Bidirectional Search Factor Based Osprey Optimization Algorithm for Task Scheduling Optimization in Cloud Computing

Yu-Cai Wang, Yu-Feng Sun, Jie-Sheng Wang*, Si-Wen Zhang, Xiao-Fei Sui

Abstract—**Cloud computing task scheduling optimization is a key technology to improve resource utilization and reduce operating costs. The traditional scheduling methods are sometimes limited in the aspects of dynamic, extensibility and multi-objective. To solve this problem, this paper proposes an Osprey Optimization Algorithm based on bidirectional search factor (IOOA) to solve the cloud computing task scheduling optimization problem. First, eight bidirectional search factors are proposed. The bidirectional search factor strategy mainly controls the bidirectional random search through different types of trigonometric functions, which greatly increases the possibility of the exploration phase and avoids the algorithm falling into the local optimal solution. Then, CEC-2022 is used to test the proposed strategy, and it is found that IOOA based on asin trigonometric function bidirectional search factor has the best effect. Compared with other SI algorithms, IOOA can get lower fitness value, which proves the superiority of the proposed method again. Finally, considering the total cost, time cost, load cost and price cost, IOOA is used to optimize the system task scheduling in small-scale and large-scale task scenarios. The results show that the proposed method has the best performance and has excellent performance in cloud computing task scheduling.**

Index Terms—**cloud computing, task scheduling, osprey optimization algorithm, bidirectional search factor**

I. Introduction

With progress of science and technology, the global digitization has been unprecedentedly developed. From the initial infrastructure virtualization technology, Cloud Computing has gradually developed into a powerful core support for the modern digital economy. Cloud Computing is a service model that provides on-demand computing resources through the Internet [1]. This means that users do not have to directly manage the underlying infrastructure, so they can flexibly access and use resources, and pay for what they actually use. The service system of cloud computing is extremely rich, and common service models are divided into various types, such as infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) [2]. IaaS provides virtualized hardware resources (virtual machines, networks, storage, etc.). PaaS provides a development environment. SaaS provides ready-to-use software directly, accessed via a browser. From the perspective of deployment mode, cloud computing can be subdivided into public cloud, private cloud [3], hybrid cloud, edge computing [4] etc. Each model has its own advantages. The public cloud is operated by a third-party professional organization. With the help of resource sharing, the cost of users is effectively reduced. Private cloud is mainly built by enterprises or entrusted to professional institutions hosting, which gives enterprises stronger data control, to ensure data security. A hybrid cloud is a combination of the two, combining the flexibility of a public cloud with the security of a private cloud. Edge computing is the practice of moving calculations closer to data sources (such as IoT devices), significantly reducing data transmission latency and improving system responsiveness.

With its powerful computing resources and flexible service model, cloud computing has become an important infrastructure supporting large-scale computing and data processing [5]. In the cloud computing system, the efficiency of task scheduling plays a decisive role in the rational utilization of system resources, the steady improvement of service quality and the effective control of operating costs. Efficient task scheduling can greatly shorten the waiting time of task execution, give full play to the potential of cloud computing resources, and achieve maximum benefits. For cloud computing technology, how to efficiently schedule tasks and make them reasonably allocated in the computing resource pool has become a key challenge [6]. The cloud computing system is huge in scale and complex in structure, and the traditional scheduling algorithm gradually shows its limitations when dealing with the complex and changeable tasks and resource environment. For example, it is easy to fall into local optimal solutions, high computational complexity and poor adaptability, and it is difficult to meet the growing needs of cloud computing systems. At the same time, because the cloud computing environment has the characteristics of dynamic, heterogeneous and resource competition, the task scheduling problem presents a high degree of complexity,

Manuscript received April 15, 2025; revised July 11, 2025. This work was supported by the Basic Scientific Research Project of Institution of Higher Learning of Liaoning Province (Grant No. LJ222410146054), and Postgraduate Education Reform Project of Liaoning Province (Grant No. LNYJG2022137).

Yu-Cai Wang is a postgraduate student of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (e-mail: wyc@stu.ustl.edu.cn).

Yu-Feng Sun is a postgraduate student of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (e-mail: sunyufeng@stu.ustl.edu.cn).

Jie-Sheng Wang is a professor of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (Corresponding author, phone: 86-0412-2538246; fax: 86-0412-2538244; e-mail: wjs@ustl.edu.cn).

Si-Wen Zhang is a postgraduate student at School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (e-mail: zsw@stu.ustl.edu.cn).

Xiao-Fei Sui is a postgraduate student at School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan 114051, China (E-mail: 232081100026@stu.ustl.edu.cn).

which is a typical NP-hard problem, and brings severe challenges to the research and practice in this field.

To solve the above problems, it is difficult for traditional heuristic algorithms and mathematical optimization methods to obtain the global optimal solution in large-scale and dynamic cloud computing environment, but swarm intelligent optimization algorithm (SI) provides a new idea and method for task scheduling optimization of cloud computing system. SI simulates the intelligent behavior of biological groups in nature, showing significant advantages in dealing with complex optimization problems, and can provide effective solutions for cloud computing task scheduling. Introducing SI algorithm into cloud computing task scheduling can effectively improve the scheduling effect and improve the overall performance of the system. At present, SI algorithm has been widely used in the research of cloud computing task scheduling. For example, Majumdar et al. proposed a new variant (LOLSSA) to solve the task scheduling problem in the cloud computing model. Lens opposition learning was used to improve the diversity of the initial population. It is found that the proposed method can meet the QoS requirements of service providers and users [7]. Using artificial fish swarm algorithm (AFSA), Khan et al. proposed an integration method of self-organizing neural network and AFSA to solve the problem of energy-saving cloud resource management. Finally, it is found that the proposed method can improve the quality of service delivery [8]. To solve the unloading problem of networked vehicle computing tasks, Zhang et al. adopted simulated annealing algorithm to enhance the global search capability of the algorithm, and proposed a new method based on PSO strategy. It is found that the proposed method has the lowest cost and significant advantages in terms of cost [9]. Tabary et al. combined GA and PSO algorithms with SMPIA for task scheduling and resource allocation respectively. It is found that compared with other algorithms, the GA method can improve the total execution time and resource utilization [10]. Wang et al. considered the complementarity between algorithms and adopted PSO for iterative optimization and GA as an evolutionary strategy to solve the task unloading problem in device-edge-cloud collaborative computing system. It is found that the proposed method has higher resource utilization and better acceptance rate [11]. Xiao et al. proposed a multi-strategy improvement method (MITSO) based on tuna population optimization algorithm. Chaotic sequence, Lévy flight and oppositional learning strategies are used to solve the limitations of the original algorithm. Compared with other well-known algorithms, MITSO has better performance [12]. Talha et al. propose a new hybrid approach based on Remora Optimization Algorithm to improve the overall performance of cloud computing service quality. The simplex strategy and quasi-opposition learning strategy are used respectively to improve and enhance the ability of the algorithm. It is found that the proposed method is superior in terms of diversity and effectiveness of solutions [13].

Inspired by the behavior of osprey in nature, Dehghani et al proposed the Osprey Optimization Algorithm (OOA) in 2023 [14]. Up to now, OOA has been applied in many fields, and has achieved good results and advantages. For optimal antenna selection, Mandipudi et al. designed a new approach combining OOA and LOA (COLO). In addition, the feature dependence selection technique is also introduced. Compared with traditional algorithms and hybrid algorithms, COLO can obtain lower fitness values and maximize SE [15]. Ramshankar et al. took the advantages of STBO and OOA and designed a hybrid method (HSTOOA) to optimize the parameters in ATN. Using the proposed method for parameter optimization can improve the performance of the customer review prediction method [16]. For image security, Satre et al. considered and designed logistic mapping with OOA to interfere with input images, providing a new model choice for quantum image cryptography [17]. For smart grid energy management, Rudrapogu et al. adopted OOA for load balancing and energy distribution. By comparing single-objective and multi-objective methods, it is found that the proposed method can improve the accuracy of energy demand prediction [18]. Alotaibi et al proposed a new technology for automated and efficient detection of different kinds of crowd density (OOADL-CDDC). The selection of hyperparameters by OOA finally proves that the accuracy of the proposed method is higher than that of the existing model [19]. Younesi et al. used OOA to design a novel task scheduling and resource allocation method (MoTiCPS). Finally, it is found that compared with other methods, the proposed method optimizes the performance of fog nodes, improves the task success rate, and reduces energy consumption [20]. To solve the unloading problem of mobile edge computing tasks, MidhulaSri et al. designed a new hybrid method (HPFOO) based on the advantages of PAO and OOA. Parameters are optimized by HPFOO in order to reduce transmission and computation delay [21].

