Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.058450

ARTICLE

Innovative Approaches to Task Scheduling in Cloud Computing Environments Using an Advanced Willow Catkin Optimization Algorithm

Jeng-Shyang Pan[1,2], Na Yu[1], Shu-Chuan Chu[1,*], An-Ning Zhang[1], Bin Yan[3] and Junzo Watada[4]

[1]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, 266590, China

[2]Department of Information Management, Chaoyang University of Technology, Taichung, 41349, Taiwan

[3]College of Electronics, Communication and Physics, Shandong University of Science and Technology, Qingdao, 266590, China

[4]Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, 808-0135, Japan

*Corresponding Author: Shu-Chuan Chu. Email: scchu0803@gmail.com

Received: 12 September 2024 Accepted: 20 November 2024 Published: 17 February 2025

ABSTRACT

The widespread adoption of cloud computing has underscored the critical importance of efficient resource allocation and management, particularly in task scheduling, which involves assigning tasks to computing resources for optimized resource utilization. Several meta-heuristic algorithms have shown effectiveness in task scheduling, among which the relatively recent Willow Catkin Optimization (WCO) algorithm has demonstrated potential, albeit with apparent needs for enhanced global search capability and convergence speed. To address these limitations of WCO in cloud computing task scheduling, this paper introduces an improved version termed the Advanced Willow Catkin Optimization (AWCO) algorithm. AWCO enhances the algorithm's performance by augmenting its global search capability through a quasi-opposition-based learning strategy and accelerating its convergence speed via sinusoidal mapping. A comprehensive evaluation utilizing the CEC2014 benchmark suite, comprising 30 test functions, demonstrates that AWCO achieves superior optimization outcomes, surpassing conventional WCO and a range of established meta-heuristics. The proposed algorithm also considers trade-offs among the cost, makespan, and load balancing objectives. Experimental results of AWCO are compared with those obtained using the other meta-heuristics, illustrating that the proposed algorithm provides superior performance in task scheduling. The method offers a robust foundation for enhancing the utilization of cloud computing resources in the domain of task scheduling within a cloud computing environment.

KEYWORDS

Willow catkin optimization algorithm; cloud computing; task scheduling; opposition-based learning strategy

1 Introduction

1.1 Research Background

Cloud computing plays a pivotal role in various fields, offering a ubiquitous resource base where users can readily access cloud services on demand [1]. The "pay-as-you-go" model, characteristic of

web-based applications, is employed in cloud computing to facilitate large-scale sharing of resources, including software and hardware [2]. Cloud computing systems utilize distributed technology and virtualization to enhance flexibility and resource utilization, thereby addressing diverse user requirements. Cloud service providers [3] assume responsibility for hardware and software maintenance and updates, while users need only specify their business requirements. Additionally, users circumvent the substantial expenses typically associated with constructing and maintaining extensive IT infrastructure.

Cloud computing operates on the principle of providing computing resources on demand through virtualization technologies. Task scheduling [4–6] plays a crucial role in this process, as it manages the allocation of computing tasks to virtual machines (VMs) for optimized resource utilization and system performance. The importance of task scheduling lies in its ability to ensure efficient task completion while minimizing costs and execution time and maintaining system load balance. However, current task scheduling faces numerous challenges, including finding an optimal balance between cost management, task execution efficiency, and load balancing to meet increasing computational demands.

1.2 Research Objective

The task scheduling problem in distributed systems [7,8] is generally considered a complex combinatorial optimization challenge, classified as a Non-deterministic Polynomial (NP)-hard problem [9]. Meta-heuristic algorithms [10,11] constitute a comprehensive framework for optimization methods and have demonstrated significant advantages in addressing complex objective optimization problems, particularly in resource allocation, task scheduling, and performance optimization. Population intelligence optimization algorithms [12–14], a subset of meta-heuristic algorithms, identify solutions by simulating group behavior or individual interactions. Examples of such techniques include Particle Swarm Optimization (PSO) [15–17], Ant Colony Optimization (ACO) [18], Genetic Algorithm (GA) [19,20], Tumbleweed Algorithm (TA) [21], Cat Swarm Optimization (CSO) [22,23], Artificial Bee Colony (ABC) [24], and Willow Catkin Optimization (WCO) [25].

The WCO algorithm is a meta-heuristic optimization technique designed to address complex optimization challenges by simulating the natural process of willow seed dispersal and aggregation in wind currents. This algorithm explores the solution space by emulating the diffusion and aggregation patterns of willow seeds in nature. However, the WCO algorithm exhibits certain limitations, particularly in its global search capability and convergence speed, both of which are comparatively slow. Additionally, the algorithm is prone to being trapped in local optima, which poses a significant challenge in identifying the optimal solution for complex problems.

To address these challenges, an improved Advanced Willow Catkin Optimization (AWCO) algorithm is developed in this study. This novel algorithm enhances global search capability by incorporating a reverse learning strategy and employs sinusoidal mapping to expedite convergence. AWCO substantially improves resource utilization efficiency in cloud computing task scheduling through optimized task allocation processes. The algorithm simulates willow diffusion and aggregation dynamics by initializing a set of candidate solutions, integrating these with a reverse learning strategy for global search, and swiftly identifying optimal task allocation schemes. In applications, AWCO assigns the most suitable VMs to each task, ensuring task completion with optimal resource consumption and minimal time. Simultaneously, AWCO accelerates algorithm convergence through sinusoidal mapping, effectively avoids local optima, and ultimately achieves load balancing and improved system performance in task scheduling.

1.3 Contributions and Structure of the Paper

This paper makes the following contributions:

- The AWCO algorithm is introduced as an innovative optimization approach. The proposed reverse learning strategy is incorporated into the individual evolution process of WCO, thereby enhancing the algorithm's capacity for exploration and exploitation, as well as its global search capability and convergence speed.
- The sinusoidal mapping strategy is integrated into the individual position updating process to minimize the distance between individuals and the optimal solution, thereby enhancing the algorithm's performance.
- The AWCO algorithm is evaluated using 30 test functions from the CEC2014 dataset and is benchmarked against various alternative algorithms, including GA, bat algorithm (BA), and PSO. The experimental results demonstrate that AWCO exhibits superior performance.
- To assess the efficacy of the proposed algorithm, it is applied to task scheduling in cloud computing environments. The experimental results indicate that, compared to similar algorithms in two heterogeneous cloud environments, the enhanced algorithm improves resource utilization for task scheduling.

Section 2 offers a comprehensive review of existing scheduling models and techniques utilized in cloud computing environments. Section 3 introduces the proposed scheduling system, followed by Section 4, which provides an in-depth examination of the WCO algorithm. Section 5 elucidates the improvements made to the WCO algorithm and illustrates the implementation of AWCO for task scheduling purposes. Section 6 presents the experimental outcomes and their subsequent performance analysis, while Section 7 concludes with a summary of the paper's key findings and contributions.

2 Related Work

2.1 Task Scheduling Models

The diverse requirements of tasks and the intricacies of cloud environments have led to the development of numerous models for task scheduling in cloud computing. Among these, scheduling methods based on mathematical models are the most precise, typically employing mathematical optimization techniques such as linear programming, integer programming, and dynamic programming to determine the optimal task allocation scheme. These models represent the objectives and constraints of task scheduling through explicit mathematical formulas. For instance, the model proposed by Alguliyev et al. [26] ensures the efficient allocation of mobile users' tasks to appropriate cloud resources. This is accomplished by defining a set of mathematical variables and constraints that minimize task processing time, reduce network latency, and lower the energy consumption of mobile devices. However, the computational complexity of these models increases significantly as the problem size grows. To address the challenges posed by large-scale task scheduling problems in cloud computing environments more effectively, meta-heuristic algorithms are adopted.

Moreover, load-balancing task scheduling models aim to ensure a relatively balanced load across computing nodes in cloud computing systems. Fang et al. [27] proposed a two-tier scheduling mechanism for efficient task execution in cloud computing environments. This approach initially assigns tasks to VMs, followed by mapping these VMs to physical resources. However, unlike the model presented in this study, these conventional methods typically focus solely on balanced task distribution without considering factors such as the optimization of task execution time and cost.

2.2 Task Scheduling Strategy

Task scheduling policies in cloud computing environments play a crucial role in efficient task allocation and scheduling to cloud computing resources, aiming to optimize system performance and resource utilization. These policies typically consider various factors, including time, cost, makespan [28], reliability, availability, throughput, and resource utilization, to ensure the reliable and cost-effective execution of tasks within a cloud environment. Common task scheduling techniques, such as dynamic priority scheduling [29] and hierarchical scheduling [30], are frequently employed for task optimization in cloud computing. Additionally, meta-heuristic algorithms are utilized to address cloud task scheduling challenges [31,32]. These algorithms offer advantages such as conducting more comprehensive searches, accommodating complex objective optimization problems, avoiding local optima, and improving resource utilization.

Task scheduling algorithms in these environments include those based on GA, PSO, and ACO. GAs in task scheduling identify optimal solutions by simulating natural selection and evolution processes. Changtian et al. [33] characterized independent task scheduling as a two-objective minimization problem, optimizing completion time and energy consumption. Sheng et al. [34] noted that most task scheduling algorithms prioritized overall task completion duration over individual task completion time and proposed a Template-Based Genetic Algorithm (TBGA) with user Quality of Service (QoS) constraints. Pirozmand et al. [35] emphasized that scheduling and associated energy consumption posed significant challenges for cloud computing systems and proposed a two-step hybrid scheduling approach. The PSO algorithm in task scheduling identifies optimal solutions by simulating collaborative and informational transfer dynamics observed in bird flocks or fish schools. Mansouri et al. [36] developed a new task scheduling algorithm, FMPSO, employing advanced PSO techniques and fuzzy systems. Task scheduling based on the ACO algorithm simulates ant colony behavior to solve task allocation and scheduling problems. Moon et al. [37] proposed an innovative ACO algorithm, SACO, for cloud computing task scheduling. Experimental findings confirmed SACO's ability to maximize cloud server utilization while solving the NP-hard issue. In contrast, AWCO demonstrates enhanced global search capabilities and dynamic adaptability in task scheduling. By simulating willow diffusion and aggregation processes, AWCO accelerates convergence speed, avoids local optima, and achieves efficient resource utilization and balanced system load.

3 Task Scheduling System

3.1 Model Design

In a cloud computing environment, the task scheduling algorithm initiates the assignment of tasks to VMs upon their addition to the task queue by the task manager. Fig. 1 illustrates a representative example of task scheduling in a cloud computing system. Presently, task scheduling in cloud computing faces significant challenges, primarily in intelligently allocating tasks to processors for cost-effectiveness, load balancing, and reduced makespan. To address the limitations of current research on task scheduling algorithms in cloud computing, this study proposes an enhanced task scheduling method utilizing an enhanced version of the Advanced Phasmatodea Population Evolution (APPE) algorithm [38].

Each task is characterized by two attributes: the number of tasks designated as n and the number of machine language instructions in millions required for task execution represented by variable *len*. VMs possess five attributes: m, $MIPS$, RAM, *bandwidth*, and *storage*. The m attribute represents the number of available VMs, while $MIPS$ indicates the average speed at which a VM processes single-word fixed-point instructions. The RAM attribute refers to the VM's memory capacity, the *bandwidth* attribute

represents the network bandwidth of a VM, and the *storage* attribute denotes a VM's storage capacity. Several key factors primarily influence the load profile of a VM. The system employs sequential task execution through task queues and scheduling algorithms, whereby each assignment is completed linearly on the VM.

