-based exploration with the Nelder-Mead method is adopted to enhance the search capability of individuals, thus improving the algorithm's effectiveness in avoiding entrapment in local optima. Comprehensive testing using CEC2017 benchmark functions and practical case studies validate HSAO's superior performance, including faster task execution, reduced service latency, and improved user satisfaction in ultra-dense network environments.**

***Index Terms*—mobile edge computing, bionic learning algorithm, aquila optimizer algorithm, resource scheduling.**

I. Introduction

THE rapid development of 5G communication technology has given rise to a wide range of new computation-intensive applications. In the context of Ultra-Dense Network (UDN), Mobile Edge Computing [1] (MEC) servers push mobile computing, network control, and storage capabilities toward the network edge [2], which enables resource-limited Mobile Smart Devices (MSD) to run computationally intensive and latency-sensitive applications, meeting the requeirements of low latency, low energy consumption, and high reliability [3] in new application domains such as virtual reality [4], health monitoring [5], and autonomous vehicles [6]. The network scenario for ultra-dense deployment of MEC servers is shown in Figure 1.

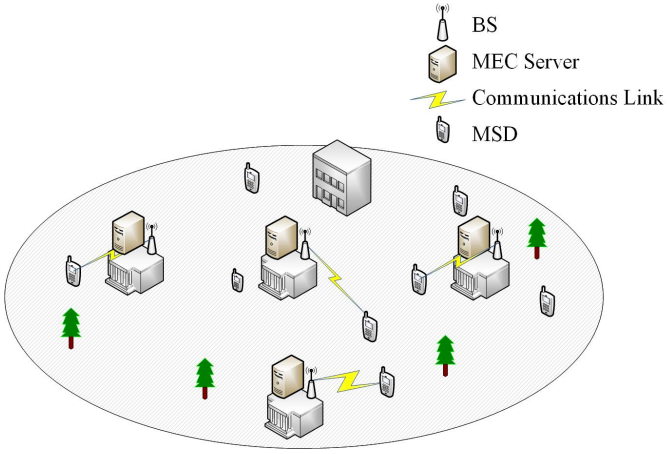


Fig. 1: Network scenarios for ultra-dense deployment of MEC servers

Currently, resource scheduling models in edge computing environments primarily include constrained continuous problem models [7], bin-packing problem models [8], mixed integer nonlinear problem models [9], and multi-objective optimization models [10]. These problems are generally challenging to solve due to their inherent complexity and are often classified as NP-hard. In recent years, with the advancement of stochastic search theory, numerous novel bio-inspired learning algorithms, such as Genetic Algorithms [11], Artificial Fish Swarm Algorithm [12], and Particle Swarm Optimization Algorithm [13], have been applied to resource scheduling problems in edge computing environments. However, these algorithms suffer from limitations such as premature convergence to local optima and slow convergence speeds. Therefore, researchers have focused on improving algorithm performance by enhancing convergence speed and precision while avoiding entrapment in local optima.

In this study, we propose a Hybrid Strategy Aquila Optimizer (HSAO) algorithm to address these challenges. The proposed algorithm introduces several innovative strategies:

Manuscript received December 3, 2024; revised May 15, 2025.

This work was supported in part by the Special Project for Key R&D and the Promotion of Science, Technology Department of Henan Province (222102210272, 232102211009 and 242102210202, 242102210196), the Postgraduate Education Reform and Quality Improvement Project of Henan Province under Grant (YJS2022JD26), the Major Science and Technology Projects of Henan Province, and the Major Public Welfare Projects of Henan Province (201300210400), Kaifeng Science and Technology Development Plan (2201010).

Yujing Wang is an associate professor of the School of Computer and Information Engineering, Henan University, Kai Feng, Henan, 475001, China (e-mail:yjwang@henu.edu.cn).

Yuanyuan Wei is a postgraduate student of the School of Computer and Information Engineering, Henan University, Kai Feng, Henan, 475001, China (e-mail:wyy0916@henu.edu.cn).

Hui Li is an associate professor of Technology&Media University of Henan Kaifeng, Kai Feng, Henan, 475000, China (Corresponding author, e-mail:HumcLihui@outlook.com).

Xiaoyu Du is a professor of Henan Engineering Laboratory of Spatial Information Processing , Henan University, Kai Feng, Henan, 475001, China (e-mail:dxy@henu.edu.cn).

Wen Zhang is a postgraduate student of the School of Computer and Information Engineering, Henan University, Kai Feng, Henan, 475001, China (e-mail:3020046235@qq.com).

1) Candidate Pattern Pruning: This approach decreases the dimensionality of the decision space, thereby streamlining the optimization procedure.

2) Tent Chaotic Mapping Mechanism: Used for population initialization, this strategy enhances the algorithm's exploration efficiency during the early stages.

3) Random Spiral Update Mechanism: Incorporated during the expanded exploitation phase of the Aquila Optimizer, this mechanism broadens the search range of individual agents, improving global search capabilities.

4) Combined Random Reverse Learning and Simplex Method: This strategy optimizes individual search mechanisms, enabling the algorithm to effectively escape local optima.

A set of experiments with different parameter settings was conducted to investigate the performance of the proposed HSAO algorithm. The obtained results were systematically compared against several well-established algorithms, including the Levy-flight-based Whale Optimization Algorithm (LWOA), the standard Whale Optimization Algorithm (WOA), the Sparrow Search Algorithm (SSA), the Genetic Algorithm (GA), the Gravitational Search Algorithm (GSA), as well as the conventional Aquila Optimizer (AO). The results demonstrate that the HSAO algorithm achieves faster convergence and superior optimization performance while maintaining high stability across different evolutionary iterations.

Chapter 2 of this paper presents the mathematical model for the collaborative resource allocation problem among multiple MEC servers in an ultra-dense network environment. Chapter 3 offers an in-depth description of the core principles underlying the Aquila Optimizer algorithm. Chapter 4 focuses on the practical realization of the enhanced Aquila Optimizer, which integrates hybrid strategies. Chapter 5 presents simulation-based comparisons and analyses to assess the effectiveness of the proposed method. Finally, the paper concludes with a summary of the findings and provides insights into future research directions.

II. System Model

A. MEC server ultra-dense deployment model

In Figure 1, a MEC server is configured next to each base station, and the MEC server and the base station are denoted by the same symbols. In this paper, we consider that the resource co-allocation system for the ultra-dense deployment of edge servers involves a set of base stations, a set of MEC servers denoted by $M = \{1, 2, ..., m, ..., M\}$, a set of MSDs denoted by $I = \{1, 2, ..., i, ..., I\}$ and a set of channels denoted by $K = \{1, 2, ..., k, ..., K\}$. For ease of representation, the location numbers of all base stations are uniformly represented by the MEC server location numbers. Each MSD has a task to be performed, e.g., the task of the ith MSD is denoted as Q_i, which can be defined by a quaternion: $Q_i = (C_i, D_i, B_i, T_i^{max})$. Where C_i represents the total computational workload (measured in CPU cycles) required to complete the task, D_i and B_i correspond to the sizes of the input and output data, respectively. Additionally, T_i^{max} defines the latency constraint for Q_i. Q_i should be completed within time T_i^{max}. The task can be completed in time $[0, T_i^{max}]$.

B. Computational model

In this paper all the tasks Q_i in the mobile edge computing system can choose the task offloading mode according to the resources of MSD and MEC server. Therefore, in this paper, the matrix o is defined as the offloading decision, e.g., when the task Q_i chooses to be executed in the MEC server m mode through the channel k , then $o_{i,m}^k = 1$, otherwise $o_{i,m}^k = 0$; when $\sum_{i=1}^{I} o_{i,m}^k = 0$, the task Q_i chooses to be computed locally. The local computational resource assigned to task Q_i is represented by $f_i \in [0, f_i^{\max}]$, while the offloaded computational resource corresponding to task Q_i is denoted as $F_i \in [0, F_{\max}]$.

In local computing mode, the task Qi selects the computing time for processing on the MSD device as:

$$T_i^L = \frac{C_i}{f_i} \tag{1}$$

Analogous to local computing, if task Q_i is offloaded by the user to the MEC server, the associated computation time is calculated as:

$$T_i^S = \frac{C_i}{F_i} \tag{2}$$

When MSD chooses to perform the task Q_i under local execution, the computational energy consumption of MSD is:

$$E_i^L = \zeta (f_i)^2 C_i \tag{3}$$

Where $\zeta = 5 \times 10^{-27}$ is determined by the specific design of the MSD chip architecture.

For the task offloading mode, MSD uses OFDM technology to upload the data to the MEC server. Once the calculation is completed, the result is returned to the mobile user over the downlink. The uplink data rate of Q_i over channel k at MEC server m is expressed as:

$$R_i = W \log_2 \left(1 + \frac{p_i^t h_{i,m}^k}{w + \sum_{j \neq i} o_{j,m}^k p_j^t h_{j,m}^k}\right) \tag{4}$$

Here, W represents the channel bandwidth; p_i^t indicates the transmission power associated with mobile device i; w corresponds to the background noise power; and $h_{i,m}^k$ describes the channel gain when task Q_i is offloaded to MEC server m via channel k. During MSD communication, each MSD occupies one channel. The same channel can be multiplexed by multiple MSDs, and there is interference between MSDs multiplexing the same channel resource. $\sum_{j \neq i} o_{j,n} p_j^t h_{j,n}^k$ is the interference between mobile users in the same channel. Since the scenario in this paper is static, the uplink rate and downlink rate are symmetric.

The time required to transmit task Q_i when processed by the MEC server is formulated as:

$$T_i^t = \frac{D_i + B_i}{R_i} \tag{5}$$

In the context of MEC-based computation, the energy consumption associated with transmitting task Q_i arises from smart device i during the upload of input data and the download of computation results, which can be expressed as:

$$E_i^t = \frac{p_i^t D_i + p_i^r B_i}{R_i} \tag{6}$$

Where p_i^r indicates the ability of the mobile user i to receive the results.