Although OOA has been applied in many fields, the algorithm still has some problems such as slow convergence speed and easy to fall into local optimal. Although a variety of algorithms have been used to solve the problem of cloud computing task scheduling optimization, there is no guarantee that all algorithms can balance the load or optimize resource allocation at any moment. Therefore, OOA needs to be improved and enhanced to better solve the problem of cloud computing task scheduling optimization. In summary, the key contributions of this paper are as follows:

(1) Eight kinds of bidirectional search factors are designed through different types of trigonometric functions, and an Osprey optimization algorithm (IOOA) based on bidirectional search factors is proposed. The bidirectional search factor can control the bidirectional random search, increase the diversity of search, accelerate the convergence speed, and avoid the algorithm falling into the local optimal solution.

(2) CEC-2022 was used to verify and test the proposed strategy, and it was found that IOOA based on asin trigonometric function bidirectional search factor had the best effect and accelerated the convergence speed. Compared with other swarm intelligent optimization algorithms, IOOA can obtain lower fitness values.

(3) In small-scale and large-scale task scenarios, IOOA is adopted to optimize cloud computing task scheduling. It is

found that the proposed method has excellent performance in cloud computing task scheduling.

The rest structure of this paper is as follows. Section II introduces the cloud computing task scheduling mathematical model and objective function. Section III describes in detail the implementation of the Osprey optimization algorithm based on bidirectional search factor. Section IV is the simulation experiment and result analysis of CEC-2022. Section V applies IOOA to cloud computing task scheduling optimization. In small-scale and large-scale task scenarios, IOOA is used to optimize task scheduling. Section VI summarizes the full text and discusses the future work.

II. Cloud Computing Task Scheduling Mathematical Model

This section introduces the mathematical model and objective function of cloud computing task scheduling.

A. System Architecture Model

Cloud computing system architecture, as the cornerstone of stable operation of cloud computing services, covers multiple hierarchical structures. All layers cooperate with each other to provide users with convenient and efficient services. Overall, the cloud computing system architecture includes the user layer, the task row layer, the VM management layer, the resource pool, the scheduling and management layer, and the task execution layer. Each layer not only has a unique functional positioning, but also collaborates with each other to build an organic whole. The task scheduling process of cloud computing system realizes the efficient task processing and the optimal utilization of resources through the close cooperation of multiple components.

(a) User Layer. The user layer, as the starting point of the task scheduling process of the cloud computing system, is a direct interface for users to interact with the system. Users submit task requests to the cloud computing system through various terminal devices (personal computers, mobile devices, etc.).

(b) Task List Layer. Tasks submitted and verified by the user layer are stored to the task column surface in an orderly manner. This layer acts as a "transit station" for tasks, queuing them up in the order in which they are submitted and the priority set by the user. The scheduling algorithm can reasonably arrange the execution sequence of tasks according to the task list.

(c) VM Management Layer. The VM management layer centrally manages VM resources in the cloud computing system. After the scheduling and management layer determine the node for executing the task, the VM management layer allocates appropriate VMS from the resource pool to the task based on the task requirements.

(d) Resource Pool. A resource pool is a collection of various physical and virtual resources in a cloud computing system, including servers, storage devices, and network bandwidth. These resources are abstracted management and can be flexibly deployed by the system. The scale and performance of the resource pool directly affect the task processing capability of the cloud computing system.

(e) Scheduling and Management Layer. The scheduling and management layer is the core component of the cloud computing system task scheduling process, which is responsible for coordinating the work of each component. The scheduling and management layer collects task information at the top of the task column, resource pool status, and running status of the VM management layer. Then, based on these information, advanced scheduling algorithm is used to select the optimal execution node and resource allocation scheme for each task. The scheduling and management layer also monitors the task execution progress in real time and adjusts and optimizes the scheduling scheme based on the real-time status of the task and the dynamic changes of resources.

(f) Task Execution Layer. The task execution layer is the end point of the task scheduling process and is responsible for the actual execution of the task. When a task is assigned to a specific virtual machine, the task execution layer starts the corresponding program to execute the task. In the process of task execution, the task execution layer will feedback the task execution status and resource usage to the scheduling and management management in real time. Once a task is completed, the task execution layer returns the execution result to the user and notifies the scheduling and management layer to release occupied resources to provide services for subsequent tasks.

B. Computational Resource Model and Objective Function

The optimization of cloud computing task scheduling is classic and complex. The specific mathematical modeling is as follows. *CN* indicates a collection of compute nodes. $CN = \{N_1, N_2, N_3, N_4, \ldots, N_m\}$, *CT* represents a collection of computational tasks. $CT = \{T_1, T_2, T_3, T_4, \ldots, T_n\}$, which satisfies $CN < CT$. The final scheduling result is as follows:

$$A_{nm} = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_n & \cdots & a_{nm} \end{bmatrix} \tag{1}$$

This article uses three metrics to measure the performance of each resource node, including processing power, load capacity, and resource bandwidth. A point-based system is a common approach when quantifying CPU [22]. With this quantification method, the complex performance characteristics of the CPU are converted into a comparable numerical value for system design, resource allocation, or performance evaluation. When scheduling and management select the execution node for the task, the point-based system quantified CPU processing power can quickly select the virtual machines that meet the task's computing requirements. If a task has high computing performance requirements, the scheduling algorithm preferentially assigns the task to a VM with a high CPU point value based on the CPU point value, ensuring efficient execution of the task. The underlying computing system can be modeled as a processing capacity vector, a resource bandwidth vector, and a load capacity vector. They are marked with the three symbols E_n, C_n and S_n respectively. The corresponding calculation tasks are represented by E_t, C_t, and S_t.

In the optimization of cloud computing task scheduling, this paper considers three objective functions: time cost, load cost and price cost, and F_1, F_2 and F_3 correspond to Eq. (2)-(4).

$$F_1 = \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\frac{E_{t,i}}{E_{n,j}} \tag{2}$$

$$F_2 = \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\frac{S_{t,i}}{S_{n,j}} \tag{3}$$

$$F_3 = \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\frac{E_{t,i}}{E_{n,j}} \times \frac{C_{t,i}}{C_{n,j}} \times P \tag{4}$$

where, P represents the price unit.

By looking at the above formula, the three goals F_1, F_2, and F_3 all pursue the required minimum value. The need to consider multiple conflicting objectives in the optimization process is itself a multi-objective optimization problem. In this paper, we consider converting multi-objective optimization into single objective optimization. First of all, due to the different dimensions of the objective function, it needs to be normalized first. The Min-Max normalization method is used for normalization processing, and the calculation formula after normalization is shown in Eq. (5)-(7). Then, the three objective functions are transformed into single objective problems by linear weighting method, and the traditional single objective optimization technique is used to solve them. In order to ensure fairness, the same weight ratio (1/3) is set for the experiment, as shown in Eq. (8).

$$f_1 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\frac{E_{t,i}/E_{n,j}}{max_{\forall i,j}\{E_{t,i}/E_{n,j}\}} \tag{5}$$

$$f_2 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\frac{S_{t,i}/S_{n,j}}{max_{\forall i,j}\{S_{t,i}/S_{n,j}\}} \tag{6}$$

$$f_3 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\frac{(P\times E_{t,i}\times C_{t,i})/(E_{n,j}\times C_{n,j})}{max_{\forall i,j}\{(P\times E_{t,i}\times C_{t,i})/(E_{n,j}\times C_{n,j})\}} \tag{7}$$

$$f_{final} = min\{(f_1 + f_2 + f_3)/3\} \tag{8}$$

III. Osprey Optimization Algorithm Based on Bidirectional Search Factor (IOOA)

This section introduces in detail the implementation of Osprey Optimization Algorithm (IOOA) based on bidirectional search factor.