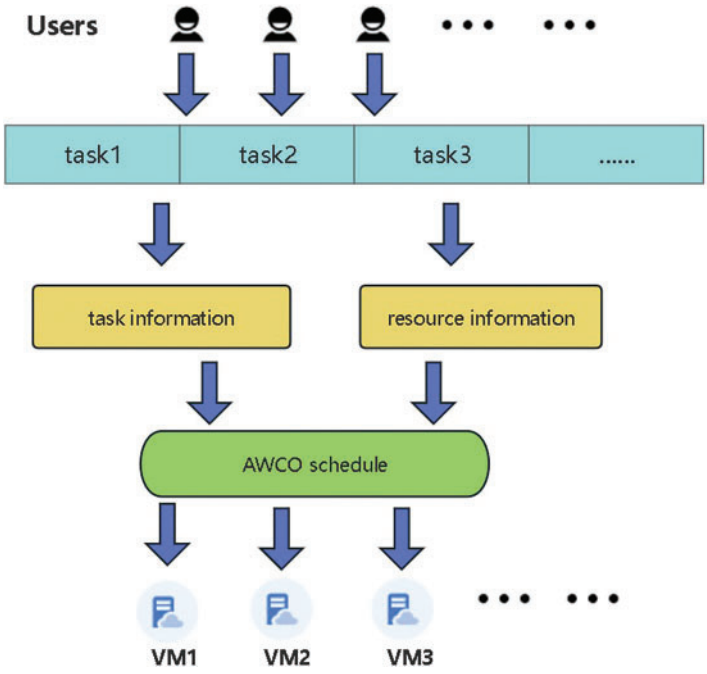

Figure 1: Task scheduling model for cloud computing environment

3.2 Formulation of the Objective Function

This article aims to assess the performance of task scheduling in makespan, cost-effectiveness, and load balancing.

Consider a scenario where a cloud user submits a task queue, denoted as Task, comprising tasks T_1, T_2, ..., up to T_n ($30 \leq n \leq 300$). Testing low and high task counts demonstrates the algorithm's adaptability to varying task quantities. Additionally, a set of VMs is defined as $VM = \{VM_1, VM_2, ..., VM_m\}$, where VM_j corresponds to the j-th VM. We introduce the variable EXT, represented as $EXT = \{EXT_{ij}\}_{n \times m}$, to signify the expected execution time. This metric indicates the time required for task i to be processed on VM_j and is calculated using the following Eq. (1):

$$EXT_{ij} = \frac{len_i}{MIPS_j} \tag{1}$$

The variable T_length, consisting of elements len_1, len_2, ..., and len_n, represents the duration of the i-th task, denoted as len_i. $MIPS_j$ denotes the execution rate of VM_j. Consequently, EXT_{ij} signifies the estimated time required to complete task i on VM_j, considering various factors such as task length and potentially relevant parameters.

(a) Cost:

The cost of a VM encompasses multiple factors, including the utilization of CPU, memory, and storage resources, as well as the operational duration of the VM. These costs are typically billed on a usage-per-unit-of-time basis. The calculation formula is presented in Eq. (2):

$$\text{Cost} = \sum_{j=1}^{m}\left(\cos t_j \times \left(\sum_{i=1}^{n} EXT_{ij} \times RUN_{ij}\right)\right) \tag{2}$$

where cost_j denotes the hourly cost of the jth VM, influenced by *MIPS*, *RAM*, *bandwidth*, and *storage* in this environment. Let $RUN = \{RUN_{ij}\}_{n \times m}$, where RUN_{ij} is a binary indicator (1 or 0) representing whether task i is executed on VM j. Specifically, $RUN_{ij} = 1$ indicates task i is assigned to VM j, while $RUN_{ij} = 0$ implies otherwise.

(b) Makespan:

In the context of task scheduling, *Makespan* represents a critical performance metric for the total time required to complete the entire set of tasks, which reflects the efficiency and productivity of task scheduling. Consequently, minimizing *Makespan* is commonly considered an optimization objective in the development of scheduling algorithms. The calculation formula is presented in Eq. (3):

$$Makespan = \max_{j=1}^{m}\left(\sum_{i=1}^{n} EXT_{ij} \times RUN_{ij}\right) \tag{3}$$

(c) Load:

The significance of load in task scheduling is exemplified by the system's selection of suitable nodes for task execution. A node with a higher load may result in increased task execution time. Consequently, task scheduling algorithms typically consider node load and select nodes with the lowest load to execute tasks, aiming to maintain system efficiency and optimize performance. The calculation formula is provided in Eq. (4):

$$Load = \frac{\sqrt{\sum_{j=1}^{m}\left(load_j - \langle load\rangle\right)^2 \times VT_j}}{m \times Makespan} \tag{4}$$

where m is the number of VMs, $load_j$ denotes the load of the jth VM, $\langle load\rangle$denotes the overall VMs' average load, and VT_j denotes the running time of the jth VM, as described in Eqs. (5)–(7). Specifically, *MIPS* is used to calculate the task execution time of a VM; *RAM* and *storage* impose constraints on the number and size of tasks a VM can host; and *bandwidth* is utilized to simulate the task transmission time over the network, reflecting network latency's impact on task completion time. Consequently, the resource constraints of the VM directly influence task allocation and execution, ensuring that the processing capacity of the VM is not exceeded during scheduling [39].

$$load_j = \alpha \times MIPS + \beta \times RAM + \gamma \times bandwidth + \delta \times storage \tag{5}$$

$$\langle load\rangle = \frac{\sum_{j=1}^{m} load_j}{m} \tag{6}$$

$$VT_j = \sum_{i=1}^{n} EXT_{ij} \times RUN_{ij} \tag{7}$$

where α, β, γ, and δ are the four parameters.

Based on these performance indicators, the expression of the fitness function is given in Eq. (8):

$$Fitness = a_1 \times Makespan + a_2 \times \mathrm{Cost} + a_3 \times Load \tag{8}$$

where a_1, a_2, and a_3 are three weights that satisfy $a_1 + a_2 + a_3 = 1$. The value of each weight can be adjusted according to specific task requirements. The objective of task scheduling is to minimize the value of the fitness function.

4 WCO Algorithm

The WCO algorithm simulates the dispersal of willow seeds, also referred to as willow flakes, as they are carried by the wind. This process is primarily influenced by two factors: wind speed and wind direction. As the willow flakes descend, they tend to cluster together, potentially adhering to one another when in close proximity. Furthermore, the algorithm reflects the gradual decrease in the dispersal range of willow seeds, representing a transition from an exploration phase to an exploitation phase in the optimization process.

4.1 Initialization Phase

The algorithm's initialization phase involves generating random seeds and distributing them across the solution space. This space can be represented by a matrix with N rows and D columns, where N signifies the total population size, and D represents the dimensionality of each individual. In this matrix, each row corresponds to a single individual, while each column represents a specific attribute of that individual. The WCO process commences with this random population in the willow tree, generating the initial random population using the following Eq. (9):

$$x_i = r \times (UB - LB) + LB, i = 1, 2, \ldots, N \tag{9}$$

Here, the random number r is within the range [0, 1], x_i denotes a D-dimensional solution, and the solution space is bounded by an upper limit UB and a lower limit LB.

4.2 Search Phase

During the search phase, following the descent of willow catkins from the willow tree, each flake is influenced by two primary factors: wind speed and direction. If individuals update their positions too closely to one another during the iterative process, they will be blown closer together by the wind. To measure the distance between individuals, the distance between the current individual and the global optimal solution, denoted by d_i, is calculated. An adhesion-prone radius R is introduced; if the distance d_i between individuals exceeds R, they are less likely to stick together, and Eq. (10) is applied to determine the stochastic wind speed and direction. Conversely, if d_i is less than or equal to R, indicating a higher probability of adhesion, Eq. (11) is employed. The weights are then computed according to Eqs. (12) and (13).

$$\begin{cases} ws = r \times R \\ wd = r \times 2\pi \end{cases} \tag{10}$$

$$\begin{cases} ws = \mu \times \left(\sum_{i=1}^{n} K_i \left| p_g - x_i \right| \right) + (1 - \mu) \times r \times R \\ wd = ar\cos\left(\frac{x_i \bullet p_g}{\|x_i\| \times \|pg\|} \right) + r \times \frac{\pi}{8} \end{cases} \tag{11}$$

$$DW = 1 - \frac{\left| p_g - x_i \right|}{\left\| x_i - p_g \right\|} \tag{12}$$

$$K = \frac{DW}{\sum_{i=1}^{n} DW_i} \tag{13}$$

where p_g represents the current global optimal solution, the symbol $\|\cdot\|$ signifies the Euclidean distance between x_i and p_g, DW denotes the weight of the distance between p_g and x_i in each dimension relative to the total distance, while μ is a random number generated within the range of 0.4 to 0.6.

By incorporating the randomly generated wind speed ws and wind direction wd into Eq. (14), we can determine the direction and distance of an individual's movement during each iterative update.

$$\begin{cases} u = -ws \times \sin(wd) \\ v = -ws \times \cos(wd) \end{cases} \tag{14}$$

The individual update process can be accomplished by decomposing the wind's direction and speed into components u and v. We employ Eq. (15) to execute the individual update.

$$x_{i+1} = x_i + a \times (u \times v) \times r + (2 - a)\left(p_g - x_i\right) \tag{15}$$

where x_i represents the current position of the individual, and a is a parameter that governs the transition from exploration to exploitation during the individual's iterative process.

$$a = c \times e^{-\left(\frac{t}{1000}\right)^2} \tag{16}$$

Here, t represents the current number of iterations, T denotes the maximum number of iterations allowed, and c is a constant fixed at 2.

5 AWCO Algorithm

The update mechanism in WCO primarily relies on local information despite its efforts to guide updates toward the global optimum in each generation. However, an update mechanism lacking global search capabilities may struggle with complex optimization problems. The proposed reverse learning strategy addresses this limitation by considering both the current and inverse solutions, thereby expanding the search space coverage and exploring a wider range of potential solutions. Additionally, improvements to the WCO algorithm's convergence speed are essential. The incorporation of sinusoidal mapping in the initialization phase and position update formula aims to enhance the algorithm's ability to identify the optimal solution.

5.1 Improvement Methods

5.1.1 Sinusoidal Mapping for Enhanced Population Initialization

Sine chaotic mapping [40] is a chaotic system derived from sinusoidal functions, which is characterized by high dynamic randomness that produces a nonlinear and unpredictable output

sequence. The nonlinear properties of this mapping increase the algorithm's complexity, particularly in scenarios where avoiding locally optimal solutions is crucial. Implementing sine chaotic mapping in the algorithm enables a more thorough exploration of the search space. Its irregular trajectory allows for the traversal of each potential solution and enhances the algorithm's convergence, thus facilitating more efficient identification of the global optimum solution. The mathematical formulation of this mapping is presented in the following Eq. (17):

$$z_{k+1} = \frac{4}{a} \sin(\pi z_k) \tag{17}$$

The sinusoidal mapping's value domain is defined as the interval $[-1, 1]$. In Eq. (17), a represents a real number within the interval $(0, 4]$, z_k denotes the mapping sequence of the k-th individual, while z_{k+1} represents the mapping sequence of the $(k+1)$-th individual.