In summary, the overall energy expenditure and the cumulative task latency are respectively formulated as:

$$E = \sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{i=1}^{I}(o_{i,m}^k E_i^t + ((1-o_{i,m}^k)E_i^T)) \quad (7)$$

$$T = \sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{i=1}^{I}(o_{i,m}^k (T_i^t + T_i^S) + ((1-o_{i,m}^k)T_i^L)) \quad (8)$$

The user cost is calculated as a weighted combination of the task execution delay and the MSD's total energy expenditure, given by:

$$J = \alpha T + (1-\alpha)E \quad (9)$$

In the equation, the weighting factor α represents the importance of the total task delay T in the total cost for users. In the simulation experiments conducted in this paper, α is set to 0.5. Under the conditions specified in this section, the edge computing system can selectively assist important users in completing computing tasks. Therefore, an important optimization metric in the problem of incomplete offloading is task completion, which can be expressed as:

$$P = \frac{V}{\sum_{i=1}^{I} V_i} \quad (10)$$

V is the preference of the edge computing service provider for performing user tasks and is the sum of the weights of all user tasks, which can be expressed as:

$$V = \sum_{k=1}^{K}\sum_{n=1}^{M+1}\sum_{i=1}^{I} o_{i,n}^k V_i \quad (11)$$

In Equation (11), $V_i \in (0,4)$ is a normally distributed random number.

C. Problem modeling

When the importance of different users in the edge system is known, the task completion can be indirectly expressed in terms of the quality of completion, and the optimization objective of the task is $min(J)\&max(V)$. In this section, function $Y3$ serves as a metric for evaluating the aggregate benefit of the task, and the overall optimization goal is defined as:

$$Y3 = \beta V - (1-\beta)J \quad (12)$$

In Equation (12), $\beta = [0,1]$ is the combined weight factor, which reflects the degree of importance of task completion in the overall ultra-dense network. As a result, the optimization task is reformulated into a minimization problem of $Y3$, which can be formally written as:

$$\begin{aligned}
\text{s.t. } & C_1 : f_i \le f_i^{\max} \\
& C_2 : \sum_{k=1}^{K} o_{i,m}^k \le 1 \\
& C_3 : f_i + F_i > 0, \forall o_{i,n}^k = 1 \\
& C_4 : \sum_{i=1}^{I}\sum_{m=1}^{M} F_i o_{i,m}^k \le F \\
& C_5 : T_i \le T_n^{\max} \\
& C_6 : \sum_{i=1}^{I} E_i \le E_{\max} \\
& C_7 : \sum_{m=1}^{M} o_{i,m}^k \le 1
\end{aligned} \quad (13)$$

C_1 ensures that the computing capacity utilized in the local execution mode remains within the device's resource threshold; C_2 ensures that each task can only select one channel for uploading; C_3 indicates that computational resources must be allocated for each task; C_4 ensures that the computing capacity utilized in the MEC computation mode remains within the resource limit of the MEC server; C_5 mandates that each task adheres to its latency limit; C_6 stipulates that the energy consumption of each task remains within bounds; C_7 imposes a selection constraint, allowing each task to offload to only one MEC server.

III. Basic Aquila Optimizer algorithm

The Aquila Optimizer (AO) [14], formulated by Laith Abualigah et al. in 2021, draws inspiration from the predatory patterns of aquila birds for its search mechanism. The algorithm's procedural framework is outlined below.

A. Expanded exploration

The vector X_t indicates the aquila's location in the search space at iteration t, and is utilized to evaluate its corresponding fitness value. Meanwhile, when the evolution number is t, the whole population of aquila maintains a global optimal position X_t^*, and other individuals scout the sky and move towards the optimal position X_t^*. The mathematical model is defined as:

$$X_{t+1} = X_t^* \times (1 - \frac{t}{T}) + (X_t^m - X_t^* \times rand) \quad (14)$$

In Equation (14), X_{t+1} indicates the updated solution at iteration $t+1$, while $rand$ represents a random scalar drawn from a uniform distribution over $[0,1]$. T specifies the total number of iterations. The term X_t^m signifies the centroid position of the current solution population at iteration t, as derived from Equation (15):

$$X_t^m = \frac{1}{N}\sum_{i=1}^{N} X_t^i, \forall j = 1,2,...,Dim \quad (15)$$

In Equation (15), Dim specifies the problem dimensionality, and N refers to the population size, indicating the total number of candidate solutions.

B. Narrowing the scope of exploration

The aquila surrounds its prey above the sky and prepares to attack, as defined by the mathematical model:

$$X_{t+1} = X_t^* \times Levy(D) + X_t^R + (y - x) \times rand \quad (16)$$

where D denotes the dimensional space, and $Levy(D)$ represents the Lévy flight distribution, as defined in Equation (17). At the t-th iteration, X_t^R corresponds to a solution randomly selected within the interval $[1, N]$.

$$Levy(D) = s \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \quad (17)$$

s is a predefined constant set to 0.01, while u and v are uniformly distributed random variables within the interval $[0, 1]$. The parameter σ is determined according to Equation (18).

$$\sigma = \frac{\Gamma(1+\beta) \times \sin(\frac{\Pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times 2^{(\frac{\beta-1}{2})} \times \beta} \quad (18)$$

β is a predefined constant set to 1.5. In Equation (16), the variables y and x are employed to model the spiral trajectory during the search process, which is formulated as follows:

$$y = r \times (\theta) \quad (19)$$

$$x = r \times (\theta) \quad (20)$$

$$r = r_1 + U \times D_1 \quad (21)$$

$$\theta = -\omega \times D_1 + \theta_1 \quad (22)$$

$$\theta_1 = \frac{3 \times \Pi}{2} \quad (23)$$

r_1 is an integer selected within the range $[1, 20]$ to define the number of search iterations, while U is a predefined small constant set to 0.00565. The variable D_1 represents an integer value ranging from 1 to the dimensionality of the search space Dim, and ω denotes a minor constant fixed at 0.005.

C. Expanded development

The mathematical definition of an aquila performing a vertical landing process, observing prey reactions and preparing to attack the prey:

$$X_{t+1} = (X_t^* - X_t^M) \times \alpha - rand + ((UB - LB) \times rand + LB) \times \delta \quad (24)$$

In Equation (24), $rand$ refers to a random variable sampled from a uniform distribution over $[0, 1]$. In this work, α and δ are both assigned a value of 0.1. LB and UB indicate the positional lower and upper limits, respectively.

D. Reducing the scope of development

Finally, the act of a skyhawk attacking its prey is mathematically represented as:

$$X_{t+1} = X_t^* \times QF + (G_1 \times X_t \times rand) - G_2 \times Levy(D) + rand \times G_1 \quad (25)$$

QF refers to the mass function that regulates the balance between exploration and exploitation strategies, as defined in Equation (26). G_1 characterizes the diverse movement patterns of the AO, determined by Equation (27). G_2 corresponds to a sequence of diminishing values ranging from 2 to 0, computed using Equation (28).

$$QF(t) = t^{\frac{2 \times rand - 1}{(1-T)^2}} \quad (26)$$

$$G_1 = 2 \times rand - 1 \quad (27)$$

$$G_2 = 2 \times (1 - \frac{t}{T}) \quad (28)$$

IV. Hybrid Strategy-based Aquila Optimizer Improvement Algorithm

With the objective of optimizing collaborative resource allocation across multiple MEC servers in ultra-dense networks, this paper improves the AO algorithm and proposes Hybrid Strategy-based Aquila Optimizer improvement (HSAO). The algorithm optimizes the objective function, reduces the solution space using a candidate mode pruning method, and enhances population randomness by introducing strategies such as Tent chaotic mapping for population initialization, random spiral update mechanism, random reverse learning, and the simplex method. The pseudo-code for the HSAO algorithm is as follows.

A. Optimizing the objective function

Since the original AO algorithm is used for unconstrained optimization, this paper needs to use an effective constraint handling technique to solve the constraint problem. In order to handle the constraints C_2, constraint C_3 and constraint C_6, the decision variable X is used to represent the current resource scheduling of all the tasks, which consists of the offloading MEC server number m, the offloading channel k, the MSD computational resources f and the MEC computational resources F. Given I MSDs in the system, vector x has a dimensionality of $4 \times I$. The final vector X is encoded as $X = \{m_1, m_2, ..., m_I, k_1, k_2, ..., k_I, f_1, f_2, ..., f_I, F_1, F_2, ..., F_I\}$. The algorithm is optimized through iterative evolution, and the final output corresponds to the task offloading strategy that minimizes the overall system overhead. Since the original AO algorithm is used for continuous problem optimization, this paper needs to use downward rounding to transform continuous variables into discrete variables, and the vector range becomes $X^{\min} < X < X^{\max} + 1 - \epsilon$. To prevent $1 - \epsilon = 1$, ϵ is taken as the smallest real number.

In this paper, the constraint C_1, constraint C_4 and constraint C_5 are transformed into the corresponding penalty functions, at which time the optimization objective is

Algorithm 1 Hybrid Strategy Aquila Optimizer(HSAO)

Input:
T,N,LB,UB,D,$t = 1$

Output:
$BestValue$

Pruning candidate decision by Algorithm2;
Initialize population using Tent chaotic mapping by Eq (40);
Calculate the parameters by Eq (26,27,28);
while $t \leq T$ **do**
  if $t \leq \frac{T}{3}$ **then**
    if $rand() < 0.5$ **then**
      Update the location of individuals by Eq (17);
    else
      Update the location of individuals by Eq (16);
    end if
  else
    if $rand() < 0.5$ **then**
      Updating individual locations using a randomized spiral update mechanism by Eq (41);
    else
      Update the location of individuals by Eq (26);
    end if
  end if
  Evaluate the objective function for each individual using Eq (12);
  Monomorphic method for updating the location of individuals by Algorithm3;
  Boundary checking and updating the value of BestValue;
  $t = t + 1$;
end while

transformed into the sum of $Y3$ and the value of the penalty function:

$$min(Y3' = Y3 + \theta) \tag{29}$$

The aggregated value of the penalty functions is denoted by θ, where o denotes the candidate solution to the problem. The computation of θ is given by:

$$\theta = \sum_{g=1}^{3} max\{0, p_g(X)\} \tag{30}$$

The function $p_g(X)$ denotes the constraint penalty term, formulated as follows:

$$p_1(X) = (\sum_{m=1}^{M} o_{i,m}^{k} F_i - F_i^{max})^2 \tag{31}$$

$$p_2(X) = (max\{T_i^L, T_i^C\} - T_i^{max})^2 \tag{32}$$

$$p_3(X) = (E_{max} - \sum_{i=1}^{I} E_i)^2 \tag{33}$$

B. Candidate model pruning

For the selection of server offloading locations in the task offloading matrix, each task has $(M + 1)$ task offloading candidate patterns, the local resource candidate pattern threshold denoted as $f_{max} = \{f_1^{max}, f_2^{max}, ..., f_i^{max}, ..., f_I^{max}\}$ and the offloading resource candidate pattern threshold denoted as F_{max}, the number of task Q_i offloading decisions is $I(M+1)$, the number of local resource allocation decisions is $I \times f_i^{max}, \forall i \in I$, and the number of MEC server offloading decisions is $I \times F_{max}$. In practice, a significant portion of the generated solutions are infeasible. To address this challenge, infeasible candidate patterns are filtered out for each task at the initial stage of the algorithm. If the local offloading mode is 1, there are 3 MEC servers and 3 users, and its candidate mode is shown in Figure 2, green indicating the computational resource allocation range, before the candidate mode pruning, tasks Q_1, Q_2 and Q_3 all MEC servers as well as local resources belong to the range of the decision variables, after the candidate mode pruning, the task Q_1 can not select the MEC server in position 3 for offloading and the corresponding allocated. The task Q_2 can only select the local computing mode and the corresponding allocated computing resources cannot be lower than the minimum value, while the task Q_3 cannot satisfy all the computing modes. The proposed strategy contributes to a substantial reduction in task offloading decisions.