A. Basic Osprey Optimization Algorithm (OOA)

Osprey Optimization Algorithm (OOA), proposed in 2023, is an algorithm inspired by the predatory behavior of osprey in nature [14]. OOA simulates the unique hunting strategy and group cooperative behavior of ospreys and can be used as a new method to solve complex optimization problems. The algorithm is divided into two stages, the first stage is to locate and catch fish, and the second stage is to bring the fish to a safe position. These two phases correspond to exploration and exploitation respectively.

(a) Locating and fishing

As efficient hunters, ospreys use their excellent eyesight to pinpoint underwater schools of fish. When they spot a target, they quickly attack and finish the hunt. OOA mimics this natural behavior. Through the establishment of the osprey predation model, the position of the search agent changes, thus enhancing the global exploration ability, helping to find the optimal solution area and avoid falling into the local optimal.

Set FP_i is shown in Eq. (9), i represents the i-th osprey and FP represents the fish set.

$$FP_i = \{X_k | k \in \{1,2,\ldots,N\} \wedge F_k < F_i\} \cup \{X_{best}\} \tag{9}$$

The osprey randomly locates the fish and catches it. The position update is shown in Eq. (10), X_i^{P1} indicates the new position of the osprey. If the new position is better than the previous position, it is replaced with the new position, otherwise, no change is made.

$$x_{i,j}^{P1} = x_{i,j} + \mathrm{r}\cdot(SF_{i,j} - \mathrm{I}\cdot x_{i,j}) \tag{10}$$

$$X_i = \begin{cases} X_i^{P1}, if\ F_i^{P1} < F_i \\ X_i, else \end{cases} \tag{11}$$

where, r is a random number between [0,1], I's value takes 1 or 2, F_i^{P1} represents the fitness value of x_i^{P1}, F_i represents the fitness value of the previous position, and SF represents the fish determined by the osprey.

(b) Bring the fish to a safe position

Ospreys take their prey to a safe place to feed. The position of the search agent was fine-tuned by modeling how the osprey carried its prey to the right location. This mechanism enhances the local exploitation ability of the algorithm, and enables it to converge further around the found high-quality solutions and obtain better results.

In the concrete implementation, OOA first randomly generates a "suitable feeding location" for each individual, as shown in Eq. (12). If the new position is better than the previous position, it is replaced with the new position, otherwise, no change is made. This mimics the behavior of the osprey in choosing a safe place to eat.

$$x_{i,j}^{P2} = x_{i,j} + (lb_j + r\cdot(ub_j - lb_j))/t \tag{12}$$

$$X_i = \begin{cases} X_i^{P2}, if\ F_i^{P2} < F_i \\ X_i, else \end{cases} \tag{13}$$

where, F_i^{P2} represents the fitness value of $x_{i,j}^{P2}$, and F_i represents the fitness value of the previous position.

B. Bidirectional Search Factor Strategy

With its powerful global search ability, SI algorithm has shown important application value in many fields. In the original OOA exploration phase, r is equivalent to the core parameter guiding individual search and step size, and its design is directly related to the overall optimization performance of the algorithm. However, the defects of the original OOA, such as slow convergence speed and insufficient local development ability, make it an important breakthrough to explore a new and efficient search mechanism to enhance OOA.

This paper innovatively proposes a bidirectional search factor mechanism based on various trigonometric functions. The core idea of this mechanism is that search factors are generated by 8 different trigonometric functions, including sine function, cosine function, tangent function, secant function, arc sine function, arc cosine function, arc tangent function and arc cotangent function. The search factor controlled by the above trigonometric function takes its value in the interval [-1, 1], forming a bidirectional search

factor and increasing the diversity of exploration. Specifically, the bidirectional search mechanism enables search individuals to explore in the search space simultaneously along the positive and negative directions through the cooperative work of positive and negative search factors. On the one hand, the positive search factor is used to guide the individual to explore the movement, on the other hand, the negative search factor is used to realize the reverse detection. It helps the algorithm to identify the potential good solution region effectively. The bidirectional search factor mechanism has the following advantages. (1) It avoids the unionization of the search direction and avoids falling into the local optimal. (2) The bidirectional search mechanism significantly expands the search coverage and enables the algorithm to explore the solution space more comprehensively. (3) The algorithm can adjust the search direction and amplitude adaptively to further improve the performance of the algorithm.

The algorithm flow chart is shown in Fig. 1. Fig. 2 shows the trend chart of various bidirectional search factors, and Table I is the abbreviation symbol comparison table, including the formulas and parameters of various bidirectional search factors for easier observation and reproduction. The exploration phase of Eq. (10) is modified. After adding various bidirectional search factors, the exploration phase of IOOA is shown in Eq. (14).

$$x_{i,j}^{P1} = x_{i,j} + B_{SF} \cdot (SF_{i,j} - I \cdot x_{i,j}) \tag{14}$$

where, B_{SF} represents various bidirectional search factors.

IV. Simulation Experiment and Result Analysis of CEC-2022

The CEC-2022 test function set is one of the commonly used standard benchmarks for evaluating the performance of optimization algorithms. The test suite contains 12 functions with different characteristics (F1-F12), covering multiple types such as single-peak, multi-peak, hybrid and composite, and can comprehensively test the performance of the algorithm in terms of exploration and development ability, local optimal avoidance, dimensional scaling, etc. This section will introduce the experimental setup, parameter configuration of each algorithm and result analysis in detail.

In order to verify the performance of the proposed strategy and improved algorithm, the experiments in this section all use CEC-2022 test function set, and the basic parameters are consistent on all test function sets. In the experiment, the dimensions were set to 10, the population size to 30, and the number of iterations to 500. The average value of 30 independent experiments was taken to avoid the chance and randomness of the experiment.

A. Choosing the Best Bidirectional Search Factor

Fig. 3 shows the convergence curve trend diagram of the original OOA algorithm and the addition of 8 bidirectional search factor strategies. Table II is a summary of the experimental data results of various algorithms on the CEC-2022 test function set. As can be seen from Fig. 3, in all other test functions except F4, the bidirectional search factor strategy greatly improves the effect of OOA, and is far superior to the original OOA in terms of convergence speed and fitness value, which fully proves the effectiveness of the bidirectional search factor strategy. In F4, the bidirectional search factor using sin, cos and sec trigonometric functions performs worse than the original OOA in terms of average fitness value. The optimal average fitness value can be obtained from 9 test functions including F1, F3, F4, F6, F7, F8, F9, F11 and F12, and the minimum fitness value can be obtained from 6 test functions including F1, F3, F4, F5, F6 and F8 by using asin bidirectional search factor. The minimum mean is obtained on F10 using sin bidirectional search factor. The optimum fitness value is obtained on F9 by cos bidirectional search factor. The tan bidirectional search factor is used to obtain the minimum average value on F5 and the optimal value on F10. The sec bidirectional search factor is used to obtain the optimal value on F12. The minimum value is obtained on F2 by using acos bidirectional search factor. atan bidirectional search factor is used to obtain the minimum mean value on F2, and the minimum fitness values on F7 and F11. The minimum value is obtained on F10 by using acot bidirectional search factor. The optimal values in Table II have been marked in bold. It can be found that IOOA using asin bidirectional search factor obtains the optimal average fitness value in 9 test functions and the minimum fitness value in 6 test functions. Sorted by Friedman, the first place result is highlighted. The strong performance of asin bidirectional search factor is proved again, which is much better than other bidirectional search factors in terms of convergence speed and fitness value.