The parameter a plays a crucial role in determining the randomness of sinusoidal mapping. To identify the optimal value of a, this study conducts a comparative experiment using CEC2014, with the results presented in Table 1. The algorithms yielding the optimal results are highlighted in bold. The comparison reveals that when a is set to 4, the algorithm achieves the highest number of optimal outcomes. Consequently, this paper adopts a value of 4 for a.

Consequently, the initialization formula is modified as follows:

$$x_i = z \times (UB - LB) + LB, i = 1, 2, \ldots, N \tag{18}$$

where z represents the sequence of sinusoidal mappings.

Table 1: Comparison of results for different values of a

Function	a = 1	a = 2	a = 3	a = 4
F1	1.03E+06	5.03E+05	3.00E+05	**2.62E+05**
F2	1.28E+06	1.20E+06	1.18E+06	**1.15E+06**
F3	1.49E+03	6.94E+02	**5.88E+02**	1.04E+03
F4	4.25E+02	4.24E+02	4.21E+02	**4.18E+02**
F5	5.20E+02	**5.20E+02**	5.20E+02	**5.20E+02**
F6	6.03E+02	**6.02E+02**	6.02E+02	6.02E+02
F7	7.01E+02	7.01E+02	**7.01E+02**	**7.01E+02**
F8	8.17E+02	8.13E+02	8.15E+02	**8.12E+02**
F9	9.16E+02	9.13E+02	9.16E+02	**9.11E+02**
F10	1.51E+03	**1.44E+03**	1.50E+03	1.50E+03
F11	1.78E+03	1.71E+03	1.62E+03	**1.60E+03**
F12	1.20E+03	1.20E+03	1.20E+03	**1.20E+03**
F13	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**
F14	**1.40E+03**	**1.40E+03**	**1.40E+03**	**1.40E+03**
F15	1.50E+03	**1.50E+03**	**1.50E+03**	**1.50E+03**
F16	1.60E+03	1.60E+03	1.60E+03	**1.60E+03**
F17	7.78E+03	7.08E+03	7.15E+03	**4.90E+03**
F18	2.02E+04	2.07E+04	**1.79E+04**	2.82E+04

(Continued)

Table 1 (continued)

Function	a = 1	a = 2	a = 3	a = 4
F19	**1.90E+03**	**1.90E+03**	**1.90E+03**	**1.90E+03**
F20	2.31E+03	2.23E+03	**2.09E+03**	2.20E+03
F21	5.86E+03	5.00E+03	**5.00E+03**	5.39E+03
F22	2.25E+03	2.25E+03	2.25E+03	**2.24E+03**
F23	**2.63E+03**	**2.63E+03**	**2.63E+03**	**2.63E+03**
F24	2.53E+03	2.52E+03	2.52E+03	**2.52E+03**
F25	**2.65E+03**	2.66E+03	2.65E+03	2.65E+03
F26	**2.70E+03**	**2.70E+03**	**2.70E+03**	**2.70E+03**
F27	**2.72E+03**	2.74E+03	2.76E+03	2.95E+03
F28	3.23E+03	**3.21E+03**	3.22E+03	3.26E+03
F29	4.34E+03	3.70E+03	3.60E+03	**3.07E+05**
F30	4.40E+03	4.04E+03	**4.01E+03**	4.27E+03
Total	7	10	12	20

5.1.2 Sinusoidal Mapping for Improved Position Update Formula

In the original algorithm, the component vectors of wind in the position update formula are generated using pseudo-random numbers to create d-dimensional vectors. However, pseudo-random number generators may not provide adequate randomness and distribution. To address this limitation, this study employs sinusoidal mapping as an alternative to pseudo-random numbers for position updates. The enhanced formula is presented in Eq. (19):

$$x_{i+1} = x_i + a \times (u \times v) \times z + (2 - a)\left(p_g - x_i\right) \tag{19}$$

5.1.3 Quasi-Opposition-Based Learning (QOBL) Strategy

QOBL [41] builds upon the concept of Opposition-Based Learning (OBL) [42], aiming to enhance the search process efficiency and global optimization performance. Unlike conventional OBL, QOBL implements a more flexible adversarial strategy. Instead of merely adopting elements from the inverse solution, it incorporates perturbations or variations of adversarial elements. This approach facilitates a more adaptable exploration of the search space, thereby strengthening the algorithm's ability to conduct global searches and avoid entrapment in local optima.

Consider x as a point in the d-dimensional space, represented by the vector $x = (x_1, x_2, \ldots, x_d)$, where each component x_j is a real number within the interval $[a_j, b_j]$, with a_j and b_j being constants. The inverse solution is defined in d-dimensions and is expressed by the following Eqs. (20)–(22):

$$\widetilde{x}_j^q = \begin{cases} rand\left(M_j, \widetilde{x}_j\right), x_j < M_j \\ rand\left(\widetilde{x}_j, M_j\right), otherwise \end{cases} \tag{20}$$

$$\widetilde{x}_j = a_j + b_j - x_j \tag{21}$$

$$M_j = \frac{a_j + b_j}{2} \tag{22}$$

where $rand(M_j, \widetilde{x}_j)$ represents a uniform random number between M_j and $\widetilde{x}_j$, $\widetilde{x}_j$ signifies the inverse solution of x_j, and $\widetilde{x}_j^q$ denotes the proposed inverse solution of x_j.

If the termination condition is not met, each individual's fitness value is evaluated after each position update. Subsequently, the lowest-performing 10% of individuals are selected for the proposed reverse learning process. Following this, individuals with superior fitness values are incorporated into the population.

The AWCO pseudo-code is presented in Algorithm 1 below for reference.

Algorithm 1: AWCO

Input: N: population size, D: dimension of the solution, T: maximum iterations, UB: upper bound, LB: lower bound.
Output: The global optimum for each individual in the population.
1: Use Eq. (18) to initialize the position for each individual in the population.
2: **while** $count < T$ **do**
3: Denote the variable a by Eq. (16).
4: **for** $i = 1$ to N **do**
5: **if** $d_i \leq R$ **then**
6: Generation of wind speed ws and wind direction wd using Eqs. (11)–(13).
7: **else**
8: Generation of wind speed ws and wind direction wd using Eq. (10).
9: **end if**
10: Update the position of the population using Eqs. (14)–(19).
11: Calculate fitness values of population.
12: **end for**
13: Select the top 10% with poor fitness values for the proposed reverse learning through Eqs. (20)–(22).
14: Selection of small fitness values to be placed in the population.
15: **end while**

5.2 AWCO-Based Task Scheduling

The primary aim of task scheduling is to allocate specific tasks to VMs. In this context, the dimension d signifies the number of tasks, while n represents both the number of solutions and the population size within the algorithm. Each individual in the population corresponds to a potential solution. The optimal solution, which represents the most efficient resource allocation plan, is determined through the application of a fitness function.

To determine the initial globally optimal solution, the population must first be initialized, followed by solving the fitness function to ascertain the fitness value. Subsequently, the position of each generation of individuals is updated using the willow position, and the global optimal solution is refined. This process continues, with fitness values recalculated until the termination condition is met. The final globally optimal solution represents the ideal scheduling plan.

In cloud computing environments, tasks often present diverse priorities, variable execution time, and fluctuating resource requirements during the execution phase. Many conventional scheduling algorithms tend to converge on local optimal solutions, leading to suboptimal resource utilization and inefficient task execution. AWCO addresses these limitations by overcoming the constraints of conventional algorithms. It demonstrates an enhanced ability to approximate optimal solutions

efficiently. These improvements enable AWCO to effectively tackle task scheduling challenges in cloud computing environments. Consequently, AWCO facilitates the optimization of resource utilization, reduces task completion time, and enhances overall system performance.

6 Experiments

This section provides a preliminary evaluation of the AWCO algorithm's performance on CEC2014, comparing it with other established algorithms such as the GA, BA [43,44], Sinusoidal Cosine Algorithm (SCA) [45], TA, PSO, Rafflesia Optimization Algorithm (ROA) [46], and WCO. Following this comparison, AWCO's performance is assessed in two distinct heterogeneous cloud environments.

6.1 Experimental Setup

The CEC2014 benchmark function set, established by the 2014 IEEE Congress on Evolutionary Computation, consists of a series of test functions. These functions have been utilized to assess and compare the performance of various optimization algorithms, particularly those related to evolutionary computation and meta-heuristic algorithms. The test suite encompasses 30 test functions, including single-peaked (F1–F3), multi-peaked (F4–F16), mixed-peaked (F17–F22), and combinatorial-peaked (F23–F30) ones. This diversity aims to assess the algorithms' performance and adaptability across various problem types. This study presents an analysis of the comparative effectiveness of several algorithms, with particular emphasis on contrasting the AWCO algorithm with the original algorithm. The comparison involved executing each benchmark function 20 times. Each iteration was performed using CEC2014 with function dimensions of 10, 30, and 50, culminating in a total of 50,000 function evaluations.

Following comprehensive simulations of the CEC2014 test functions, practical application tests were conducted on the AWCO algorithm in an enhanced mixed cloud environment. This approach facilitated a comparison with other algorithms, including WCO, PSO, GA, BA, SCA, TA, and APPE. To ensure a fair comparison, identical initial conditions were maintained, with the population size and iteration times set to 100. The performance of the two independent heterogeneous environment evaluation algorithms was enhanced. In the initial improved environment (hereinafter referred to as Improved Environment 1), the *MIPS* and *storage* capacity values of the VMs were distinct, while the remaining factors remained constant. In the second improved environment (hereinafter referred to as Improved Environment 2), a random number generator was employed to select values for *RAM*, *MIPS*, *bandwidth*, and *storage* of the VMs, which were then assigned to specific ranges. The relevant parameters are presented in Tables 2 and 3. The costs associated with resource utilization were determined by the specifications delineated for the VMs. The resulting calculations yielded the following costs: $0.12, $0.13, $0.17, $0.48, $0.52, and $0.96 per hour [47].

Table 2: Experimental settings in Improved Environment 1

Cloud entities	Parameters	Values
Task	Num of task	30–300
	Length	100–1000

(Continued)

Table 2 (continued)

Cloud entities	Parameters	Values
Host	Num of host	2
	RAM	2048 MB
	Storage	1,000,000
	Bandwidth	10,000
Virtual machine	Num of VMs	15
	RAM	512 MB
	MIPS	100–1000
	Bandwidth	1000 MB
	Storage	1000–1,000,000 MB
	Size	10,000
	VVM	XUN
	Operating-system	Linux
	Num of CPUs	1
Data center	Num of DCs	2

Table 3: Experimental settings in Improved Environment 2

Cloud entities	Parameters	Values
Task	Num of task	30–300
	Length	100–1000
Host	Num of host	2
	RAM	20 GB
	Storage	1 TB
	Bandwidth	10 GB
Virtual machine	Num of VMs	25
	RAM	128–15,360 MB
	MIPS	256–30,720
	Bandwidth	128–15,360 MB
	Storage	128–15,360 MB
	Size	10 GB
	VVM	XUN
	Operating-system	Linux
	Num of CPUs	1
Data center	Num of DCs	2

6.2 Comparison Results of CEC2014 Benchmark Test Suite

Table 4 employs the following notations: "+" indicates superior performance of AWCO compared to other algorithms, "−" denotes inferior performance, and "=" signifies comparable performance. The AWCO algorithm demonstrates superior performance across all functions when compared to GA, SCA, TA, and WCO. Furthermore, it outperforms BA and ROA in 29 functions and PSO in 26 functions. AWCO exhibits superior performance in all single-peak functions, attributable to its implementation of sinusoidal mapping for position updates. The AWCO algorithm only shows marginal inferiority to PSO regarding F27. Additionally, AWCO demonstrates superior performance in the remaining mixed-peak and combinatorial functions, which often comprise multiple local and global optimal solutions. The QOBL strategy employed by AWCO effectively navigates around local optima, showcasing robust global search capabilities. Moreover, AWCO demonstrates an effective balance between exploration and exploitation processes, enabling thorough exploration of unvisited regions while accurately searching for optimal solutions near local optima.