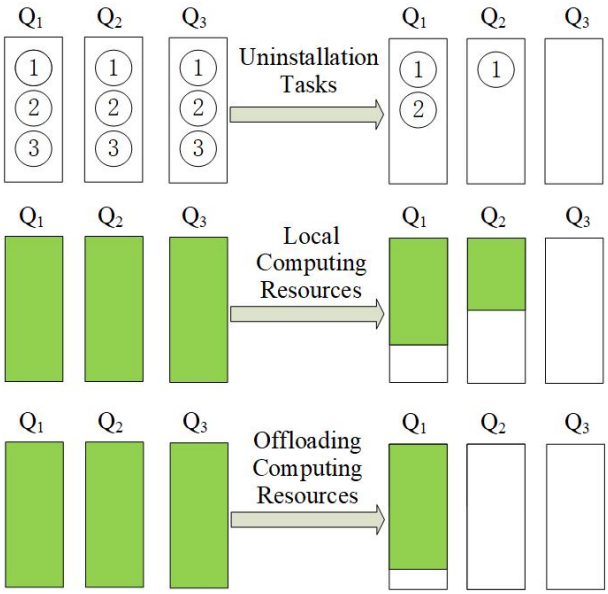


Fig. 2: Comparison of Decision Variable Lengths for Candidate Models Improved to Pruned Candidate Models

The conditions for judging the feasibility of candidate models are given below: If the minimum computational resource demand of a task under a specific mode cannot be fulfilled, that mode will be considered an invalid option for executing the task. The minimum computational resource requirements based on C_5, Q_i in the MEC server offload mode and in the local computing mode can be expressed as follows:

$$F_i^{min} = \frac{C_i}{T_i^{max} - \frac{D_i}{R_i} + \frac{B_i}{R_i}} \tag{34}$$

$$f_i^{min} = \frac{C_i}{T_i^{max}} \tag{35}$$

Algorithm 2 Candidate Pattern Pruning (CPP)

Input:
Q_i, M, f_i^{max}, F_{max}, F_i^{min}

Output:
X_i^{lim}

Initialize $m_i^{min} = r_i^{min} = F_i'^{min} = 0$, $m_i^{max} = M$, $r_i^{max} = f_i^{max}$, $F_i^{max} = F_{max}$;
if $F_{max} \geq F_i^{min}$ **and** $r_i^{max} \geq F_i^{min}$ **then**
    Update F'^{min}_i and r_i^{min} by Eq. (34);
    Set $m_i^{max} = M$;
else if $F_{max} \geq F_i^{min}$ **and** $r_i^{max} < F_i^{min}$ **then**
    Set $F_i^{max} = 0$, $m_i^{max} = 0$;
    Update r_i^{min} by Eq. (34);
else if $F_{max} < F_i^{min}$ **and** $r_i^{max} \geq F_i^{min}$ **then**
    Set $m_i^{max} = 0$, $F_i^{max} = 0$;
    Update r_i^{min} by Eq. (34);
else
    Set $F_i^{max} = m_i^{max} = r_i^{min} = 0$;
end if
Construct $X_i^{lim} = \{m_i^{min}, r_i^{min}, F'^{min}_i, m_i^{max}, r_i^{max}, F_i^{max}\}$;
return X_i^{lim};

In accordance with constraints C_3 through C_5 and C_7, a candidate model is deemed acceptable if it meets the criteria outlined below:

$$f_i^{\min} < f_i^{\max}, \forall i \in I \tag{36}$$

$$F_i^{min} < F_i^{max}, \forall i \in I \tag{37}$$

$$F_i^{min} + f_i^{min} > 0, \forall i \in I \tag{38}$$

For the computational offloading mode, since it is not possible to predict in advance which tasks the mode performs during the initialization phase, it is not possible to obtain it by eliminating channel interference, and it is not possible to obtain the data rate, which leads to the unavailability of the minimum computational resource requirement in Equation (4). In this case, Equation (5) cannot be directly used to determine the feasibility of the candidate mode. Under the assumption that a single task is handled by the MEC server, the upper bound of the uplink data rate, denoted as $\overline{R_i}$, is determined by Equation (39).

$$\overline{R_i} = W \log_2(1 + \frac{p_i^t h_{i,m}^k}{w}) \tag{39}$$

The maximum achievable downlink data rate $\overline{R_i}$ can be derived analogously. Substituting Equation (39) into Equation (34) yields a lower bound on the computational resource requirements. Since the infeasible patterns have been pruned, it facilitates the quick construction of feasible offloading decisions.

C. Tent chaos to initialize populations

In recent years, many scholars have introduced Tent chaotic mapping into heuristic optimization algorithms such as Gray Wolf Optimization Algorithm [15], Sparrow Search Algorithm [16] and Vulture Algorithm [17] to enhance the searching ability of bionic learning algorithms. The mean dot plots and mean histograms of the original random population and the Tent chaotic mapping population with a population size of 1000 under 100 independent experiments using the rand function in Matlab are shown in Figure 3. It can be inferred that Tent chaotic mapping significantly enhances population diversity and accelerates the algorithm's convergence during the initial phase.

It should be noted that the Tent chaotic mapping system is short-periodic when $\rho = 0.5$ and the initial value of the system x_t cannot be the same as the system parameter ρ, otherwise it will be periodic. In this section, Tent chaotic mapping is chosen instead to enhance the population diversity of the proposed HSAO with the following equations:

$$X_{t+1} = \begin{cases} X_t/\rho & , X_t \in [0, \rho] \\ (1 - X_t)/(1 - \rho) & , X_t \in [\rho, 1] \end{cases} \tag{40}$$

D. Randomized spiral update mechanism

In the ocean, whales like to prey on krill colonies or small fish close to the water surface by generating unique bubbles along a circular or "9"-shaped path, a foraging behavior known as spiral bubble attack. From the way the position is updated in the expanded development phase of the AO algorithm, it is clear that the difficulty of the algorithm to escape the local optimum gradually increases when the number of evolutions gradually increases. Inspired by the whale spiral exploration model, randomized spiral exploration is introduced in the expanded exploitation phase, which further extends the ability of individuals to explore unknown regions.

In this paper, we replace the average position update in the AO algorithm Equation (25) with the spiral exploration formula of the bubble network attack in WOA, and add a random factor $r * D_{rand}$ into the whale spiral exploration formula, which ensures the influence of the optimal position of the algorithm and enhances the ability of the individual to learn the information carried by the other individuals among the populations, and the stochastic spiral updating mechanism is formulated as follows:

$$X_{t+1} = X_t + D' e^{bl} \cos(2\pi l) + r * D_{rand} \tag{41}$$

$$D' = |X_t^* - X_t| \tag{42}$$

$$D_{rand} = |X_{rand} - X_t| \tag{43}$$

b is a constant that defines the shape of the logarithmic spiral, l and r are random numbers generated in the interval $[-1, 1]$.

E. Mixed variants based on stochastic inverse learning and simplex methods

The Monomorphic Method generates an individual with better quality by putting the individual through symmetry, expansion, exocontraction and endocontraction operations as shown in Figure 4. The ability of the monomorphic method to effectively improve the optimization of bionic learning algorithms has been demonstrated, such as the Whale Optimization Algorithm [18], the Ant-Lion Optimization Algorithm [19], and the Harris Hawk Optimization Algorithm [20].

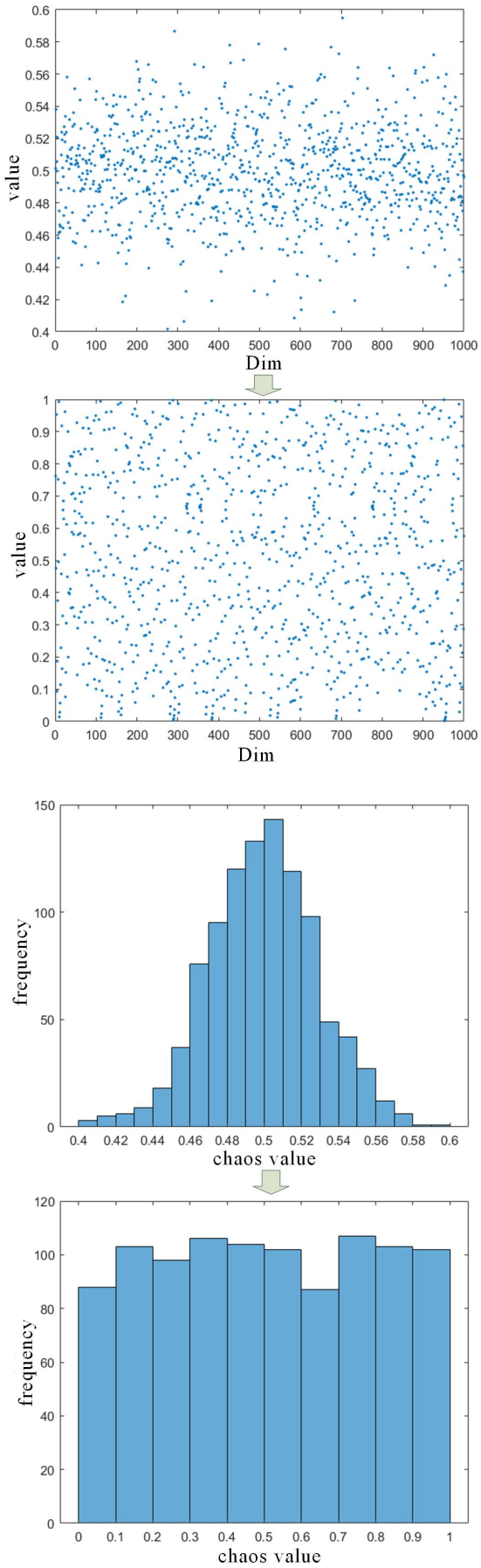


Fig. 3: Comparison of 100 times average scatter plots and mean distribution histograms of populations generated by random number improvement for Tent chaotic mapping