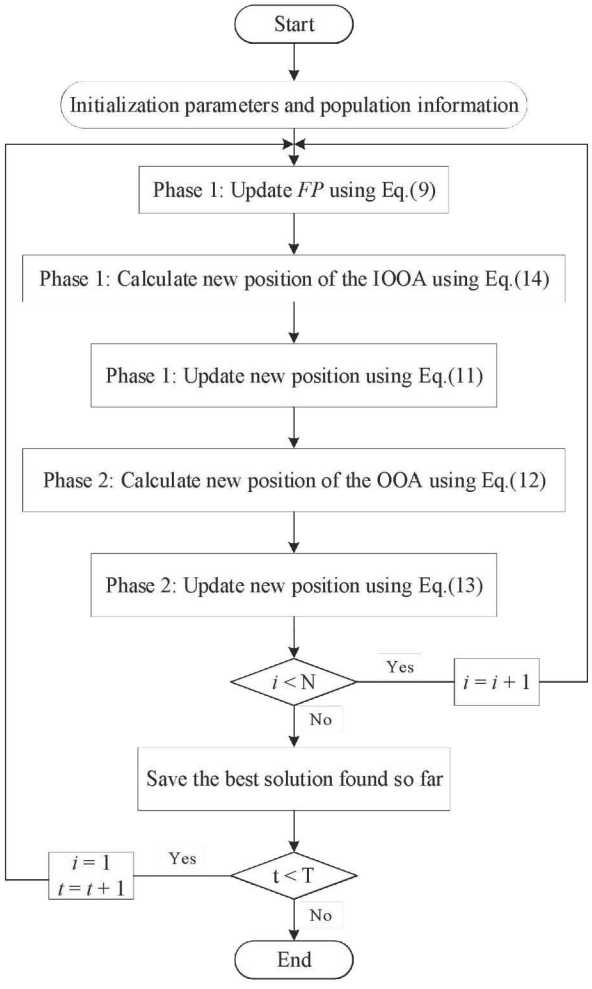
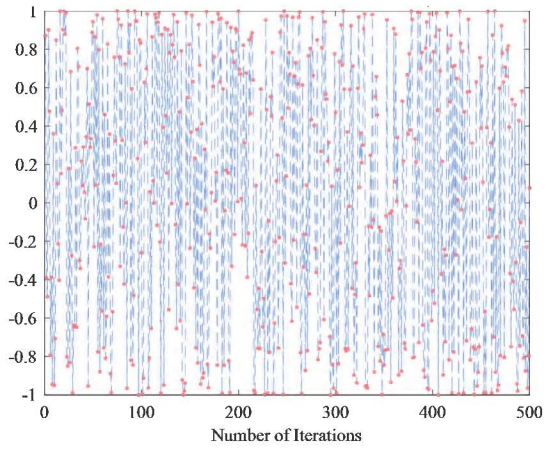
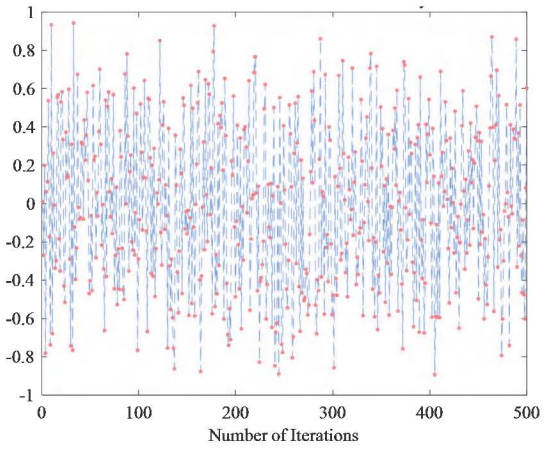


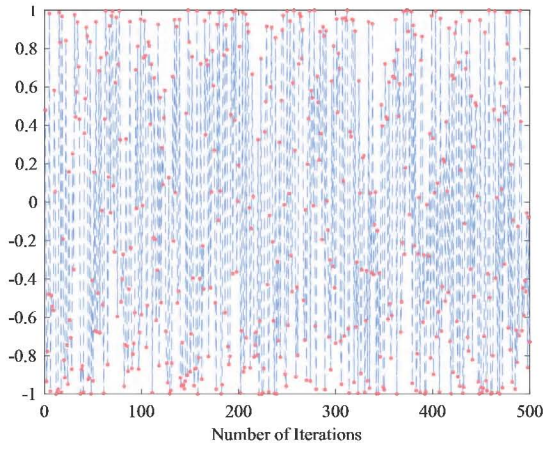
Fig. 1 IOOA flow chart.