Table 4: Simulation results of 10-D on CEC2014 benchmark functions

Function	GA	BA	SCA	TA	PSO	ROA	WCO	AWCO
F1	3.3692E+07	2.9623E+05	7.5325E+06	2.3451E+06	8.2691E+05	1.1591E+05	1.0122E+06	1.9733E+04
F2	1.4910E+09	3.9005E+05	7.1288E+08	2.1925E+04	4.1385E+05	6.6973E+03	6.2796E+06	2.0943E+03
F3	1.8185E+04	1.7564E+04	4.7005E+03	4.7589E+03	8.7381E+02	1.6748E+03	4.4211E+03	3.0023E+02
F4	5.8367E+02	4.0295E+02	4.6068E+02	4.3028E+02	4.0202E+02	4.2355E+02	4.2399E+02	4.1112E+02
F5	5.2016E+02	5.2039E+02	5.2040E+02	5.2014E+02	5.0567E+02	5.2003E+02	5.1962E+02	5.0518E+02
F6	6.0714E+02	6.0850E+02	6.0673E+02	6.0431E+02	6.0172E+02	6.0611E+02	6.0333E+02	6.0038E+02
F7	7.2539E+02	7.0070E+02	7.0994E+02	7.0031E+02	7.0062E+02	7.0040E+02	7.0105E+02	7.0022E+02
F8	8.4102E+02	8.4755E+02	8.3933E+02	8.2260E+02	8.1296E+02	8.0468E+02	8.2191E+02	8.0696E+02
F9	9.4750E+02	9.4962E+02	9.4139E+02	9.3008E+02	9.0436E+02	9.4847E+02	9.2371E+02	9.0801E+02
F10	1.3783E+03	2.2024E+03	2.0070E+03	1.4001E+03	1.0020E+03	1.7718E+03	1.5235E+03	1.2306E+03
F11	2.5164E+03	2.2526E+03	2.4614E+03	1.8839E+03	1.5159E+03	2.1689E+03	1.8979E+03	1.2485E+03
F12	1.2009E+03	1.2005E+03	1.2013E+03	1.2003E+03	1.2004E+03	1.2005E+03	1.2011E+03	1.2001E+03
F13	1.3008E+03	1.3005E+03	1.3006E+03	1.3004E+03	1.3002E+03	1.3004E+03	1.3004E+03	1.3001E+03
F14	1.4033E+03	1.4003E+03	1.4011E+03	1.4003E+03	1.4001E+03	1.4004E+03	1.4002E+03	1.4000E+03
F15	1.5571E+03	1.5063E+03	1.5088E+03	1.5021E+03	1.5030E+03	1.5031E+03	1.5038E+03	1.5007E+03
F16	1.6035E+03	1.6036E+03	1.6033E+03	1.6030E+03	1.6018E+03	1.6033E+03	1.6028E+03	1.6016E+03
F17	8.6229E+05	5.6940E+03	4.4810E+04	8.7665E+03	2.7036E+03	6.5333E+03	5.9134E+03	2.2692E+03
F18	3.2137E+06	1.2924E+04	1.9066E+04	1.1849E+04	2.2497E+03	1.1173E+04	1.8511E+04	2.7201E+03
F19	1.9146E+03	1.9054E+03	1.9049E+03	1.9040E+03	1.9016E+03	1.9051E+03	1.9041E+03	1.9016E+03
F20	5.1715E+05	7.8475E+03	5.2997E+03	4.1522E+03	2.1224E+03	5.2315E+03	2.4332E+03	2.0325E+03
F21	4.3903E+05	9.3234E+03	1.0926E+04	4.5448E+03	2.4545E+03	8.2155E+03	5.7204E+03	2.4131E+03
F22	2.4672E+03	2.3706E+03	2.2609E+03	2.2714E+03	2.3437E+03	2.3248E+03	2.2449E+03	2.2226E+03
F23	2.6668E+03	2.6295E+03	2.6418E+03	2.6299E+03	2.6341E+03	2.6295E+03	2.6302E+03	2.6100E+03
F24	2.5698E+03	2.5690E+03	2.5518E+03	2.5413E+03	2.5599E+03	2.5573E+03	2.5301E+03	2.5145E+03
F25	2.6987E+03	2.6991E+03	2.6972E+03	2.6908E+03	2.7017E+03	2.7012E+03	2.6473E+03	2.6155E+03
F26	2.7011E+03	2.7004E+03	2.7007E+03	2.7004E+03	2.7002E+03	2.7003E+03	2.7003E+03	2.7001E+03
F27	3.0619E+03	3.0539E+03	2.9829E+03	2.9017E+03	2.7061E+03	3.0412E+03	2.9227E+03	2.8216E+03
F28	3.5059E+03	3.3248E+03	3.2618E+03	3.2967E+03	3.9219E+03	3.2922E+03	3.5156E+03	3.1924E+03
F29	1.5440E+05	3.8737E+03	9.3586E+03	4.5525E+03	3.1562E+03	8.9931E+04	4.0903E+03	3.2211E+03
F30	9.9212E+03	4.6551E+03	4.7705E+03	4.9427E+03	5.1399E+03	4.4713E+03	5.4584E+03	3.6746E+03
+/=/−	30/0/0	29/0/1	30/0/0	30/0/0	26/1/3	29/0/1	30/0/0	

Table 5 presents a comparative analysis of the algorithms across 30 dimensions. The data in the table demonstrates that the AWCO algorithm outperforms GA, SCA, TA, and WCO in various scenarios, including single-peak, multi-peak, and mixed-peak problems. The results indicate the algorithm's robustness in handling diverse problem types, from simple single-peak problems to multi-peak problems with multiple local optima and mixed-peak problems with complex structures. Moreover, AWCO exhibits enhanced performance compared to BA and ROA in single-peak and mixed-peak scenarios. In the mixed-peak function, AWCO slightly underperforms PSO in F22 but surpasses the other algorithms in the remaining cases. These findings suggest that as problem dimensionality increases, AWCO demonstrates a superior ability to avoid local optima, making it more suitable for complex optimization problems.

Table 5: Simulation results of 30-D on CEC2014 benchmark functions

Function	GA	BA	SCA	TA	PSO	ROA	WCO	AWCO
F1	5.1498E+08	7.5536E+06	3.9371E+08	7.6362E+07	6.8864E+06	1.2146E+07	5.4967E+07	1.7146E+06
F2	2.7862E+10	8.8175E+06	2.3465E+10	8.7470E+07	7.5377E+09	1.1438E+04	3.3709E+08	9.6693E+03
F3	8.4302E+04	1.1740E+05	5.3086E+04	6.5284E+04	1.6845E+04	2.3347E+04	4.9244E+04	9.4146E+02
F4	2.3322E+03	4.5183E+02	2.1548E+03	7.1887E+02	4.5298E+02	5.2681E+02	6.4657E+02	4.9585E+02
F5	5.2029E+02	5.2101E+02	5.2101E+02	5.2079E+02	5.2017E+02	5.2015E+02	5.2102E+02	5.2004E+02
F6	6.3060E+02	6.3640E+02	6.3633E+02	6.2768E+02	6.3399E+02	6.3281E+02	6.2555E+02	6.1364E+02
F7	9.0180E+02	7.0108E+02	8.8772E+02	7.0156E+02	7.0435E+02	7.0002E+02	7.0369E+02	7.0003E+02
F8	1.0390E+03	1.0481E+03	1.0652E+03	9.4564E+02	9.6607E+02	8.4306E+02	9.7247E+02	8.5985E+02
F9	1.1494E+03	1.2209E+03	1.1897E+03	1.0900E+03	1.0587E+03	1.2099E+03	1.1050E+03	9.6646E+02
F10	5.0638E+03	5.3235E+03	7.4304E+03	4.6908E+03	3.2765E+03	5.1668E+03	5.6752E+03	3.7912E+03
F11	6.9880E+03	5.5300E+03	8.3974E+03	5.5327E+03	5.0801E+03	5.7445E+03	6.7904E+03	4.1396E+03
F12	1.2008E+03	1.2013E+03	1.2029E+03	1.2012E+03	1.2011E+03	1.2013E+03	1.2029E+03	1.2005E+03
F13	1.3038E+03	1.3005E+03	1.3037E+03	1.3005E+03	1.3002E+03	1.3006E+03	1.3006E+03	1.3004E+03
F14	1.4778E+03	1.4003E+03	1.4639E+03	1.4003E+03	1.4743E+03	1.4004E+03	1.4005E+03	1.4001E+03
F15	1.9787E+04	1.5388E+03	1.5040E+04	1.5309E+03	1.1922E+04	1.5354E+03	1.5312E+03	1.5079E+03
F16	1.6123E+03	1.6132E+03	1.6131E+03	1.6124E+03	1.6133E+03	1.6130E+03	1.6125E+03	1.6102E+03
F17	3.3878E+07	4.3109E+05	9.4532E+06	2.1658E+06	1.3232E+07	7.3447E+05	2.0768E+06	1.5649E+05
F18	8.1982E+08	2.1737E+05	2.3357E+08	5.7248E+04	2.6385E+07	1.3499E+04	3.6448E+06	2.8808E+03
F19	2.1309E+03	1.9233E+03	2.0181E+03	1.9441E+03	1.9166E+03	1.9365E+03	1.9213E+03	1.9136E+03
F20	6.8129E+04	3.3610E+04	3.4939E+04	2.1341E+04	2.6896E+03	1.6412E+04	1.5870E+04	2.3089E+03
F21	1.6261E+07	2.1297E+05	3.3913E+06	4.6685E+05	3.2140E+05	4.8655E+05	6.4345E+05	6.4855E+04
F22	3.3230E+03	3.2712E+03	3.1530E+03	2.7761E+03	2.4477E+03	3.0662E+03	2.7176E+03	2.4693E+03
F23	2.7416E+03	2.6156E+03	2.6969E+03	2.6545E+03	2.7299E+03	2.6180E+03	2.6461E+03	2.6178E+03
F24	2.6016E+03	2.6661E+03	2.6140E+03	2.6546E+03	2.6471E+03	2.6589E+03	2.6442E+03	2.6255E+03
F25	2.7044E+03	2.7301E+03	2.7342E+03	2.7242E+03	2.7245E+03	2.7227E+03	2.7200E+03	2.7087E+03
F26	2.7291E+03	2.7105E+03	2.7030E+03	2.7007E+03	2.7042E+03	2.7006E+03	2.7006E+03	2.7003E+03
F27	3.7492E+03	3.8063E+03	3.7772E+03	3.5160E+03	4.0892E+03	3.6577E+03	3.1342E+03	3.2639E+03
F28	6.1691E+03	6.4960E+03	5.4214E+03	5.5251E+03	6.5688E+03	4.7911E+03	6.3290E+03	5.5042E+03
F29	4.9573E+07	3.1303E+03	2.5734E+07	3.4876E+06	2.7717E+05	1.0927E+07	2.2662E+08	2.3920E+06
F30	5.9544E+05	1.5817E+04	4.5924E+05	1.0550E+05	2.0503E+05	2.7140E+04	1.0143E+05	1.3553E+04
+/=/−	28/0/2	27/0/3	29/0/1	30/0/0	25/0/5	27/0/3	29/0/1	

Table 6 presents a comparative analysis of the algorithms across 50 dimensions. Among the 50 dimensions, AWCO exhibits 28 test functions with superior performance when compared to SCA, 26 test functions surpassing GA and TA, and 25 test functions outperforming WCO. These results indicate that the AWCO algorithm demonstrates effective global search capabilities. Except for F12 and F16, which show performance characteristics similar to those of SCA, all functions exhibit

superior performance in AWCO compared to SCA. Moreover, AWCO outperforms the remaining algorithms by a margin exceeding 80% in high-dimensional problems, suggesting its proficiency in navigating high-dimensional spaces. High-dimensional problems typically present complex search spaces with numerous local optima. AWCO employs the QOBL strategy to maintain population diversity throughout the search process, preventing solution space reduction due to over-aggregation and ensuring adequate exploration capabilities in high-dimensional problems. The commendable performance of AWCO demonstrates its ability to efficiently navigate these complex spaces, avoiding the pitfalls of local optima. AWCO outperforms other classical algorithms on most test functions, thus exhibiting robustness and consistency.