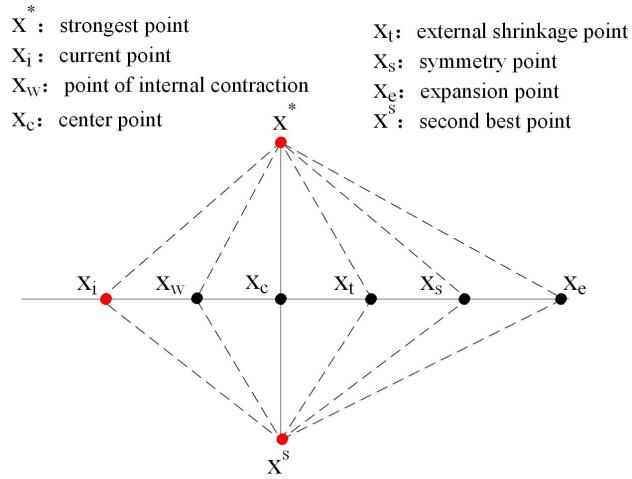


Fig. 4: Simplex method

Reverse learning [21] is the process of expanding the search by computing the inverse solution of the current solution in the search space. There have been bionic learning algorithms having demonstrated that the reverse learning approach can enhance the search capability, such as the Gray Wolf Optimization Algorithm [22], the Raven Search Algorithm [23], and the Aquila Optimization [24] algorithms. Based on the introduction of stochastic factors in the inverse learning strategy, the search range of population individuals is extended, and the ability of population individuals to jump out of the local extremes is enhanced. Then the stochastic backward learning and simplex method are combined, and the specific implementation steps of its mixed variational strategy (MVS) are as follows.

V. Experimental Results And Analysis

A. CEC2017 Benchmark Test Functions

This study evaluates the optimization performance of the HSAO algorithm using the CEC2017 benchmark test suite. This test suite consists of a series of nonlinear functions with high dimensionality and complexity, containing numerous local extrema and saddle points within their parameter space, which pose significant challenges to the solving capabilities of optimization algorithms. The test suite is divided into four categories: unimodal functions ($f_1(x)$ - $f_3(x)$), simple multimodal functions ($f_4(x)$ - $f_{10}(x)$), hybrid functions ($f_{11}(x)$ - $f_{20}(x)$), and composite functions ($f_{21}(x)$ - $f_{30}(x)$). Since all 30 benchmark functions in this suite have been subjected to translation and rotation transformations, their theoretical optimal values are no longer zero but exhibit a stepwise distribution ranging from 100 to 3000, with an increment of 100. Additionally, function $f_2(x)$ has been removed from the CEC2017 test suite due to significant numerical instability observed in high-dimensional spaces.

1) Optimization accuracy analysis: To comprehensively evaluate the algorithm's performance, this subsection presents a comparative experimental analysis of the HSAO algorithm with the AO algorithm, Gravitational Search Algorithm (GSA), and Whale Optimization Algorithm (WOA). To ensure fairness and objectivity in the algorithm performance comparison, all four algorithms were tested on the same hardware and software platform: Windows

Algorithm 3 Mixed Variational Strategy (MVS)

Input:
A, X, F_x, X^*, X^S

Output:
X^*, Fx^*

Initialize population: $X = \{X_1, X_2, ..., X_a, ..., X_A\}$, $Fx = \{Fx_1, Fx_2, ..., Fx_a, ..., Fx_A\}$;
Initialize parameter: a;
for $a = 1 : A$ **do**
    Calculate symmetry point X_r, center point X_c, expansion point X_e,
        external constriction point X_t, internal contraction point X_w;
    if $Fx_r < Fx^*$ **and** $Fx_e < Fx^*$ **then**
        Update $X_a \leftarrow X_e$, $Fx_a \leftarrow Fx_e$;
    else if $Fx_r < Fx^*$ **and** $Fx_e > Fx^*$ **and** $Fx_a > Fx_t$ **then**
        Update $X_a \leftarrow X_t$, $Fx_a \leftarrow Fx_t$;
    end if
    if $Fx_r < Fx_a$ **and** $Fx_r > Fx^*$ **and** $Fx_w < Fx^*$ **then**
        Update $X_a \leftarrow X_w$, $Fx_a \leftarrow Fx_w$;
    end if
    Apply reverse learning: Randomize current point X_a;
    if $Fx^* > Fx_a$ **then**
        Update $X_a \leftarrow X^*$, $Fx_a \leftarrow Fx^*$;
    end if
end for
return Fx^*, X^*;

11 operating system, Intel i7-13650 CPU, and MATLAB R2021a environment. All algorithms used the same population size N, spatial dimension D, and maximum number of iterations T, with values of $N = 30$, $D = 10/50/100$, and $T = 500$. Each algorithm was independently run 10 times.

The test results of 15 benchmark functions are presented, including unimodal functions f_1 and f_3, simple multimodal functions f_4 and f_7, hybrid functions f_{11}, f_{13}, f_{15}, f_{17}, f_{19}, and f_{20}, as well as composition functions f_{21}, f_{25}, f_{27}, f_{29}, and f_{30}. The results of other test functions are similar and are not elaborated here for brevity.

Tables I,II,III summarize the best values, mean values, and variances of the HSAO, AO, GSA, and WOA algorithms in function optimization tests across 10-dimensional, 50-dimensional, and 100-dimensional search spaces. The experimental results demonstrate that the HSAO algorithm outperforms the other three algorithms across all dimensions, particularly in terms of achieving optimal values, mean values, and variances. This highlights its strong global search capability and solution stability.

In the 10-dimensional problem, the HSAO algorithm performs exceptionally well, particularly in obtaining optimal values with a clear advantage. For instance, the optimal solution for the F1 function is 1.01×10^5, while AO, WOA, and GSA achieve solutions of 4.84×10^6, 5.53×10^6, and 1.60×10^2, respectively. HSAO significantly outperforms

TABLE I: Performance comparison of HSAO, AO, WOA, and GSA algorithms on benchmark functions(D=10)

Function	Metric	HSAO	AO	WOA	GSA
f_1	Best	1.01E+05	4.84E+06	5.53E+06	1.60E+02
	Mean	4.21E+05	1.80E+07	5.77E+07	7.31E+02
	Std	2.89E+05	1.69E+07	6.22E+07	8.90E+02
f_3	Best	3.03E+02	8.57E+02	9.90E+02	6.55E+03
	Mean	3.07E+02	2.11E+03	6.54E+03	1.44E+04
	Std	4.19E+00	7.36E+02	6.45E+03	3.97E+03
f_4	Best	4.00E+02	4.04E+02	4.05E+02	4.06E+02
	Mean	4.05E+02	4.25E+02	4.55E+02	4.13E+02
	Std	2.54E+00	2.21E+01	6.04E+01	1.93E+01
f_7	Best	7.21E+02	7.34E+02	7.69E+02	7.24E+02
	Mean	7.50E+02	7.58E+02	7.97E+02	7.33E+02
	Std	7.83E+00	2.25E+01	3.18E+01	1.05E+01
f_{11}	Best	1.11E+03	1.13E+03	1.15E+03	1.14E+03
	Mean	1.18E+03	1.22E+03	1.24E+03	1.43E+03
	Std	4.63E+01	9.10E+01	9.81E+01	3.36E+02
f_{13}	Best	1.91E+03	7.73E+03	2.11E+03	9.69E+03
	Mean	4.80E+03	1.89E+04	1.60E+04	1.23E+04
	Std	1.72E+03	1.04E+04	1.43E+04	3.16E+03
f_{15}	Best	1.63E+03	2.10E+03	4.55E+03	1.03E+04
	Mean	2.40E+03	6.14E+03	1.46E+04	1.99E+04
	Std	1.26E+03	2.51E+03	9.24E+03	8.11E+03
f_{17}	Best	1.74E+03	1.76E+03	1.77E+03	1.75E+03
	Mean	1.78E+03	1.79E+03	1.83E+03	1.96E+03
	Std	2.31E+01	2.91E+01	6.26E+01	1.15E+02
f_{19}	Best	1.94E+03	2.12E+03	6.94E+03	5.35E+04
	Mean	3.22E+03	1.07E+04	4.18E+04	1.70E+05
	Std	3.23E+03	1.04E+04	5.89E+04	1.02E+05
f_{20}	Best	2.05E+03	2.07E+03	2.13E+03	2.20E+03
	Mean	2.13E+03	2.16E+03	2.21E+03	2.34E+03
	Std	5.81E+01	7.01E+01	8.46E+01	9.70E+01
f_{21}	Best	2.20E+03	2.22E+03	2.34E+03	2.39E+03
	Mean	2.24E+03	2.32E+03	2.35E+03	2.40E+03
	Std	5.15E+01	5.80E+01	5.28E+01	5.37E+01
f_{25}	Best	2.62E+03	2.90E+03	2.95E+03	2.94E+03
	Mean	2.91E+03	2.93E+03	2.96E+03	2.94E+03
	Std	2.00E+01	2.45E+01	4.17E+01	2.12E+01
f_{27}	Best	3.09E+03	3.10E+03	3.10E+03	3.26E+03
	Mean	3.10E+03	3.11E+03	3.16E+03	3.37E+03
	Std	2.76E+00	1.13E+01	5.12E+01	9.79E+01
f_{29}	Best	3.17E+03	3.20E+03	3.25E+03	3.38E+03
	Mean	3.24E+03	3.28E+03	3.41E+03	3.57E+03
	Std	3.20E+01	5.16E+01	1.00E+02	2.56E+02
f_{30}	Best	7.25E+03	7.95E+04	1.58E+05	5.52E+05
	Mean	4.41E+05	1.69E+06	1.50E+06	1.45E+06
	Std	6.37E+05	1.55E+06	1.79E+06	1.13E+06

the other algorithms in terms of solution accuracy for this function. Meanwhile, the standard deviation of HSAO is also relatively small, indicating good stability in this dimension. Regarding the mean value, HSAO's mean is noticeably better than that of the other algorithms, suggesting that it not only converges quickly to the global optimum but also maintains superior performance across multiple runs.