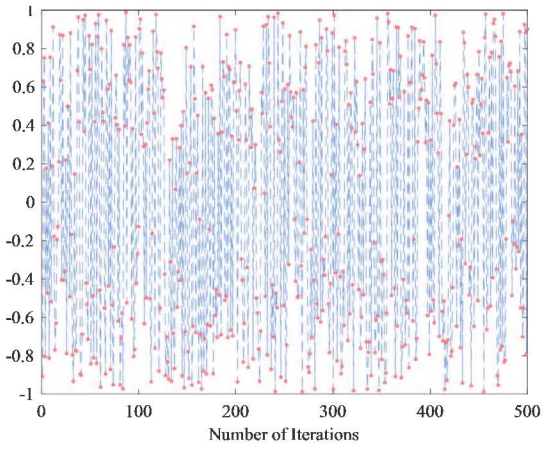
(a) sin



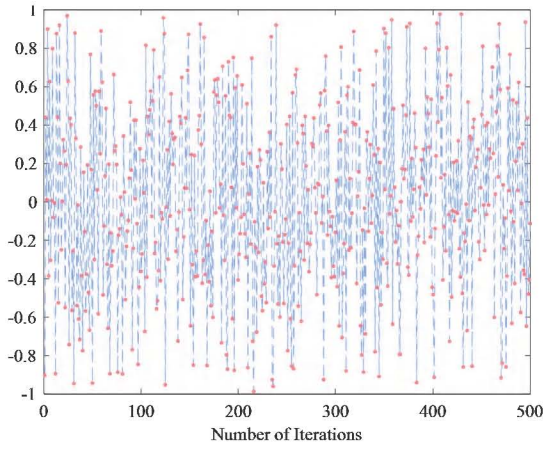
(e) asin



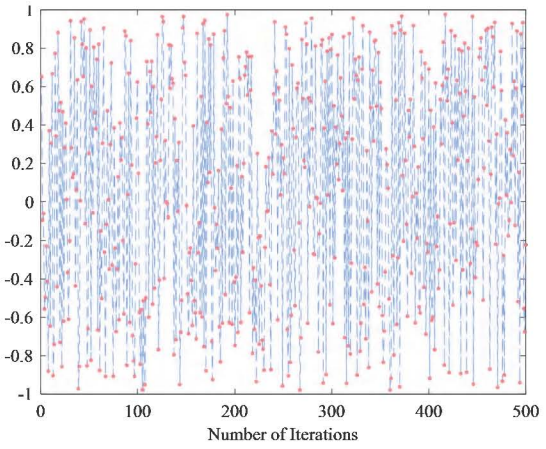
(b) cos



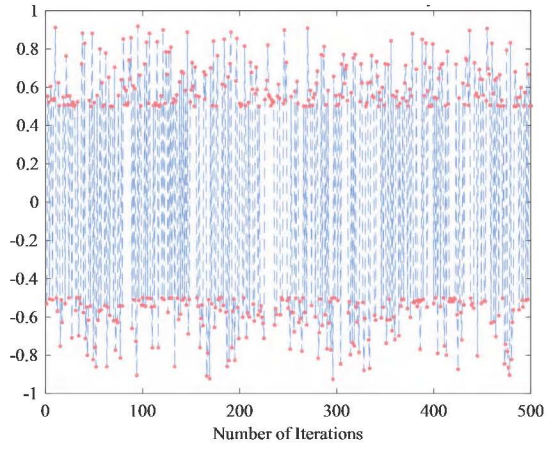
(f) acos



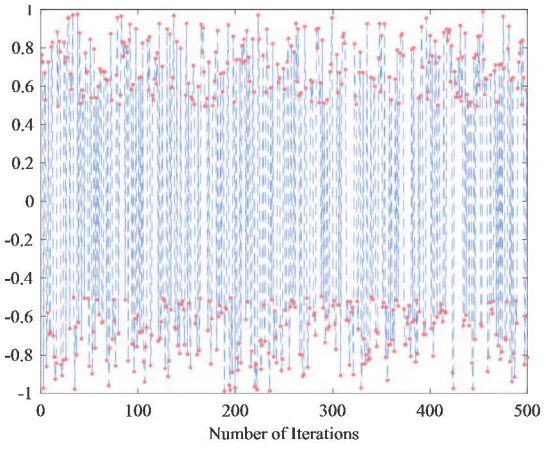
(c) tan



(g) atan

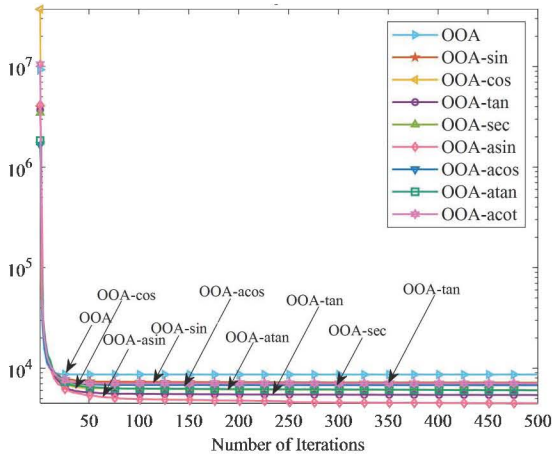


(d) sec

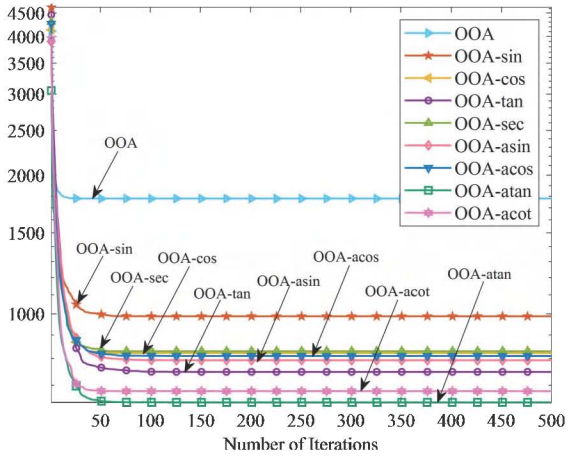


(h) acot

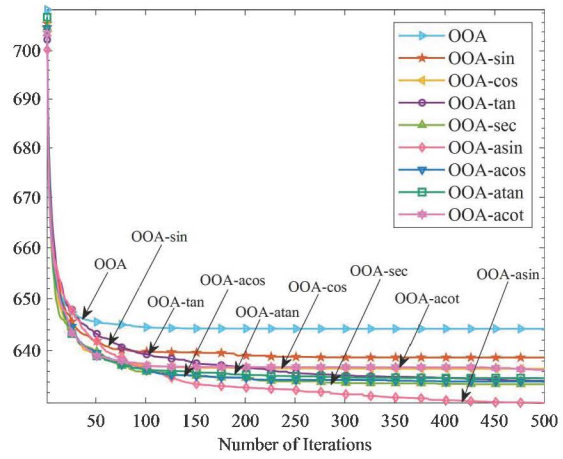
Fig. 2 The changing trend of various bidirectional search factors.