Table 6: Simulation results of 50-D on CEC2014 benchmark functions

Function	GA	BA	SCA	TA	PSO	ROA	WCO	AWCO
F1	7.7200E+08	1.3200E+07	9.5500E+08	1.9600E+08	2.8000E+07	3.3500E+07	1.4000E+08	3.3000E+06
F2	6.5200E+10	2.8100E+07	6.6800E+10	2.8900E+09	1.6500E+09	2.7400E+05	1.7200E+09	1.7800E+05
F3	1.2200E+05	1.5000E+05	1.2700E+05	1.1200E+05	5.6100E+04	6.8400E+04	7.9800E+04	6.8800E+03
F4	1.0900E+04	5.3100E+02	1.1500E+04	1.5000E+03	6.7500E+02	5.9600E+02	1.0300E+03	5.5900E+02
F5	5.2100E+02	5.2100E+02	5.2100E+02	5.2100E+02	5.1300E+02	5.2000E+02	5.2100E+02	5.2000E+02
F6	6.6000E+02	6.6800E+02	6.7100E+02	6.5500E+02	6.2400E+02	6.6500E+02	6.5100E+02	6.4100E+02
F7	1.2600E+03	7.0100E+02	1.3300E+03	7.2800E+02	7.2000E+02	7.0000E+02	7.2100E+02	7.0100E+02
F8	1.2800E+03	1.2700E+03	1.3500E+03	1.1700E+03	9.7800E+02	9.2600E+02	1.2200E+03	9.6300E+02
F9	1.3900E+03	1.4900E+03	1.5000E+03	1.3600E+03	1.1300E+03	1.5100E+03	1.3700E+03	1.0800E+03
F10	9.9200E+03	9.2500E+03	1.4200E+04	8.3700E+03	8.2100E+03	9.2000E+03	1.0800E+04	5.9800E+03
F11	1.2500E+04	9.2500E+03	1.5200E+04	1.0200E+04	8.1400E+03	9.8700E+03	1.3200E+04	7.6600E+03
F12	1.2000E+03	1.2000E+03	1.2000E+03	1.2000E+03	1.2000E+03	1.2000E+03	1.2000E+03	1.2000E+03
F13	1.3100E+03	1.3000E+03	1.3100E+03	1.3000E+03	1.3000E+03	1.3000E+03	1.3000E+03	1.3000E+03
F14	1.5400E+03	1.4000E+03	1.5700E+03	1.4000E+03	1.4000E+03	1.4000E+03	1.4000E+03	1.4000E+03
F15	3.9800E+05	1.5700E+03	3.9000E+05	2.1700E+03	1.5500E+03	1.6100E+03	2.0800E+03	1.5800E+03
F16	1.6200E+03	1.6200E+03	1.6200E+03	1.6200E+03	1.6200E+03	1.6200E+03	1.6200E+03	1.6200E+03
F17	1.0600E+08	1.1200E+06	8.2100E+07	1.2600E+07	7.6200E+05	2.3600E+06	7.9700E+06	6.2100E+05
F18	4.3000E+09	1.0700E+06	2.3300E+09	6.8600E+06	3.5800E+07	6.0500E+03	1.8300E+07	2.7500E+03
F19	2.4200E+03	1.9500E+03	2.2700E+03	2.0000E+03	1.9400E+03	1.9600E+03	1.9900E+03	1.9400E+03
F20	9.3800E+04	4.8600E+04	7.4500E+04	4.0100E+04	1.0300E+04	3.6300E+04	4.0800E+04	2.8400E+03
F21	3.1800E+07	8.3000E+05	2.0500E+07	4.3900E+06	9.0300E+05	1.4100E+06	5.0000E+06	3.0600E+05
F22	5.2900E+03	4.1500E+03	4.9100E+03	3.4900E+03	2.6800E+03	3.9700E+03	3.7600E+03	3.2900E+03
F23	3.3100E+03	2.6500E+03	3.1600E+03	2.7700E+03	3.2000E+03	2.6600E+03	2.7500E+03	2.6700E+03
F24	2.6700E+03	2.7700E+03	2.7600E+03	2.7200E+03	2.7200E+03	2.7200E+03	2.7200E+03	2.6800E+03
F25	2.7100E+03	2.7700E+03	2.7900E+03	2.7600E+03	2.7600E+03	2.7600E+03	2.7500E+03	2.7100E+03
F26	2.7900E+03	2.7200E+03	2.7100E+03	2.7600E+03	2.8100E+03	2.7200E+03	2.7000E+03	2.7000E+03
F27	4.6400E+03	5.1000E+03	4.9200E+03	4.4900E+03	5.4300E+03	4.8200E+03	4.3400E+03	4.1500E+03
F28	1.0300E+04	1.1900E+04	1.0000E+04	9.6300E+03	1.4300E+04	7.7800E+03	1.0100E+04	7.4500E+03
F29	4.2900E+08	3.1700E+03	2.8400E+08	1.3000E+08	2.7800E+08	1.6900E+08	4.9400E+08	1.8800E+04
F30	5.1100E+06	1.6500E+04	3.5600E+06	6.4700E+05	9.8800E+06	5.6900E+04	3.9800E+05	7.0800E+04
+/=/−	26/3/1	20/5/5	28/2/0	26/4/0	21/5/4	22/5/3	25/5/0	

To further demonstrate the superior performance of AWCO, we conducted convergence performance tests on 30 benchmark functions in CEC2014, including convergence graphs of 10-D, 30-D, and 50-D. Each dimension was tested on functions exhibiting single, multiple, mixed, and combined peaks, enabling a comprehensive evaluation of AWCO's convergence and robustness. Fig. 2 illustrates the convergence performance of AWCO on the 10-D dataset. Compared to other algorithms such as GA, WCO, and SCA, the results show that AWCO exhibits superior performance, particularly for

multiple and combined peaks. This is attributed to AWCO's utilization of a reverse learning strategy to identify the optimal solution, allowing it to converge to the global optimal solution more rapidly.

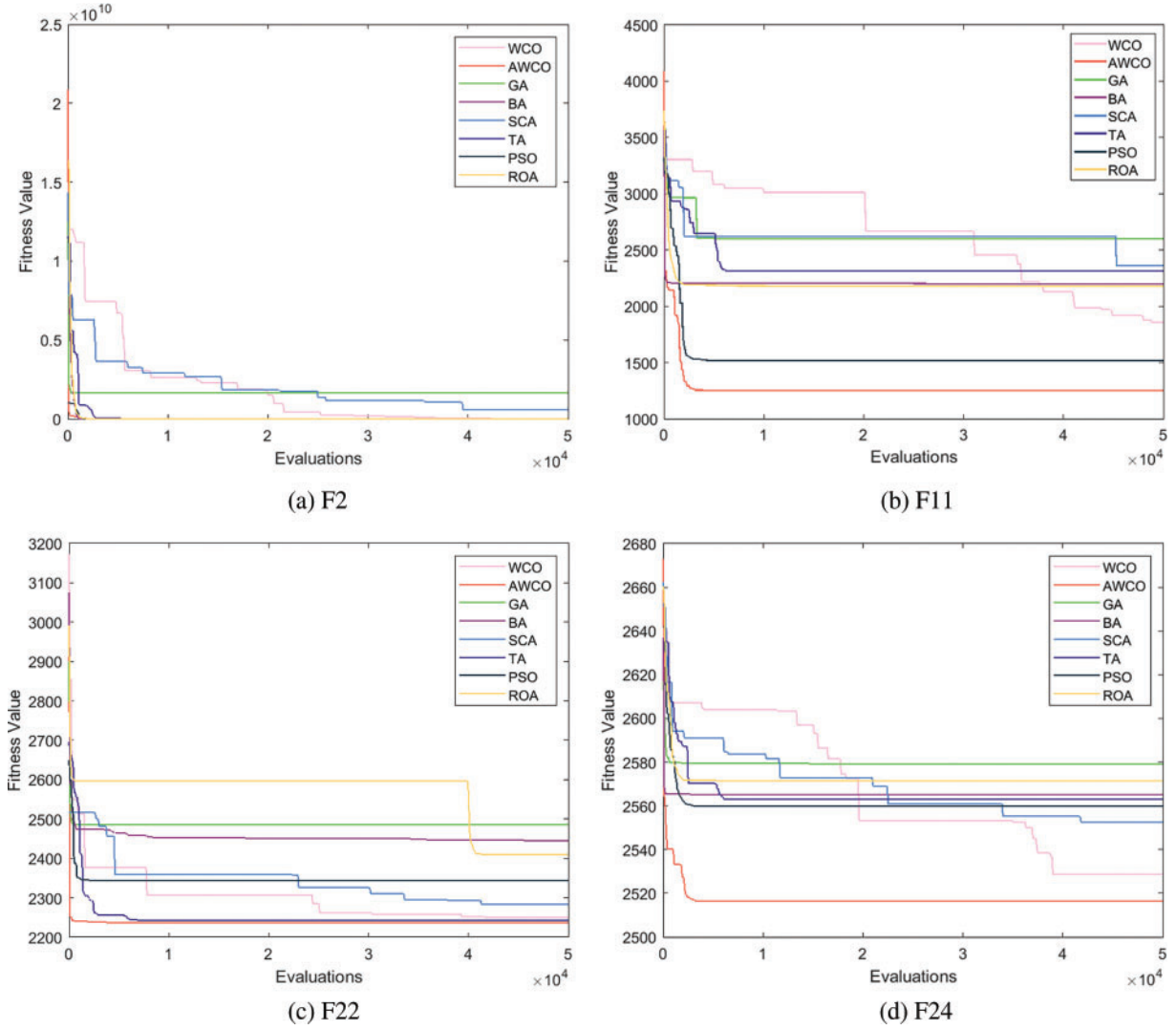

Figure 2: Convergence graph of CEC2014 benchmark functions on 10-D

Fig. 3 illustrates the convergence performance of AWCO on 30-D. AWCO demonstrates superior performance compared to alternative algorithms, particularly in processing data related to functions with multiple and combined peaks. The AWCO algorithm converges notably faster than other algorithms for most functions. AWCO's ability to achieve low fitness values with limited evaluations highlights its exceptional global search capability and efficiency in identifying optimal solutions. The graph displays several fitness curves, with AWCO's curve being the most prominent, especially in the early stages. The sinusoidal mapping enables a more uniform initial population distribution across the solution space, facilitating rapid identification of high-quality solutions in the algorithm's initial phase. The AWCO curve exhibits minimal fluctuation, indicating stability throughout the optimization process and reduced susceptibility to significant performance variations.