As the dimensionality increases, the complexity of the optimization problem also rises. In the case of 50 dimensions, the HSAO algorithm still performs excellently, achieving near-optimal solutions across multiple benchmark functions. For example, for the F3 function, the optimal solution for HSAO is 3.7×10^8, while AO's optimal solution is 1.26×10^{10}, WOA is 1.14×10^{10}, and GSA is 4.71×10^{10}, clearly showing that HSAO outperforms the other algorithms. Additionally, HSAO maintains lower mean values and standard deviations across several functions, which effectively balance accuracy and stability, ensuring better solutions while avoiding excessive local search.

In high-dimensional problems with 100 dimensions, the accuracy and stability of algorithms face greater challenges.

TABLE II: Performance comparison of HSAO, AO, WOA, and GSA algorithms on benchmark functions(D=50)

Function	Metric	HSAO	AO	WOA	GSA
f_1	Best	3.7E+08	1.26E+10	1.14E+10	4.71E+10
	Std	2.1E+08	4.94E+09	2.98E+09	6.97E+09
	Mean	7.4E+08	2.07E+10	1.86E+10	5.82E+10
f_3	Best	113406.3	237451.2	206701.2	178372.1
	Std	14117.32	56630.9	102328.7	17376.86
	Mean	135096.4	331139.8	291963.5	198690.8
f_4	Best	745.3046	2293.959	2728.272	11941.2
	Std	81.60256	1216.698	1065.703	2172.399
	Mean	832.2251	3901.462	4241.038	15742.17
f_7	Best	1370.019	1473.212	1704.75	1437.456
	Std	104.6451	123.0221	153.8348	133.9941
	Mean	1576.715	1596.766	1895.744	1586.935
f_{11}	Best	1639.596	5103.627	5045.148	22641.52
	Std	158.3358	1544.635	1866.033	2426.693
	Mean	1847.867	7291.718	8506.888	25884.43
f_{13}	Best	2634000	2.36E+08	1.23E+08	1.1E+10
	Std	5481636	7.25E+08	4.51E+08	4.54E+09
	Mean	10196677	1.46E+09	5.76E+08	1.73E+10
f_{15}	Best	4.69E+05	1.01E+06	1.04E+07	8.15E+06
	Std	4.12E+05	2.79E+07	3.79E+07	2.72E+08
	Mean	1.16E+06	1.77E+07	5.60E+07	2.70E+08
f_{17}	Best	2.73E+03	3.71E+03	3.81E+03	3.22E+03
	Std	3.21E+02	4.58E+02	5.21E+02	5.58E+02
	Mean	3.68E+03	3.91E+03	4.36E+03	3.86E+03
f_{19}	Best	2.76E+05	2.51E+06	7.25E+06	4.59E+05
	Std	3.50E+06	4.72E+06	4.77E+06	4.36E+08
	Mean	3.19E+06	6.05E+06	1.33E+07	2.25E+08
f_{20}	Best	2.71E+03	3.08E+03	3.40E+03	3.26E+03
	Std	1.84E+02	5.04E+02	4.21E+02	4.50E+02
	Mean	3.31E+03	3.48E+03	3.99E+03	3.80E+03
f_{21}	Best	2.65E+03	2.70E+03	2.93E+03	2.87E+03
	Std	4.69E+01	4.99E+01	8.62E+01	7.30E+01
	Mean	2.77E+03	2.79E+03	3.11E+03	3.03E+03
f_{25}	Best	3131.77	3931.749	4891.661	7945.652
	Std	85.48339	554.2433	415.2051	651.0807
	Mean	3280.317	4721.024	5364.05	8795.344
f_{27}	Best	3590.163	3962.695	4165.663	8131.507
	Std	176.8759	177.005	283.8162	378.932
	Mean	3823.754	4222.418	4521.682	8828.147
f_{29}	Best	5667.312	6477.441	6675.149	17287.29
	Std	790.1246	1648.844	1807.265	10743.45
	Mean	6578.101	8529.382	9693.203	28013.75
f_{30}	Best	47916483	1.09E+08	1.78E+08	4.89E+08
	Std	28462130	80339868	1.16E+08	6.28E+08
	Mean	97878947	2.1E+08	2.99E+08	1.5E+09

TABLE III: Performance comparison of HSAO, AO, WOA, and GSA algorithms on benchmark functions(D=100)

Function	Metric	HSAO	AO	WOA	GSA
f_1	Best	1.13E+10	8.26E+10	8.77E+10	1.84E+11
	Mean	1.50E+10	9.42E+10	1.10E+11	2.09E+11
	Std	3.57E+09	1.28E+10	1.09E+10	1.96E+10
f_3	Best	2.62E+05	3.45E+05	8.03E+05	3.52E+05
	Mean	3.11E+05	3.55E+05	9.07E+05	3.74E+05
	Std	1.36E+04	1.52E+04	2.72E+05	2.36E+04
f_4	Best	2.44E+03	1.79E+04	1.46E+04	5.36E+04
	Mean	2.99E+03	2.06E+04	2.22E+04	6.66E+04
	Std	5.48E+02	1.44E+03	4.30E+03	8.34E+03
f_7	Best	2.85E+03	3.26E+03	3.65E+03	3.06E+03
	Mean	3.22E+03	3.57E+03	3.85E+03	3.37E+03
	Std	1.52E+02	1.88E+02	1.84E+02	2.51E+02
f_{11}	Best	7.90E+04	2.83E+05	1.14E+05	1.71E+05
	Mean	9.67E+04	3.34E+05	2.72E+05	1.98E+05
	Std	1.33E+04	4.27E+04	1.68E+05	2.09E+04
f_{13}	Best	1.67E+07	1.24E+09	1.21E+09	2.56E+10
	Mean	2.94E+07	3.05E+09	3.39E+09	3.13E+10
	Std	8.97E+06	1.05E+09	1.20E+09	5.67E+09
f_{15}	Best	3.03E+06	7.69E+07	1.33E+08	9.86E+09
	Mean	6.19E+06	5.74E+08	4.47E+08	1.44E+10
	Std	2.92E+06	4.69E+08	3.23E+08	2.37E+09
f_{17}	Best	5.33E+03	9.95E+03	9.29E+03	1.47E+06
	Mean	6.69E+03	1.94E+04	3.75E+04	3.27E+06
	Std	5.44E+02	7.61E+03	2.21E+04	1.56E+06
f_{19}	Best	4.42E+06	2.41E+08	1.88E+08	8.63E+09
	Mean	2.67E+07	5.83E+08	4.19E+08	1.45E+10
	Std	2.01E+07	2.80E+08	2.00E+08	3.62E+09
f_{20}	Best	5.23E+03	5.48E+03	7.14E+03	5.47E+03
	Mean	6.24E+03	6.33E+03	7.66E+03	6.43E+03
	Std	3.64E+02	4.91E+02	6.26E+02	7.94E+02
f_{21}	Best	3.69E+03	4.08E+03	4.23E+03	5.21E+03
	Mean	4.42E+03	4.51E+03	4.52E+03	5.55E+03
	Std	1.98E+02	2.89E+02	2.63E+02	3.05E+02
f_{25}	Best	4.50E+03	8.21E+03	9.35E+03	1.68E+04
	Mean	4.89E+03	9.76E+03	1.03E+04	2.08E+04
	Std	2.75E+02	9.72E+02	7.96E+02	2.33E+03
f_{27}	Best	4.10E+03	6.34E+03	4.95E+03	1.32E+04
	Mean	4.46E+03	7.42E+03	6.32E+03	1.60E+04
	Std	2.95E+02	1.00E+03	1.09E+03	1.76E+03
f_{29}	Best	9.62E+03	1.31E+04	1.79E+04	1.19E+05
	Mean	1.22E+04	1.77E+04	2.31E+04	2.69E+05
	Std	1.50E+03	3.30E+03	4.59E+03	1.28E+05
f_{30}	Best	1.62E+08	1.44E+09	1.59E+09	2.57E+10
	Mean	2.67E+08	4.08E+09	2.60E+09	3.01E+10
	Std	8.65E+07	1.48E+09	7.62E+08	3.58E+09

However, the HSAO algorithm still demonstrates outstanding performance in this dimension. Taking the F1 function as an example, the optimal solution of HSAO is 1.13×10^{10}, which is clearly better than the other three algorithms (AO: 8.26×10^{10}, WOA: 8.77×10^{10}, GSA: 1.84×10^{11}). Despite the increased complexity due to the higher dimensionality, HSAO still provides lower mean values and standard deviations, indicating that its stability and accuracy in high-dimensional problems remain strong.

By analyzing the experimental results for different dimensions, it can be observed that the HSAO algorithm exhibits strong optimization capabilities across different dimensions, especially in terms of accuracy and stability, which far exceed those of AO, WOA, and GSA. Whether in low-dimensional or high-dimensional problems, HSAO effectively avoids local optima, ensures convergence to better solutions, and demonstrates strong global search capabilities, making it suitable for complex engineering optimization problems.

2) Convergence curve analysis: In solving optimization problems, the convergence and stability of an algorithm are key factors that determine the quality and efficiency of the solution. This section compares the HSAO algorithm with the AO, GA, WOA, and GSA algorithms. To provide a more intuitive comparison of the performance of the five algorithms, Figure 5 shows the convergence curves for the 15 benchmark functions mentioned above. From the convergence behavior, it is clear that the HSAO algorithm demonstrates significant superiority in most test functions, particularly in terms of early convergence speed and final solution accuracy.

Specifically, on several benchmark functions (e.g., f_1, f_3, f_4, f_{19}, etc.), the HSAO algorithm is able to rapidly reduce the objective function value within fewer iterations and maintain a stable decreasing trend, ultimately converging to a lower optimal solution. This indicates that HSAO has strong global search capability and can effectively avoid getting trapped in local optima. In contrast, the AO, GA, WOA, and GSA algorithms exhibit slower convergence speeds and higher final convergence values on these functions, indicating that they are more prone to getting stuck in local optima when dealing with complex optimization problems.