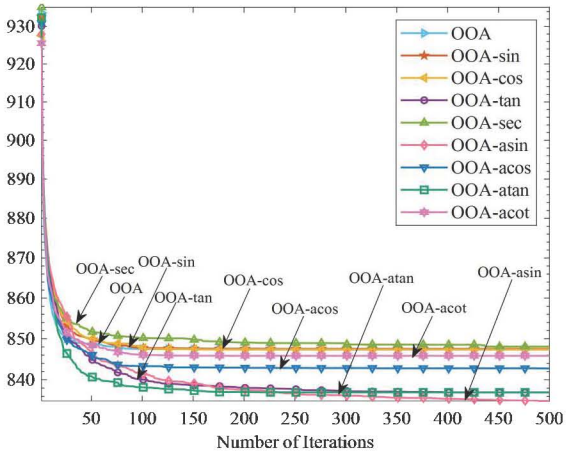
(a) F_1



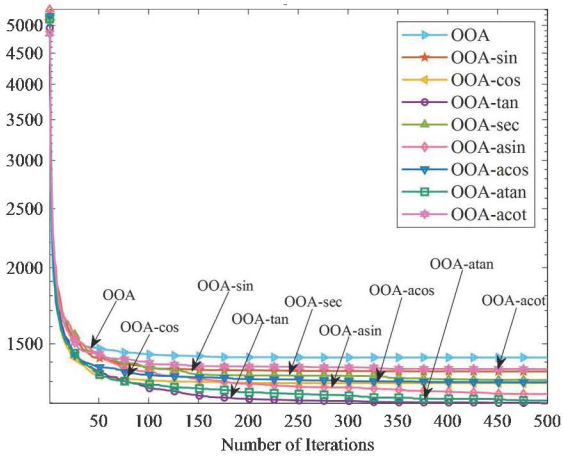
(b) F_2



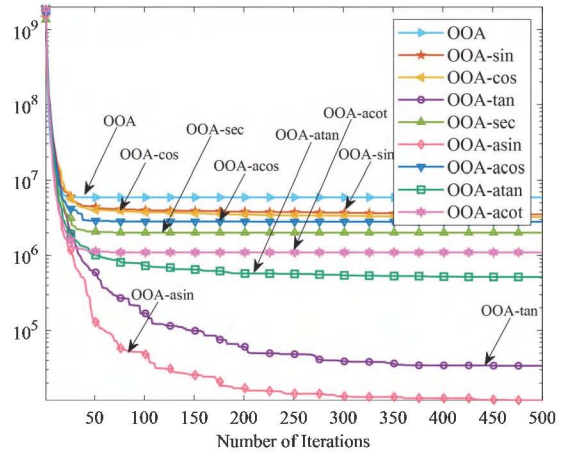
(c) F_3



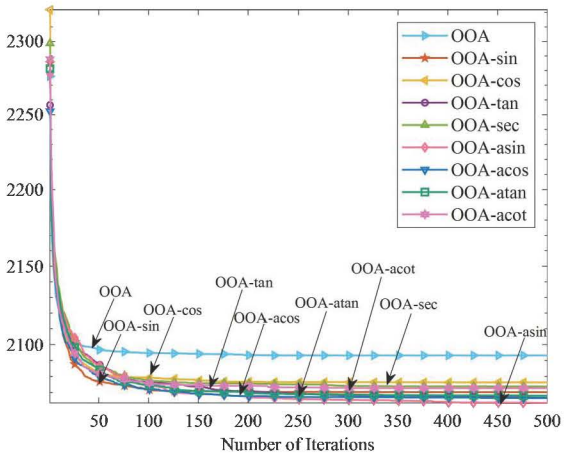
(d) F_4



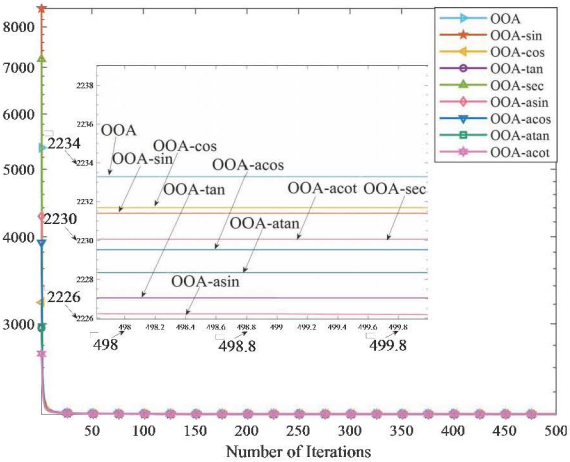
(e) F_5



(f) F_6



(g) F_7



(h) F_8

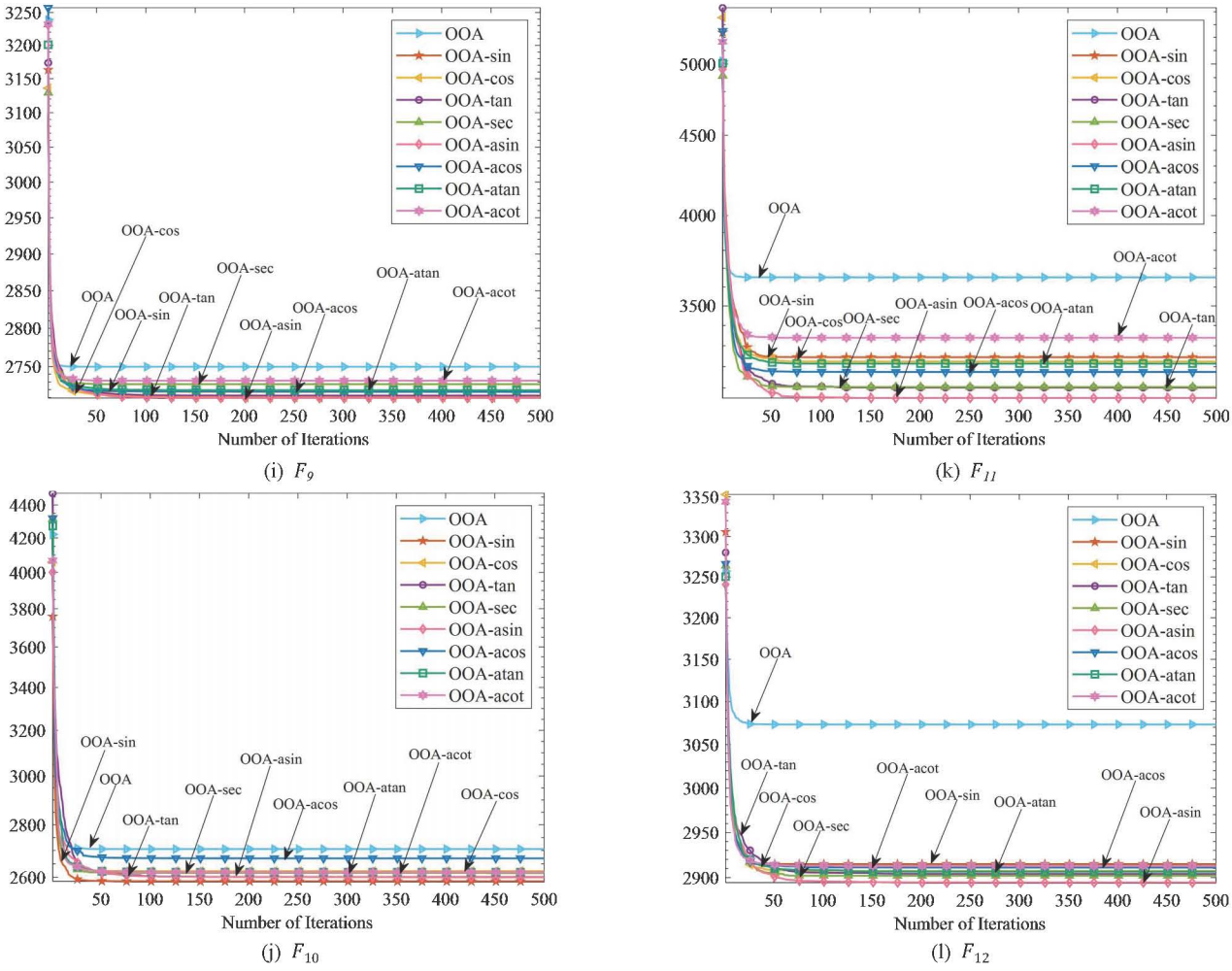


Fig. 3 Convergence curves of various bidirectional search factors on CEC-2022.

TABLE I. DETAILED NAME AND PARAMETER INFORMATION FOR EACH BIDIRECTIONAL SEARCH FACTOR

Full title	Abbreviation	Parameters setting
Sine function bidirectional search factor	sin	$B_{SF1} = sin(2 \cdot pi \cdot \varphi), \varphi \sim U(0,1)$
Cosine function bidirectional search factor	cos	$B_{SF2} = cos(2 \cdot pi \cdot \varphi), \varphi \sim U(0,1)$
Tangent function bidirectional search factor	tan	$B_{SF3} = 0.64 \cdot tan(\varphi) \cdot \sigma, \varphi \sim U(0,1), \sigma = randi([0,1]) * 2 - 1$
Secant function bidirectional search factor	sec	$B_{SF4} = 0.5 \cdot sec(\varphi) \cdot \sigma, \varphi \sim U(0,1), \sigma = randi([0,1]) * 2 - 1$
Arc sine function bidirectional search factor	asin	$B_{SF5} = 0.63 \cdot asin(\varphi) \cdot \sigma, \varphi \sim U(0,1), \sigma = randi([0,1]) * 2 - 1$
Arc cosine function bidirectional search factor	acos	$B_{SF6} = 0.63 \cdot acos(\varphi) \cdot \sigma, \varphi \sim U(0,1), \sigma = randi([0,1]) * 2 - 1$
Arc tangent function bidirectional search factor	atan	$B_{SF7} = 1.25 \cdot atan(\varphi) \cdot \sigma, \varphi \sim U(0,1), \sigma = randi([0,1]) * 2 - 1$
Arc cotangent function bidirectional search factor	acot	$B_{SF8} = 0.63 \cdot acot(\varphi) \cdot \sigma, \varphi \sim U(0,1), \sigma = randi([0,1]) * 2 - 1$

B. Comparison Between IOOA and Other Algorithms

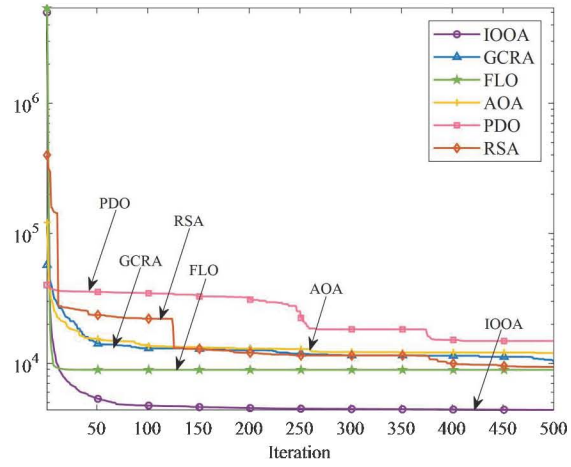
For convenience, the algorithm that adopts the asin bidirectional search factor strategy is named IOOA. This section compares IOOA with other SI optimization algorithms, including GCR A[23], FLO [24], AOA [25], PDO [26] and RSA [27]. The detailed name and parameter information for each algorithm is shown in Table III. Fig. 4 shows the convergence curve trend diagram of IOOA and other 5 SI optimization algorithms. Table IV is a summary of the experimental data results of various algorithms on the CEC-2022 test function set. It can be seen from Fig. 4 that compared with other algorithms, except F10 and F12, IOOA can converge to the optimal value and obtain the minimum average fitness value. The minimum fitness values can be obtained in 9 test functions: F1, F2, F3, F5, F6, F7, F8, F10 and F12. In terms of variance, IOOA can obtain the minimum standard deviation in the six test functions of F2, F4, F6, F7, F8 and F9, indicating that it is more stable. The above analysis fully proves the superiority of IOOA in solving the CEC-2022 test function set. Compared with other algorithms, GCRA obtained the optimal average fitness values in F10 and F12. The minimum fitness value of AOA is obtained on F9. PDO obtains the optimum fitness value on F11. The optimal values in Table IV have been marked in bold, and it can be clearly seen that IOOA obtains the best average fitness value in 10 test functions and the minimum fitness value in 9 test functions. In addition to this, IOOA is highlighted with the first place result after Friedman's ranking.