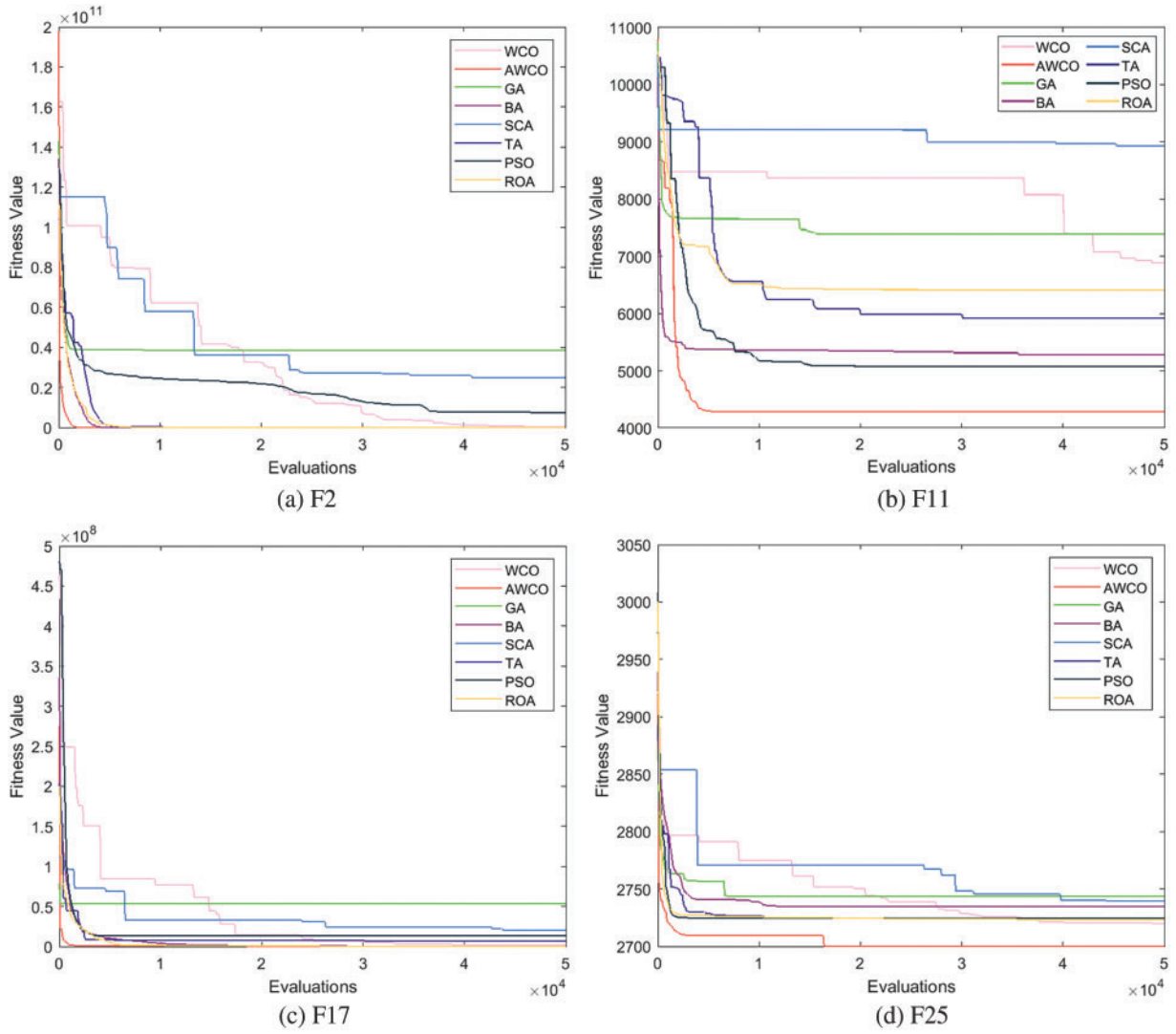

Figure 3: Convergence graph of CEC2014 benchmark functions on 30-D

Fig. 4 illustrates the convergence performance of AWCO on the 50-dimensional data set. While the performance of many algorithms may deteriorate with increasing dimensionality, AWCO consistently demonstrates superior convergence and achieves lower fitness values more quickly than other algorithms for most functions. This suggests that AWCO possesses enhanced global search capabilities and convergence speed when addressing high-dimensional problems. Although increased dimensionality typically results in performance fluctuations for many algorithms, AWCO's fitness curves remain smooth with minimal variations, indicating its robustness is well-preserved even in high-dimensional problems. This stability is attributed to the sinusoidal mapping, which enhances the stochastic nature of positional updates while maintaining a uniform distribution of population initialization, thus providing a robust foundation for the algorithm's exploration from the outset. AWCO's superiority in various aspects renders it a noteworthy and applicable algorithm.

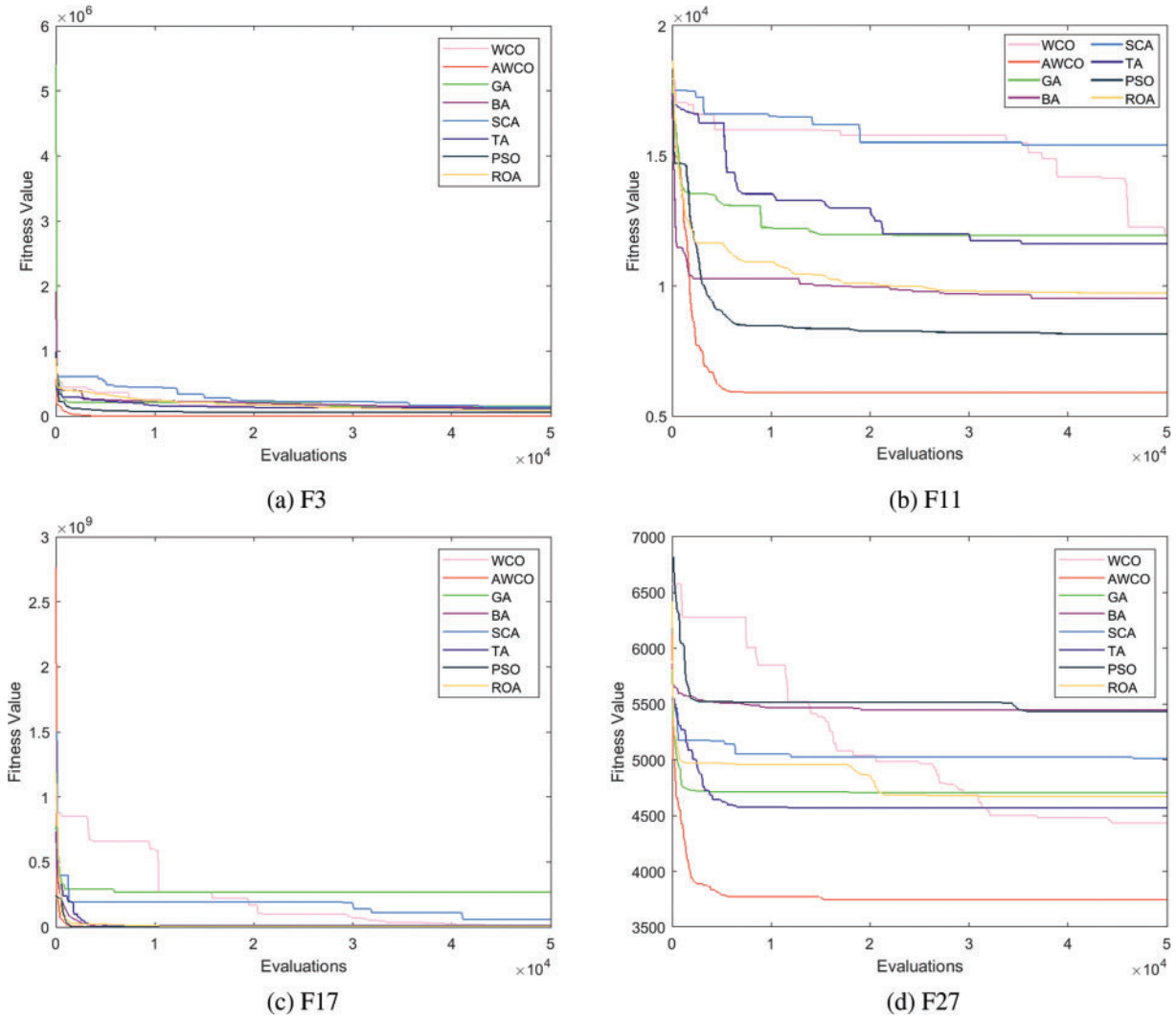

Figure 4: Convergence graph of CEC2014 benchmark functions on 50-D

6.3 Experimental Results of Task Scheduling in Improved Heterogeneous Cloud Environments

The APPE algorithm was employed to evaluate the performance of both the original and improved heterogeneous cloud environments, and the results are presented in Table 7. The objective function values of Improved Environments 1 and 2 are significantly lower than those of the original environments, indicating that the improved environments are more conducive to optimization and demonstrate superior performance across all dimensions. This suggests that in these improved environments, the optimization algorithm can identify outcomes more closely aligned with the global optimal solution. Since the goal is to minimize the objective function value, the lower values in the improved environments indicate greater efficiency. In dimensions ranging from 50-D to 300-D, the improved environments (1 and 2) consistently exhibit lower objective function values, demonstrating that the enhanced cloud environments maintain their optimization capability for high-dimensional problems.

Table 7: Comparison of heterogeneous cloud environments

Dimension	Original Environment 1	Original Environment 2	Improved Environment 1	Improved Environment 2
50-D	0.1251	0.0014	38.1194	2.8414
100-D	0.1557	0.0009	44.5321	3.0891
200-D	0.103	0.0006	35.9361	2.6448
300-D	0.1318	0.001	46.6529	4.2776

To rigorously evaluate the effectiveness of the AWCO algorithm, 30 trials were conducted for each of the following algorithms: WCO, PSO, GA, BA, SCA, TA, and APPE. These trials were performed in the improved heterogeneous cloud environment, and the resulting averages were compared across dimensions of 30-D, 50-D, 100-D, 200-D, and 300-D, where the dimension refers to the task volume. The comparative analysis of these averages is presented in Tables 8 and 9. The data in these tables indicate that AWCO consistently demonstrates the lowest mean value compared to other algorithms, suggesting superior efficiency and global search capability in optimizing cloud-based tasks. Furthermore, the values in Table 9 (Improved Environment 2) are generally lower than those in Table 8 (Improved Environment 1), particularly at smaller dimensions (30-D and 50-D). This implies more effective task allocation management in the context of enhanced load capacity. In this environment, the reduced average values across all algorithms indicate improved efficiency in resource management and task scheduling.