Especially on high-difficulty functions (e.g., f_{15}, f_{17}, and f_{30}), the performance of the HSAO algorithm is more pronounced. Compared to other algorithms, HSAO not only converges faster on these functions but also finds better solutions, indicating its stronger solving ability and stability when tackling complex high-dimensional optimization problems. Although AO, GA, WOA, and GSA also make certain progress in the early stages, their convergence process is more fluctuating, and especially in later iterations, they tend to experience stagnation or rebound.

From the convergence curves, it is evident that the HSAO algorithm outperforms AO, GA, WOA, and GSA on multiple benchmark functions. It can find better solutions within fewer iterations and maintain lower fluctuations during the solution process. These results demonstrate that the HSAO algorithm not only possesses powerful global exploration capability but also maintains high stability throughout the solving process, showcasing significant performance advantages.

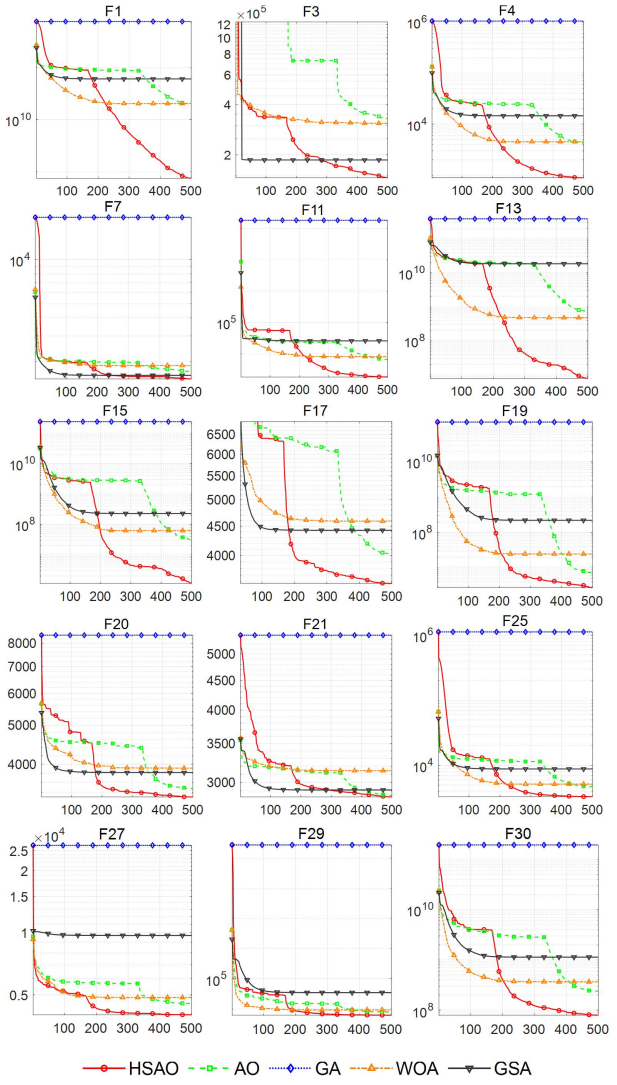


Fig. 5: Convergence performance analysis of the five algorithms on diverse test functions

3) Wilcoxon rank-sum test statistical analysis: The Wilcoxon rank-sum test, a widely used non-parametric statistical approach, is employed to determine whether the performance differences between two optimization algorithms are statistically meaningful. It functions as a crucial method for verifying the validity of algorithmic enhancements. To assess whether the HSAO algorithm exhibits superior optimization effectiveness compared to other benchmark algorithms, this study utilizes the Wilcoxon rank-sum test for significance testing. Table V reports the corresponding test results between HSAO and the competing methods. In this table, the symbol "+" denotes that HSAO achieves statistically superior optimization results, "-" indicates inferior performance relative to the compared algorithm, while "=" signifies no statistically discernible difference or equivalent performance. A p-value below 0.05 leads to the rejection of the null hypothesis, confirming the existence of a significant performance gap between the algorithms.

TABLE IV: Wilcoxon rank-sum test p-value between HSAO and comparison algorithms

Function	HSAO vs AO p-value win	HSAO vs GA p-value win	HSAO vs WOA p-value win	HSAO vs GSA p-value win
F1	3.04e-34 +	1.19e-187 +	2.40e-06 +	3.92e-18 +
F3	9.36e-145 +	1.15e-187 +	1.73e-94 +	1.39e-07 +
F4	8.75e-24 +	1.20e-187 +	5.92e-09 +	8.51e-20 +
F7	5.08e-22 +	1.20e-187 +	4.99e-28 +	1.37e-97 +
F11	2.41e-19 +	1.10e-187 +	1.44e-06 +	1.61e-17 +
F13	5.30e-29 +	1.19e-187 +	5.27e-07 +	8.32e-27 +
F15	1.73e-53 +	1.17e-187 +	5.48e-10 +	1.88e-11 +
F17	1.88e-16 +	1.18e-187 +	1.79e-20 +	9.81e-15 +
F19	1.56e-15 +	1.20e-187 +	2.63e-09 +	3.28e-14 +
F20	1.52e-11 +	1.14e-187 +	7.54e-21 +	4.19e-07 +
F21	1.25e-02 +	1.20e-187 +	2.48e-21 +	8.03e-16 +
F25	1.41e-22 +	1.20e-187 +	7.24e-11 +	2.44e-18 +
F27	2.81e-78 +	1.19e-187 +	2.53e-30 +	4.43e-161 +
F29	1.43e-39 +	1.14e-187 +	1.47e-18 +	8.13e-113 +
F30	6.90e-20 +	1.20e-187 +	1.10e-10 +	3.23e-22 +
+/=/-	15/0/0	15/0/0	15/0/0	15/0/0

From the data in the table, it can be observed that HSAO demonstrates significant superiority in the comparison of 15 benchmark functions, with all p-values being substantially less than 0.05. This indicates that, when addressing these typical optimization problems, HSAO is capable of finding the global optimum more effectively compared to other algorithms, showcasing robust search capabilities and high-quality solutions. In contrast to HSAO, AO, GA, WOA, and GSA fail to exhibit significant advantages on most benchmark functions. Although the differences between these algorithms and HSAO may be relatively minor on certain functions, overall, HSAO demonstrates more stable and superior performance over diverse benchmark functions.

B. Algorithm Performance Analysis

To evaluate the effectiveness of the HSAO algorithm in addressing the co-allocation of MEC server resources under ultra-dense deployment scenarios, 16 deployed base stations are separated from each other by $60m$ on the site of $300m \times 300m$ to test the algorithm's performance. In this paper, we experimentally test the optimization performance of the HSAO algorithm for MSD values in the range of $[80, 100, 120, 140, 160, 180, 200]$, with the channel gain obeying $127 + 30logd$, and the number of channels in all experimental scenarios in this paper is set to 10. All the simulation experiments in this paper refer to the literature [25], [26] for adjusting the parameters such as the computational resources of the MEC servers and the

computational resources of the MSD, and the specific parameter setting values are shown in Table V. In addition, all the test cases in all the experiments in this paper are randomly generated in 30 independent tests within the specified range, such as channel gain and user position, etc. Moreover, all the experiments in this paper use the average value of the algorithm's 30 convergences as the convergence value, and the convergence value is rounded up to five significant digits.

TABLE V: Simulation parameter list

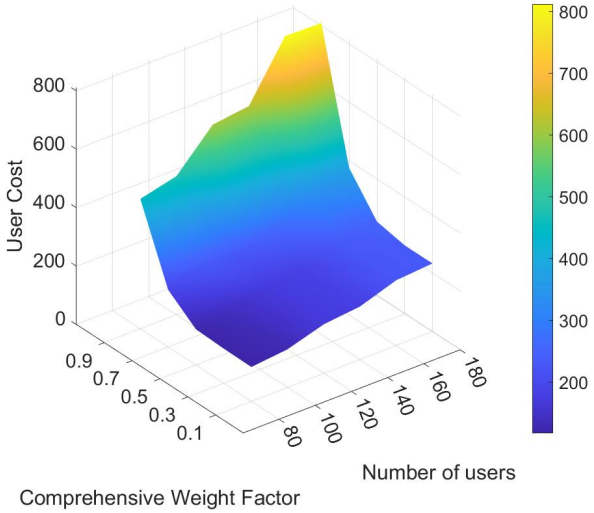
Parameter symbol	Numerical value	Parameter symbol	Numerical value
C_i	$[0.01, 2.5]$ Gigacycles	E_{max}	1000J
D_i	$[0.5, 2]$ MB	W	20MZH
B_i	$[0.1, 1]$ MB	ω	1×10^{-23}
f_i^{max}	$[0.5, 1.5]$ GHZ	p_i^t	0.5W
F_{max}	20 GHZ	p_i^r	0.1W
T_i^{max}	1s		

In this section, the cost weighting factor $\aleph$ is fixed to 0.5 and different comprehensive weighting factor β and number of populations are set to compare the effect. Figure 6 shows the variation of task completion $rate \setminus user$ cost and minimum cost maximum completion value with the integrated weight factor β for the scenarios of 80, 100, 120, 140, 160 and 180 total users respectively.

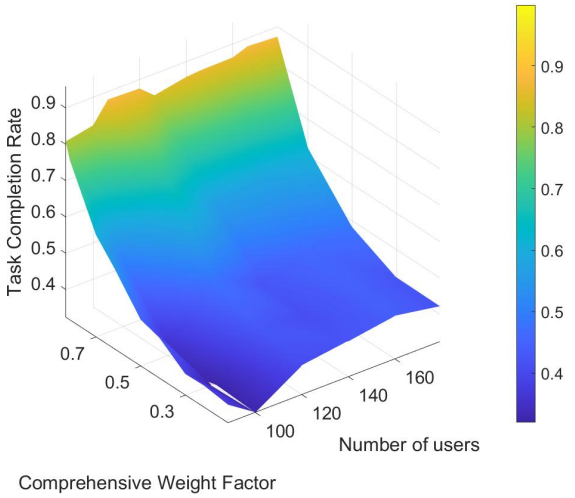
When the combined weight factor β is 0.9, it can be seen from Figure 6a that the user cost grows linearly as the number of users increases. Figure 6b demonstrates that the task completion rate grows at an accelerating rate as the combined weight factor grows when the number of users is constant. Figure 6c demonstrates that as the integrated weight factor β grows, the decline curve of the composite index with the integrated weight factor β of 0.9 is significantly better than the composite index of other integrated weight factors. Therefore, when the comprehensive weight factor β is 0.9, it has the best convergence result, which can get a more reasonable user cost and maximum task completion.