TABLE II. SUMMARY TABLE OF EXPERIMENTAL DATA RESULTS

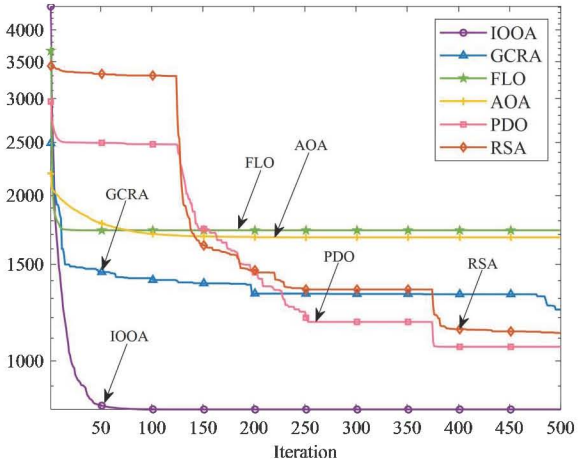
F		OOA	OOA-sin	OOA-cos	OOA-tan	OOA-sec	OOA-asin	OOA-acos	OOA-atan	OOA-acot
	Ave	8.641E+03	7.188E+03	6.069E+03	5.412E+03	6.855E+03	**4.493E+03**	6.779E+03	6.026E+03	7.057E+03
F_1	Std	**1.689E+03**	1.718E+03	2.764E+03	2.091E+03	2.516E+03	1.835E+03	2.285E+03	2.052E+03	2.240E+03
	Best	4.418E+03	3.608E+03	1.967E+03	2.109E+03	2.006E+03	**1.162E+03**	2.446E+03	2.397E+03	3.143E+03
	Ave	1.780E+03	9.883E+02	8.221E+02	7.486E+02	8.299E+02	7.931E+02	8.103E+02	**6.430E+02**	6.797E+02
F_2	Std	7.420E+02	7.265E+02	2.977E+02	2.640E+02	3.157E+02	4.404E+02	3.505E+02	1.581E+02	**1.125E+02**
	Best	9.480E+02	5.590E+02	5.095E+02	4.473E+02	5.055E+02	4.541E+02	**4.275E+02**	4.780E+02	5.035E+02
	Ave	6.442E+02	6.387E+02	6.365E+02	6.343E+02	6.336E+02	**6.301E+02**	6.341E+02	6.347E+02	6.363E+02
F_3	Std	8.423E+00	1.168E+01	1.196E+01	1.091E+01	8.994E+00	9.903E+00	1.025E+01	**7.541E+00**	9.910E+00
	Best	6.282E+02	6.159E+02	6.190E+02	6.112E+02	6.148E+02	**6.111E+02**	6.167E+02	6.231E+02	6.190E+02
	Ave	8.473E+02	8.475E+02	8.474E+02	8.369E+02	8.481E+02	**8.349E+02**	8.428E+02	8.370E+02	8.458E+02
F_4	Std	1.202E+01	8.238E+00	8.722E+00	**6.925E+00**	8.285E+00	8.561E+00	9.197E+00	7.780E+00	1.068E+01
	Best	8.190E+02	8.311E+02	8.225E+02	8.237E+02	8.235E+02	**8.120E+02**	8.251E+02	8.238E+02	8.298E+02
	Ave	1.424E+03	1.351E+03	1.293E+03	**1.199E+03**	1.308E+03	1.240E+03	1.297E+03	1.210E+03	1.363E+03
F_5	Std	2.110E+02	1.914E+02	1.722E+02	**1.325E+02**	2.130E+02	1.807E+02	2.381E+02	1.564E+02	1.678E+02
	Best	1.101E+03	1.045E+03	9.773E+02	1.014E+03	9.722E+02	**9.217E+02**	9.491E+02	9.584E+02	1.089E+03
	Ave	5.921E+06	3.518E+06	3.228E+06	3.390E+04	2.005E+06	**1.175E+04**	2.803E+06	5.137E+05	1.096E+06
F_6	Std	1.304E+07	7.360E+06	1.223E+07	1.180E+05	5.849E+06	**3.596E+04**	1.023E+07	1.899E+06	3.040E+06
	Best	2.510E+03	2.094E+03	1.896E+03	1.896E+03	1.879E+03	**1.860E+03**	1.890E+03	1.861E+03	1.965E+03
	Ave	2.093E+03	2.070E+03	2.076E+03	2.067E+03	2.074E+03	**2.064E+03**	2.067E+03	2.068E+03	2.073E+03
F_7	Std	2.432E+01	2.060E+01	2.156E+01	2.034E+01	2.158E+01	2.082E+01	2.019E+01	**2.001E+01**	2.219E+01
	Best	2.050E+03	2.036E+03	2.032E+03	2.034E+03	2.037E+03	2.024E+03	2.034E+03	**2.022E+03**	2.040E+03
	Ave	2.233E+03	2.231E+03	2.232E+03	2.227E+03	2.230E+03	**2.226E+03**	2.230E+03	2.228E+03	2.230E+03
F_8	Std	9.815E+00	7.977E+00	7.608E+00	3.611E+00	3.971E+00	4.037E+00	3.968E+00	**3.369E+00**	3.543E+00
	Best	2.223E+03	2.224E+03	2.223E+03	2.214E+03	2.221E+03	**2.211E+03**	2.219E+03	2.220E+03	2.225E+03
	Ave	2.750E+03	2.717E+03	2.710E+03	2.712E+03	2.727E+03	**2.709E+03**	2.717E+03	2.719E+03	2.732E+03
F_9	Std	4.543E+01	3.351E+01	3.792E+01	2.692E+01	3.133E+01	**2.411E+01**	4.306E+01	2.994E+01	3.132E+01
	Best	2.634E+03	2.644E+03	**2.619E+03**	2.663E+03	2.679E+03	2.659E+03	2.631E+03	2.642E+03	2.669E+03
	Ave	2.706E+03	**2.585E+03**	2.623E+03	2.601E+03	2.615E+03	2.600E+03	2.670E+03	2.618E+03	2.618E+03
F_{10}	Std	1.572E+02	8.903E+01	9.396E+01	8.070E+01	1.005E+02	**7.991E+01**	1.398E+02	1.113E+02	9.357E+01
	Best	2.516E+03	2.505E+03	2.505E+03	**2.501E+03**	**2.501E+03**	2.504E+03	2.505E+03	2.508E+03	**2.501E+03**
	Ave	3.651E+03	3.244E+03	3.224E+03	3.102E+03	3.105E+03	**3.054E+03**	3.174E+03	3.214E+03	3.339E+03
F_{11}	Std	5.041E+02	3.536E+02	4.035E+02	2.584E+02	2.707E+02	**2.218E+02**	3.645E+02	4.380E+02	3.287E+02
	Best	2.937E+03	2.846E+03	2.779E+03	2.813E+03	2.800E+03	2.775E+03	2.804E+03	**2.770E+03**	2.815E+03
	Ave	3.073E+03	2.915E+03	2.907E+03	2.904E+03	2.902E+03	**2.895E+03**	2.911E+03	2.907E+03	2.913E+03
F_{12}	Std	1.018E+02	3.193E+01	3.989E+01	2.990E+01	2.979E+01	**2.378E+01**	4.020E+01	2.965E+01	4.186E+01
	Best	2.949E+03	2.871E+03	2.870E+03	2.871E+03	**2.865E+03**	2.867E+03	2.869E+03	2.866E+03	2.872E+03
Friedman		8.75	6.67	5.92	2.50	5.25	**1.50**	4.67	3.67	6.08
Rank		9	8	6	2	5	**1**	4	3	7