Table 8: Comparison of mean values of various dimensions in Improved Environment 1

Dimension	WCO	PSO	GA	BA	SCA	TA	APPE	AWCO
30-D	25.2	21.07	156.03	23.82	24.91	24.85	26.65	12.29
50-D	30.67	32.14	67.12	32.93	32.7	44.25	38.11	13.48
100-D	35.15	43.38	54.63	36.11	37.13	45.11	44.53	21.08
200-D	61.64	40.37	226.31	57.37	46.91	40.82	35.94	26.12
300-D	91.58	72.28	229.47	70.1	98.18	58.11	46.65	38.49

Table 9: Comparison of mean values of various dimensions in Improved Environment 2

Dimension	WCO	PSO	GA	BA	SCA	TA	APPE	AWCO
30-D	0.48	1.61	2.2	0.87	1.53	2.02	2.2	0.45
50-D	0.07	2.85	0.1	3.03	2.28	2.91	2.84	0.06
100-D	2.58	3.02	6.49	3.28	4.68	2.54	3.09	1.44
200-D	1.96	3.47	8.55	2.75	3.37	2.49	2.64	1.78
300-D	4.86	4.37	20.75	4.39	5.17	4.07	4.28	2.97

At elevated task counts (e.g., 200-D and 300-D tasks), the mean value of AWCO is significantly lower than that of the other algorithms. This suggests that AWCO not only performs optimally

with low task volumes but also demonstrates notable adaptability and robustness in high-volume task scenarios. The sinusoidal mapping strategy and QOBL approach contribute to the algorithm's enhanced flexibility within the solution space. The dynamic adjustment of task scheduling, based on feedback during the search process, enables the algorithm to utilize high-quality solutions more efficiently while maintaining the ability to explore new solutions.

Figs. 5 and 6 illustrate the convergence of the algorithm in both improved environments with varying task volumes. The convergence plots demonstrate three scenarios: low, medium, and high task volumes. The BA, TA, and APPE algorithms exhibit greater consistency across all task volumes, although they do not perform as well as the AWCO algorithm under high task volumes. The AWCO algorithm demonstrates superior performance under varying task volumes, particularly in high task volumes, where it shows higher convergence speeds and final convergence values. While the other algorithms may perform better under specific task volumes, they generally do not match the effectiveness of the AWCO algorithm. A comparison of the AWCO algorithm with the other algorithms reveals its notable advantage in terms of convergence speed and capacity to handle different task volumes and heterogeneous environments. This indicates that the AWCO algorithm is more efficient and robust than similar algorithms for task scheduling. Moreover, AWCO not only demonstrates notable performance under a single task volume but also exhibits adaptability in diverse heterogeneous environments. This positions the AWCO algorithm as an effective tool for addressing task scheduling challenges and a reliable solution for task allocation in complex environments.

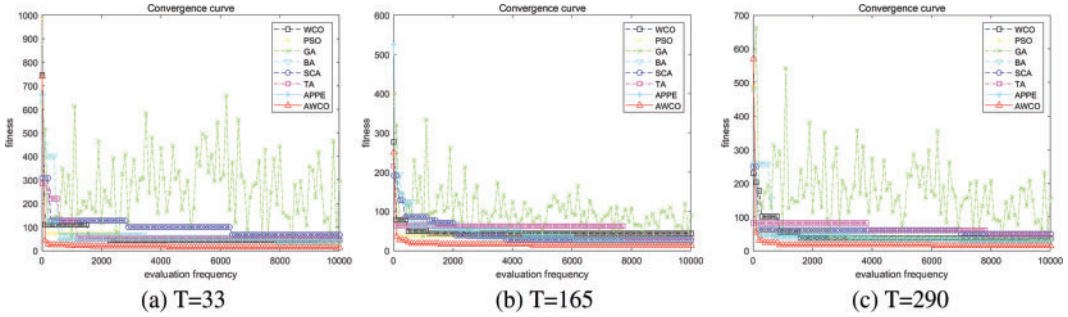

Figure 5: Comparison of convergence for different tasks in Improved Environment 1

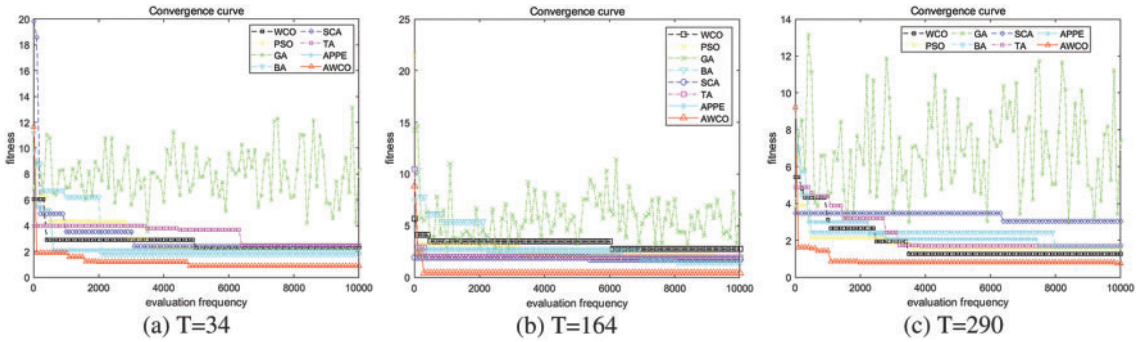

Figure 6: Comparison of convergence for different tasks in Improved Environment 2

To effectively demonstrate the efficacy of the AWCO algorithm in cloud environments, experiments were conducted to analyze the makespan, cost, and load. Figs. 7 and 8 illustrate the experiments

conducted in homogeneous (Improved Environment 1) and heterogeneous (Improved Environment 2) cloud environments, respectively. The number of tasks varied between 30 and 300, with a step size of 90. The experimental results presented in Fig. 7 demonstrate that the AWCO algorithm exhibits notable superiority over the other meta-heuristic algorithms in terms of cost, completion time, and load as the number of tasks increases. The ability to achieve the lowest cost, shortest completion time, and most effective load balancing is of great significance in task scheduling. This is attributed to the AWCO algorithm's effective exploration and development capabilities, enabling it to identify more suitable solutions for task allocation. The experimental results in Fig. 8 are analogous to those in Fig. 7, with the AWCO algorithm again demonstrating advantages over the other algorithms. In the heterogeneous Improved Environment 2, the completion time is reduced in comparison to the cost. This is because a greater number of VMs can be assigned to the tasks, thereby improving efficiency.

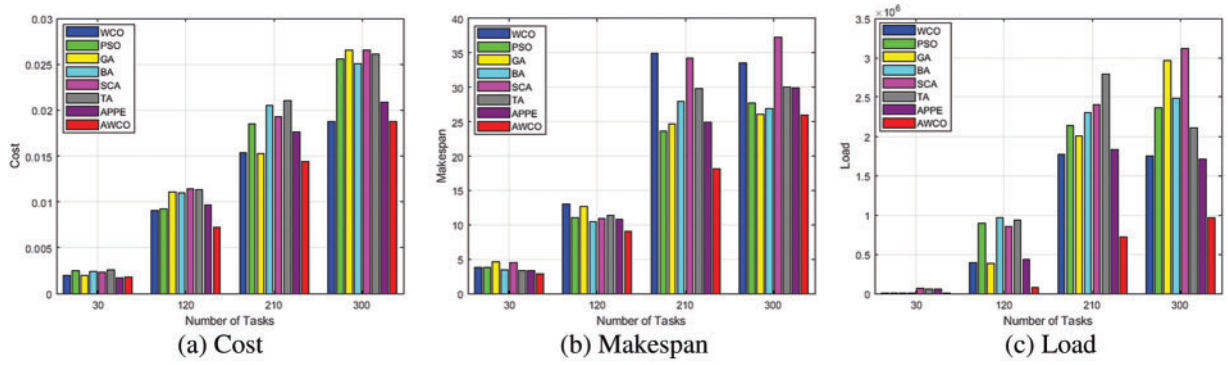

Figure 7: Comparison of makespan, cost, and load balancing level of each algorithm for different task volumes in Improved Environment 1

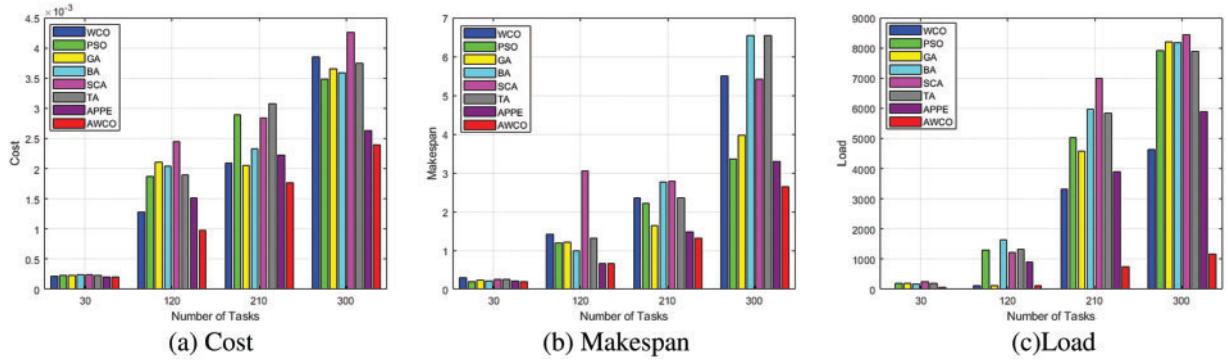

Figure 8: Comparison of makespan, cost, and load balancing level of each algorithm for different task volumes in Improved Environment 2

7 Conclusions

To enhance global search capability, this study implements the QOBL strategy in the proposed AWCO algorithm, which proves effective in achieving the desired outcome. Additionally, the incorporation of chaotic mapping improves the convergence rate, strengthens its exploration and exploitation capabilities, and facilitates rapid and accurate identification of the optimal solution. In the

CEC2014 benchmark functions, the AWCO algorithm demonstrates superior performance compared to the original WCO algorithm, effectively addressing the latter's convergence and instability issues. Moreover, it yields favorable results in task scheduling within cloud computing environments. These improvements offer a more viable and reliable solution for the practical implementation of WCO.

While AWCO demonstrates effectiveness in solving task scheduling problems, it may not be the optimal solution for highly complex and multimodal issues. In such scenarios, AWCO might struggle to efficiently escape local optima, potentially limiting its global search capabilities. To address these limitations, future research should focus on incorporating an adaptive mechanism to dynamically adjust AWCO parameters and enhance its ability to navigate complex problem spaces. Although this study has provided valuable insights into the task scheduling algorithm's efficacy, it primarily emphasizes algorithmic optimization and has not fully examined the impacts of cloud and data center dynamics on task scheduling in cloud computing environments. This includes crucial factors such as dynamic resource management, data transmission overheads, and energy consumption optimization. Future investigations could integrate cloud computing infrastructure characteristics with a more comprehensive analysis of data center resource fluctuations and network environment dynamics. This approach would better align with task scheduling requirements in real-world cloud computing scenarios, thereby improving the algorithm's practicality and efficiency.

Acknowledgement: The authors would like to express their gratitude to all the anonymous reviewers and the editorial team for their valuable feedback and suggestions.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Jeng-Shyang Pan, Na Yu; data collection: Na Yu; analysis and interpretation of results: An-Ning Zhang; draft manuscript preparation: Shu-Chuan Chu, Bin Yan, Junzo Watada. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, Shu-Chuan Chu, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] R. Kaewpuang, D. Niyato, P. Wang, and E. Hossain, "A framework for cooperative resource management in mobile cloud computing," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2685–2700, 2013. doi: 10.1109/JSAC.2013.131209.