The number of populations determines the length of the search vector, and under the same number of evolutions, the higher the number of populations, the more positions the algorithm traverses, and the stronger the algorithm's search function. The effect of the number of populations on the experimental results is shown in Figure 7. In this section, we test the convergence of HSAO with 20, 30 and 60 populations when the number of users is 180 and the combined weight factor β is 0.9.

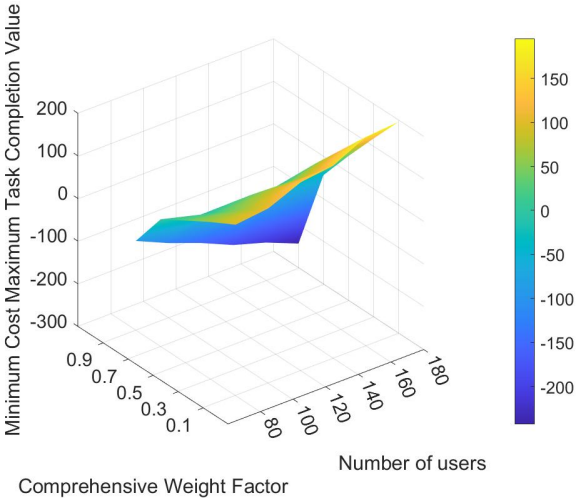
When the number of populations is 20, due to the relatively small number of populations, the algorithm converges slowly, and basically converges at about 779 evolutions, with a lower convergence accuracy of about -235.95. When the number of populations is 30, the algorithm converges faster, and converges to about -235.70 at 636 evolutions, and at 60, the algorithm converges relatively faster, and at 494 evolutions converges to about the optimal solution of the other algorithms, and converges to about -236.44. When the number of evolutions is 1000, the algorithm converges at -245.21 for a population size of 20, -245.94 for a population size of 30, and -246.56 for a population size of 60. Therefore, as the number of populations increases, the accuracy of convergence increases, but the time cost limits the number



(a) The impact of β on user costs



(b) The impact of β on task completion rate



(c) The impact of β on minimum cost maximum task completion value

Fig. 6: Effect of combined weighting factors β on algorithmic metrics

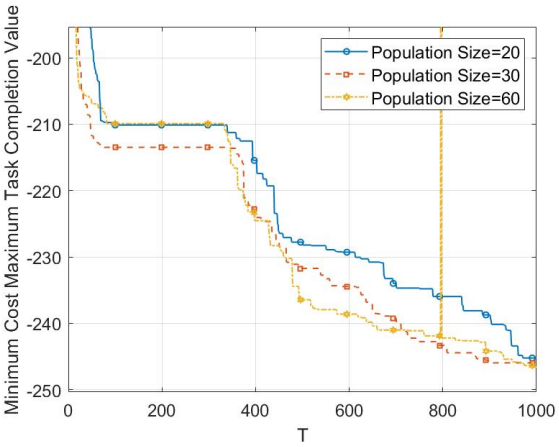


Fig. 7: Effect of the number of populations on the algorithm proposed in this paper

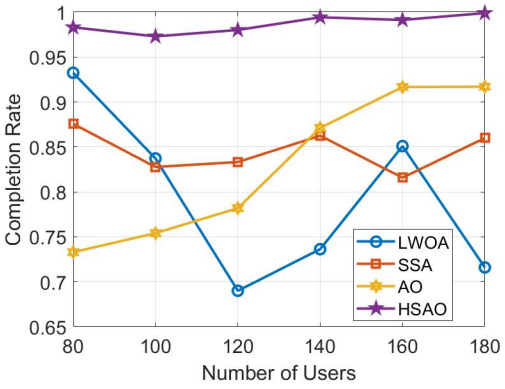
of populations, and for reasons of time cost and convergence accuracy, the algorithm converges at -235.70 for a population size of 636. For the comprehensive consideration of time cost and convergence accuracy, all the experiments in this paper set the number of populations to 30.

C. Algorithm Comparison and Analysis

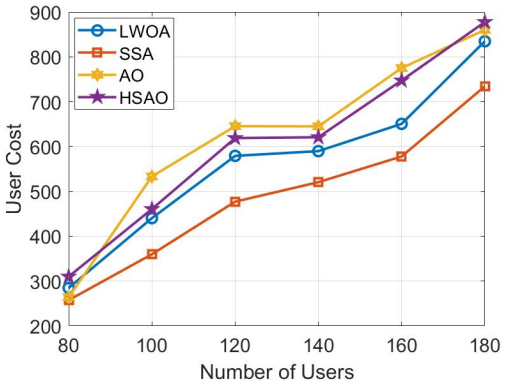
In order to verify the effectiveness and superiority of the HSAO algorithm, this section compares the optimization searching effect of the LWOA, SSA, and AO algorithms, testing the number of users as 80, 100, 120, 140, 160 and 180 respectively, the number of stationary populations as 30, the weighting factor of the user cost and the task completion as β as 0.9, and the weighting factor of the latency and the energy consumption as α as 0.5, the convergence and algorithmic stability of the four algorithms over 1000 convergence and algorithm stability of the four algorithms in 1000 iterations.

1) Comparison between user costs and task completion rates: Figure 8 illustrates the user cost and task completion for four algorithms, HSAO algorithm, LWOA, SSA, and AO algorithm in three different metrics. As can be seen from Figure 8, the HSAO algorithm obtains higher task completion, more reasonable user cost and optimal comprehensive indexes under the above several parameter condition settings. In terms of task completion, HSAO has been maintaining close to 100% task completion and is far ahead of other bionic learning algorithms. And the HSAO algorithm improves up to about 42.0% over LWOA, about 21.5% over SSA, and about 34.1% over AO algorithm. With the increase in the number of users, the demand for resources for edge user tasks increases, in order to improve the completion of the task will inevitably improve the user's cost, but through the optimization of the algorithm can be relatively reasonable user cost to achieve a larger task completion. It can be seen that although the HSAO algorithm is higher than other algorithms in terms of user cost, the HSAO algorithm is greatly ahead of other algorithms in terms of task completion. In general, as the number of users increases, the gap between HSAO's advantage and other algorithms will further widen.

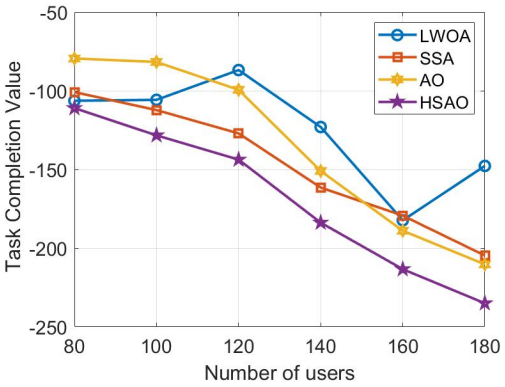
2) Comparison of optimization between minimum cost and maximum completion value: To intuitively analyze



(a) The impact of user quantity on task completion rate



(b) The impact of user quantity on user costs

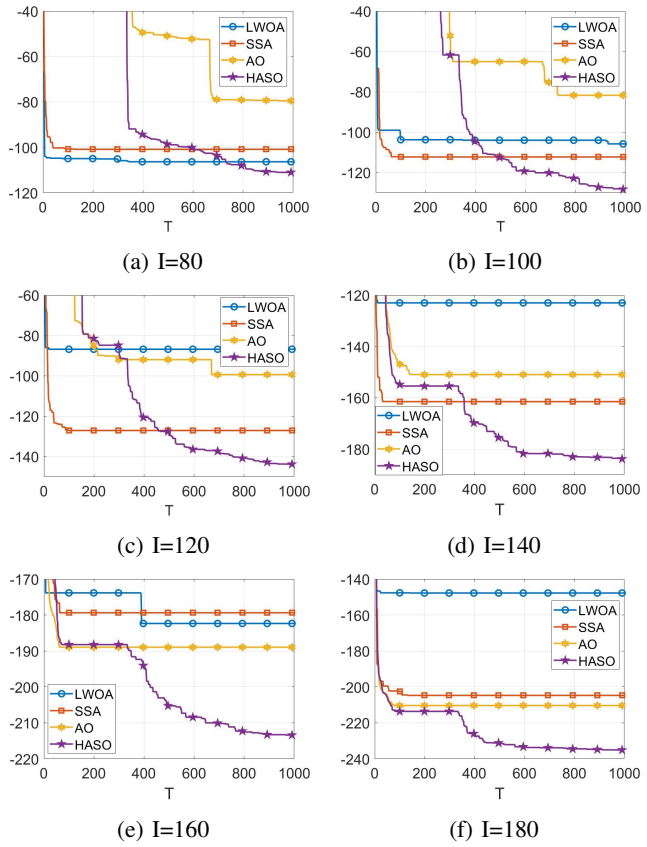


(c) The impact of user quantity on comprehensive indicators

Fig. 8: Influence of user scale on evaluation indicators

the convergence efficiency and outcomes of the four algorithms, the experiment is conducted again according to the simulation environment mentioned above, and Figure 9 depicts the convergence trend graphs of the four algorithms of the system utility under different numbers of users in this experiment. For the convenience of displaying and comparing the experimental results, the optimal objective function value and the convergence situation near the optimal objective function value are intercepted in this experiment.