[2] J. Huang, W. Susilo, F. Guo, G. Wu, Z. Zhao and Q. Huang, "An anonymous authentication system for pay-as-you-go cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1280–1291, 2020. doi: 10.1109/TDSC.2020.3007633.

[3] B. Calder *et al.*, "Windows azure storage: A highly available cloud storage service with strong consistency," in *Proc. Twenty-Third ACM Symp. Oper. Syst. Princ.*, Cascais, Portugal, 2011, pp. 143–157.

[4] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, no. 4, pp. 407–415, 2019. doi: 10.1016/j.future.2018.09.014.

[5] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends," *Swarm Evol. Comput.*, vol. 62, no. 3, 2021, Art. no. 100841. doi: 10.1016/j.swevo.2021.100841.
[6] N. Mansouri *et al.*, "An efficient task scheduling based on seagull optimization algorithm for heterogeneous cloud computing platforms," *Int. J. Eng.*, vol. 35, no. 2, pp. 433–450, 2022. doi: 10.5829/IJE.2022.35.02B.20.
[7] K. Efe, "Heuristic models of task assignment scheduling in distributed systems," *Computer*, vol. 15, no. 6, pp. 50–56, 1982. doi: 10.1109/MC.1982.1654050.
[8] J. Huang, C. Lin, and B. Cheng, "Energy efficient speed scaling and task scheduling for distributed computing systems," *Chin. J. Electron.*, vol. 24, no. 3, pp. 468–473, 2015. doi: 10.1049/cje.2015.07.005.
[9] D. S. Hochba, "Approximation algorithms for np-hard problems," *ACM Sigact News*, vol. 28, no. 2, pp. 40–52, 1997. doi: 10.1145/261342.571216.
[10] Z. Beheshti and S. M. H. Shamsuddin, "A review of population-based meta-heuristic algorithms," *Int. J. Adv. Soft Comput. Appl.*, vol. 5, no. 1, pp. 1–35, 2013.
[11] M. Hubálovská, Š. Hubálovský, and P. Hubálovský, "Botox optimization algorithm: A new human-based metaheuristic algorithm for solving optimization problems," *Biomimetics*, vol. 9, no. 3, 2024, Art. no. 137. doi: 10.3390/biomimetics9030137.
[12] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm Evol. Comput.*, vol. 33, no. 11, pp. 1–17, 2017. doi: 10.1016/j.swevo.2016.12.005.
[13] H. Ma, S. Shen, M. Yu, Z. Yang, M. Fei and H. Zhou, "Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey," *Swarm Evol. Comput.*, vol. 44, no. 10, pp. 365–387, 2019. doi: 10.1016/j.swevo.2018.04.011.
[14] J. Tang, G. Liu, and Q. Pan, "A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 10, pp. 1627–1643, 2021. doi: 10.1109/JAS.2021.1004129.
[15] F. Marini and B. Walczak, "Particle swarm optimization (PSO). A tutorial," *Chemometr. Intell. Lab. Syst.*, vol. 149, pp. 153–165, 2015. doi: 10.1016/j.chemolab.2015.08.020.
[16] C. -L. Huang and J. -F. Dun, "A distributed PSO-SVM hybrid system with feature selection and parameter optimization," *Appl. Soft Comput.*, vol. 8, no. 4, pp. 1381–1391, 2008. doi: 10.1016/j.asoc.2007.10.007.
[17] A. R. Jordehi, "Enhanced leader PSO (ELPSO): A new PSO variant for solving global optimisation problems," *Appl. Soft Comput.*, vol. 26, pp. 401–417, 2015. doi: 10.1016/j.asoc.2014.10.026.
[18] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, no. 2–3, pp. 243–278, 2005. doi: 10.1016/j.tcs.2005.05.020.
[19] J. H. Holland, "Genetic algorithms," *Sci. Am.*, vol. 267, no. 1, pp. 66–73, 1992. doi: 10.1145/114055.114056.
[20] J. Vasconcelos, J. A. Ramirez, R. Takahashi, and R. Saldanha, "Improvements in genetic algorithms," *IEEE Trans. Magn.*, vol. 37, no. 5, pp. 3414–3417, 2001. doi: 10.1109/20.952626.
[21] Q. -Y Yang, S. -C. Chu, A. Liang, and J. -S. Pan, "Tumbleweed algorithm and its application for solving location problem of logistics distribution center," in *Genet. Evol. Comput.: Proc. Fourteenth Int. Conf. Genet. Evol. Comput.*, Jilin, China, Springer, 2022, pp. 641–652.
[22] A. Seyyedabbasi and F. Kiani, "Sand cat swarm optimization: A nature-inspired algorithm to solve global optimization problems," *Eng. Comput.*, vol. 39, no. 4, pp. 2627–2651, 2023. doi: 10.1007/s00366-022-01604-x.
[23] Y. Kumar and P. K. Singh, "Improved cat swarm optimization algorithm for solving global optimization problems and its application to clustering," *Appl. Intell.*, vol. 48, no. 9, pp. 2681–2697, 2018. doi: 10.1007/s10489-017-1096-8.
[24] Y. He, X. Xue, and S. Zhang, "Using artificial bee colony algorithm for optimizing ontology alignment," *J. Inf. Hiding Multim. Signal Process*, vol. 8, no. 4, pp. 766–773, 2017.
[25] J. -S. Pan, S. -Q. Zhang, S. -C. Chu, H. -M. Yang, and B. Yan, "Willow catkin optimization algorithm applied in the TDOA-FDOA joint location problem," *Entropy*, vol. 25, no. 1, 2023, Art. no. 171. doi: 10.3390/e25010171.

[26] R. M. Alguliyev and R. G. Alakbarov, "Integer programming models for task scheduling and resource allocation in mobile cloud computing," *Int. J. Comput. Netw. Inform. Security*, vol. 15, no. 5, pp. 13–16, 2023. doi: 10.5815/ijcnis.2023.05.02.

[27] Y. Fang, F. Wang, and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing," in *Web Inf. Syst. Min.: Int. Conf., WISM 2010*, Sanya, China, Springer, 2010, pp. 271–277.

[28] S. Gupta *et al.*, "Efficient prioritization and processor selection schemes for heft algorithm: A makespan optimizer for task scheduling in cloud environment," *Electronics*, vol. 11, no. 16, 2022, Art. no. 2557. doi: 10.3390/electronics11162557.

[29] C. S. Pawar and R. B. Wagh, "Priority based dynamic resource allocation in cloud computing," in *2012 Int. Symp. Cloud Serv. Comput.*, Mangalore, India, IEEE, 2012, pp. 1–6.

[30] Z. Wu, X. Liu, Z. Ni, D. Yuan, and Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems," *J. Supercomput.*, vol. 63, no. 1, pp. 256–293, 2013. doi: 10.1007/s11227-011-0578-4.

[31] M. Chhabra and S. Basheer, "Recent task scheduling-based heuristic and meta-heuristics methods in cloud computing: A review," in *2022 5th Int. Conf. Contemp. Comput. Inf. (IC3I)*, Uttar Pradesh, India, IEEE, 2022, pp. 2236–2242.

[32] X. Chen *et al.*, "A woa-based optimization approach for task scheduling in cloud computing systems," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3117–3128, 2020. doi: 10.1109/JSYST.2019.2960088.

[33] Y. Changtian and Y. Jiong, "Energy-aware genetic algorithms for task scheduling in cloud computing," in *2012 Seventh China Grid Annu. Conf.*, Beijing, China, IEEE, 2012, pp. 43–48.

[34] X. Sheng and Q. Li, "Template-based genetic algorithm for qos-aware task scheduling in cloud computing," in *2016 Int. Conf. Adv. Cloud Big Data (CBD)*, Chengdu, China, IEEE, 2016, pp. 25–30.

[35] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural Comput. Appl.*, vol. 33, no. 19, pp. 13075–13088, 2021. doi: 10.1007/s00521-021-06002-w.

[36] N. Mansouri, B. M. H. Zade, and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Comput. Ind. Eng.*, vol. 130, no. 5, pp. 597–633, 2019. doi: 10.1016/j.cie.2019.03.006.

[37] Y. Moon, H. Yu, J. -M. Gil, and J. Lim, "A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments," *Hum. Centric Comput. Inf. Sci.*, vol. 7, no. 1, pp. 1–10, 2017. doi: 10.1186/s13673-017-0109-2.

[38] A. -N. Zhang, S. -C. Chu, P. -C. Song, H. Wang, and J. -S. Pan, "Task scheduling in cloud computing environment using advanced phasmatodea population evolution algorithms," *Electronics*, vol. 11, no. 9, 2022, Art. no. 1451. doi: 10.3390/electronics11091451.

[39] L. Tang *et al.*, "Online and offline based load balance algorithm in cloud computing," *Knowl. Based Syst.*, vol. 138, no. 4, pp. 91–104, 2017. doi: 10.1016/j.knosys.2017.09.040.

[40] F. B. Demir, T. Tuncer, and A. F. Kocamaz, "A chaotic optimization method based on logistic-sine map for numerical function optimization," *Neural Comput. Appl.*, vol. 32, no. 17, pp. 14227–14239, 2020. doi: 10.1007/s00521-020-04815-9.

[41] S. Zhao, Y. Wu, S. Tan, J. Wu, Z. Cui and Y. -G. Wang, "QQLMPA: A quasi-opposition learning and q-learning based marine predators algorithm," *Expert. Syst. Appl.*, vol. 213, no. 190, 2023, Art. no. 119246. doi: 10.1016/j.eswa.2022.119246.

[42] Y. Wang, W. Wang, I. Ahmad, and E. Tag-Eldin, "Multi-objective quantum-inspired seagull optimization algorithm," *Electronics*, vol. 11, no. 12, 2022, Art. no. 1834. doi: 10.3390/electronics11121834.

[43] A. Alharbi, W. Alosaimi, H. Alyami, H. T. Rauf, and R. Damaševičius, "Botnet attack detection using local global best bat algorithm for industrial internet of things," *Electronics*, vol. 10, no. 11, 2021, Art. no. 1341. doi: 10.3390/electronics10111341.

[44] X. Cai, X. -Z Gao, and Y. Xue, "Improved bat algorithm with optimal forage strategy and random disturbance strategy," *Int. J. Bioinspired Comput.*, vol. 8, no. 4, pp. 205–214, 2016. doi: 10.1504/IJBIC.2016.078666.

[45] N. Neggaz, A. A. Ewees, M. Abd Elaziz, and M. Mafarja, "Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection," *Expert Syst. Appl.*, vol. 145, no. 1181, 2020, Art. no. 113103. doi: 10.1016/j.eswa.2019.113103.

[46] J. -S. Pan, Z. Fu, C. -C. Hu, P. -W. Tsai, and S. -C. Chu, "Rafflesia optimization algorithm applied in the logistics distribution centers location problem," *J. Internet Technol.*, vol. 23, no. 7, pp. 1541–1555, 2022. doi: 10.53106/160792642022122307009.

[47] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in iaas cloud computing environment," *PLoS One*, vol. 12, no. 5, 2017, Art. no. e0176321. doi: 10.1371/journal.pone.0176321.