As can be seen from Figure 9a, when the number of users is small such as I=80, although the HSAO algorithm does not have a significant advantage in convergence speed, the number of evolutionary times reaches 700 when it basically converges to the value of the optimal objective function of the other algorithms, and with the increase in the number



(a) I=80 (b) I=100

(c) I=120 (d) I=140

(e) I=160 (f) I=180

Fig. 9: Convergence plots of optimizing minimum cost and maximizing completion value for four algorithms

of evolutionary times, the HSAO algorithm converges to the optimal value of the function of all the algorithms. From Figure 9b, when I = 100, the HSAO algorithm converges to the optimal convergence result compared to other algorithms when the number of evolution reaches 500, and after 500 times of evolution, the optimization result of the HSAO algorithm is significantly better than other algorithms, and the algorithm is still in the stage of continuous optimization when it reaches 1000 times of evolution. From Figure 9c, when I=120, the HSAO algorithm converges to the optimal convergence result of SSA and LWOA when the number of evolutions reaches 330, and after 500 evolutions, the optimization result of the HSAO algorithm is obviously better than the other algorithms, and the algorithm is still obviously in the stage of continuous optimization. From Figure 9d, Figure 9e and Figure 9f, it is observed that as the user count increases, specifically when $I = 140$, $I = 160$, and $I = 180$, the HSAO algorithm exhibits markedly superior convergence performance compared to other methods, whereas the LWOA algorithm demonstrates the poorest convergence behavior. Prior to 100 iterations, the HSAO algorithm nearly aligns with the optimal objective function values achieved by AO and other competitors. Beyond 100 iterations, however, HSAO consistently surpasses its counterparts. These results indicate that, although the convergence rate of HSAO is not initially the fastest among bio-inspired algorithms, its superiority becomes increasingly pronounced as the number of users grows, enabling it to rapidly approach the optimal solutions obtained by other bio-inspired methods.

3) Stability Comparison: To further validate the stability of the HSAO algorithm, this section tests all scenarios using four different algorithms with a fixed number of evolutions of 1,000 and a population size of 30. The experimental data was collected 30 times, resulting in the optimal solution, the worst solution, the average solution, and the variance, and the results of the stability of the four algorithms are listed in Table VI.

TABLE VI: Stability indicators of different algorithms

Numbers of users	Evaluation metrics	LWOA	SSA	AO	HSAO
80	average value	-106.35	-100.87	-79.518	-111.03
	variance	108.17	81.449	69.213	30.260
	worst case solution	-84.199	-89.505	-66.843	-100.21
	optimal solution	-115.17	-177.63	-92.693	-188.17
100	average value	-105.78	-112.28	-81.709	-128.34
	variance	800.80	75.010	358.59	104.61
	worst case solution	-65.407	-104.17	-58.439	-106.70
	optimal solution	-132.36	-128.28	-114.99	-138.38
120	average value	-86.850	-127.09	-99.414	-143.77
	variance	657.07	208.71	481.75	99.784
	worst case solution	-69.759	-111.09	-74.386	-124.15
	optimal solution	-140.82	-155.42	-141.58	-155.45
140	average value	-123.04	-161.51	-150.98	-183.81
	variance	915.45	409.12	145.64	34.2610
	worst case solution	-96.215	-138.06	-126.79	-172.06
	optimal solution	-177.90	-194.12	-171.66	-194.24
160	average value	-182.35	-179.37	-188.95	-213.39
	variance	642.96	168.567	149.79	47.182
	worst case solution	-125.17	-160.51	-164.33	-202.71
	optimal solution	-208.43	-198.45	-206.91	-225.30
180	average value	-147.77	-204.71	-210.34	-235.06
	variance	834.752	877.595	860.09	734.24
	worst case solution	1917.5	617.35	60.766	29.777
	optimal solution	-109.74	-164.36	-198.32	-224.71

From Table VI, when solving the resource scheduling problem of MSD with the collaborative assistance of MEC servers in ultra-dense deployment, LWOA has the worst stability and poor convergence accuracy; HSAO algorithm has a higher mean than the other three algorithms as well as a smaller variance and a better algorithmic stability, which makes it more suitable for solving large-scale optimization problems with large-scale number of users.

In different scenarios, by testing the weighting factors on the performance of the algorithm, the demonstration of different convergence curves, and the analysis of the stability of the algorithm. It can be concluded that under identical conditions, the HSAO algorithm achieves accelerated convergence and enhanced solution accuracy, while exhibiting a lower tendency to become trapped in local optima. Moreover, it is capable of deriving the optimal resource co-allocation strategy within a reasonable number of iterations.

VI. Conclusion

This study investigates the resource allocation problem in ultra-dense networks involving mobile edge computing (MEC) servers. By constructing a system cost model, a novel improved Aquila Optimizer algorithm, HSAO (Hybrid Strategy Aquila Optimizer), is proposed to address the resource allocation optimization issue. The HSAO algorithm incorporates several advanced strategies, including candidate

pattern pruning, Tent chaotic population initialization, random spiral update mechanisms, and a hybrid mutation strategy combining random reverse learning with the simplex method, significantly enhancing its global search capability and convergence performance. Experimental results demonstrate that HSAO outperforms traditional bio-inspired algorithms (such as WOA, GSA, and AO) in terms of optimization accuracy, convergence speed, and stability, both on CEC2017 benchmark functions and practical application scenarios. Furthermore, through comparative analysis of user costs and task completion rates, the superiority of HSAO in reducing service latency, improving task execution efficiency, and enhancing user satisfaction is validated. This research provides an efficient and reliable optimization solution for resource allocation in ultra-dense networks. Future studies will focus on the design and implementation of dynamic real-time scheduling strategies to meet diverse requirements in high-speed mobile network scenarios such as autonomous driving.

REFERENCES

[1] A. Sarah, G. Nencioni, and M. M. I. Khan, "Resource allocation in multi-access edge computing for 5g-and-beyond networks," *Computer Networks*, vol. 227, p. 109720, 2023.

[2] M. Chowdhury, "Campaign: A personalized offloading, semantic communication, latency-aware resource slicing and sfc orchestration for sdn and nfv empowered 6g serverless computing network," *IAENG International Journal of Computer Science*, vol. 51, no. 10, pp. 1480–1515, 2024.

[3] Z. Qin, F. Xu, Y. Xie, Z. Zhang, and G. Li, "An improved top-k algorithm for edge servers deployment in smart city," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 8, p. e4249, 2021.

[4] Z. Wang, J. Lv, Y. Hou, and D. Song, "Enhancing experience: Investigating the impact of different personal perspectives in virtual reality with lower limb rehabilitation robots on participants' motivation, experience, and engagement," *International Journal of Industrial Ergonomics*, vol. 99, p. 103496, 2024.

[5] M. Sun, P. Li, H. Qin, N. Liu, H. Ma, Z. Zhang, J. Li, B. Lu, X. Pan, and L. Wu, "Liquid metal/cnts hydrogel-based transparent strain sensor for wireless health monitoring of aquatic animals," *Chemical Engineering Journal*, vol. 454, p. 140459, 2023.

[6] J. Lee, R. Mao, and A. Pervez, "Perceived risk of crime on driverless public bus and ride-pooling services in china," *Travel Behaviour and Society*, vol. 35, p. 100730, 2024.

[7] J. Wang, W. Wu, Z. Liao, R. S. Sherratt, G.-J. Kim, O. Alfarraj, A. Alzubi, and A. Tolba, "A probability preferred priori offloading mechanism in mobile edge computing," *IEEE Access*, vol. 8, pp. 39 758–39 767, 2020.

[8] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Service-Oriented Computing: 16th International Conference, ICSOC 2018, Hangzhou, China, November 12-15, 2018, Proceedings 16*. Springer, 2018, pp. 230–245.

[9] M.-T. Thai, Y.-D. Lin, Y.-C. Lai, and H.-T. Chien, "Workload and capacity optimization for cloud-edge computing systems with vertical and horizontal offloading," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 227–238, 2019.

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

[11] Z. Li and Q. Zhu, "Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing," *Information*, vol. 11, no. 2, p. 83, 2020.

[12] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5g," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6398–6409, 2018.

[13] L. N. Huynh, Q.-V. Pham, X.-Q. Pham, T. D. Nguyen, M. D. Hossain, and E.-N. Huh, "Efficient computation offloading in multi-tier multi-access edge computing systems: A particle swarm optimization approach," *Applied Sciences*, vol. 10, no. 1, p. 203, 2019.

[14] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. Al-Qaness, and A. H. Gandomi, "Aquila optimizer: a novel meta-heuristic optimization algorithm," *Computers & Industrial Engineering*, vol. 157, p. 107250, 2021.

[15] Z.-j. Teng, J.-l. Lv, and L.-w. Guo, "An improved hybrid grey wolf optimization algorithm," *Soft computing*, vol. 23, pp. 6617–6631, 2019.

[16] Q. Yang, Y. Gao, and Y. Song, "A tent lévy flying sparrow search algorithm for wrapper-based feature selection: A covid-19 case study," *Symmetry*, vol. 15, no. 2, p. 316, 2023.

[17] J. Fan, Y. Li, and T. Wang, "An improved african vultures optimization algorithm based on tent chaotic mapping and time-varying mechanism," *Plos one*, vol. 16, no. 11, p. e0260725, 2021.

[18] M. Abdel-Basset, R. Mohamed, and S. Mirjalili, "A novel whale optimization algorithm integrated with nelder–mead simplex for multi-objective optimization problems," *Knowledge-Based Systems*, vol. 212, p. 106619, 2021.

[19] C. Chen and L. Yu, "A hybrid ant lion optimizer with improved nelder–mead algorithm for structural damage detection by improving weighted trace lasso regularization," *Advances in Structural Engineering*, vol. 23, no. 3, pp. 468–484, 2020.

[20] R. M. Rizk-Allah and A. E. Hassanien, "A hybrid harris hawks-nelder-mead optimization for practical nonlinear ordinary differential equations," *Evolutionary Intelligence*, vol. 15, no. 1, pp. 141–165, 2022.

[21] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)*, vol. 1. IEEE, 2005, pp. 695–701.

[22] X. Yu, W. Xu, and C. Li, "Opposition-based learning grey wolf optimizer for global optimization," *Knowledge-Based Systems*, vol. 226, p. 107139, 2021.

[23] S. Shekhawat and A. Saxena, "Development and applications of an intelligent crow search algorithm based on opposition based learning," *ISA transactions*, vol. 99, pp. 210–230, 2020.

[24] S. Ekinci, D. Izci, E. Eker, and L. Abualigah, "An effective control design approach based on novel enhanced aquila optimizer for automatic voltage regulator," *Artificial Intelligence Review*, vol. 56, no. 2, pp. 1731–1762, 2023.

[25] Q.-V. Pham, S. Mirjalili, N. Kumar, M. Alazab, and W.-J. Hwang, "Whale optimization algorithm with applications to resource allocation in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4285–4297, 2020.

[26] P.-Q. Huang, Y. Wang, K. Wang, and Z.-Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE transactions on cybernetics*, vol. 50, no. 10, pp. 4228–4241, 2019.