TABLE III. DETAILED NAME AND PARAMETER INFORMATION FOR EACH ALGORITHM

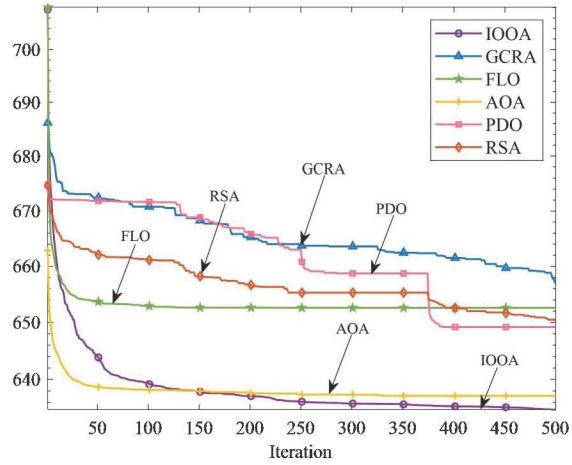
Full title	Abbreviation	Parameters setting
Improved Osprey Optimization Algorithm	IOOA	$I = 1 or 2, cf = 0.63, F = randi([0,1]) * 2 - 1$
Greater Cane Rat Algorithm	GCRA	C is a random number, $\mu = rand[1,4]$
Frilled Lizard Optimization	FLO	r is a normally distributed random number in the interval [0,1], $I = 1 or 2$
Arithmetic Optimization Algorithm	AOA	$c_1 = 2, c_2 = 6, c_3 = 1, c_4 = 2, u = 0.9, l = 0.1$
Prairie Dog Optimization Algorithm	PDO	$\rho = 0.1, \varepsilon = 0.005$
Reptile Search Algorithm	RSA	$\alpha = 0.1, \beta = 0.005$



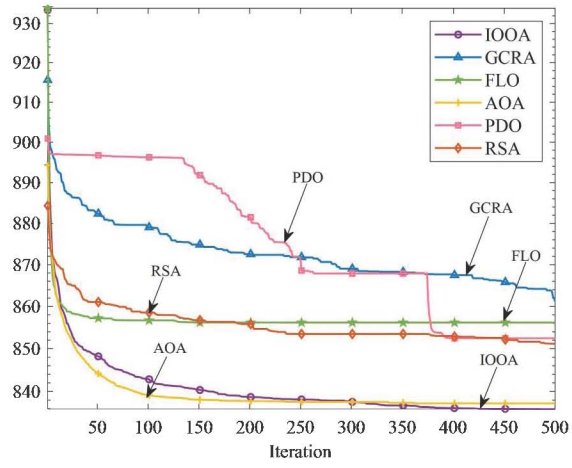
(a) F_1



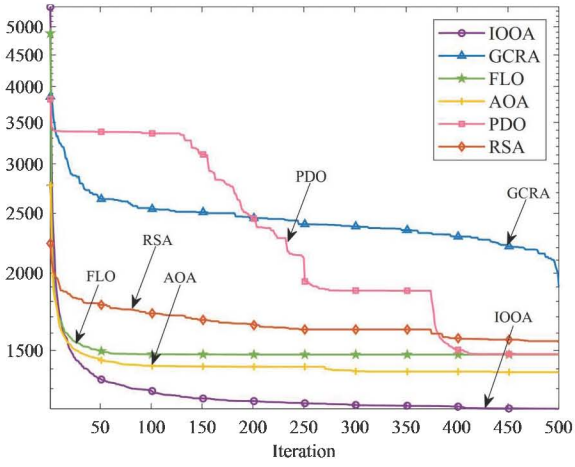
(b) F_2



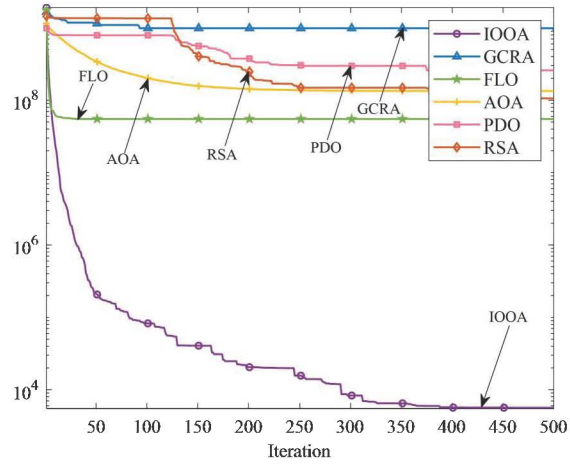
(c) F_3



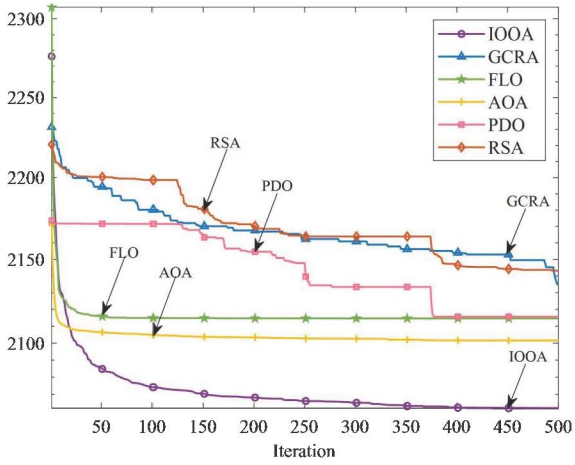
(d) F_4



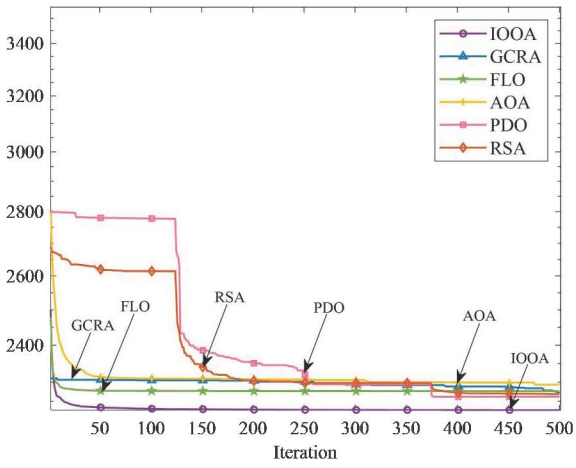
(e) F_5



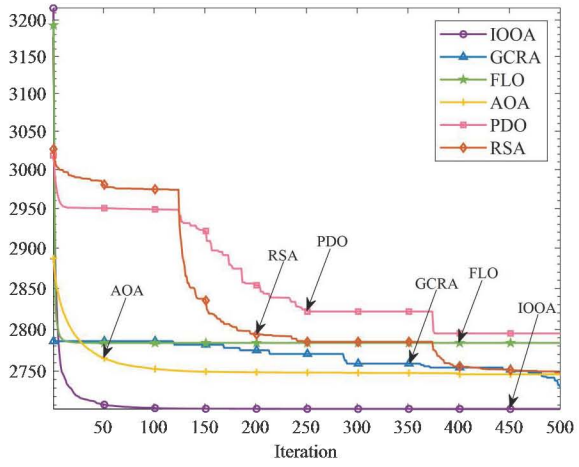
(f) F_6



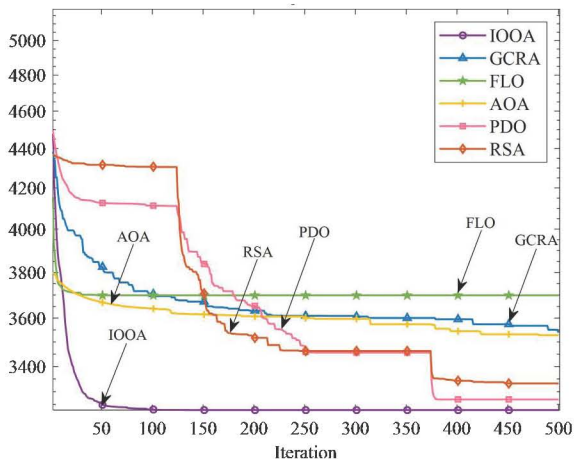
(g) F_7



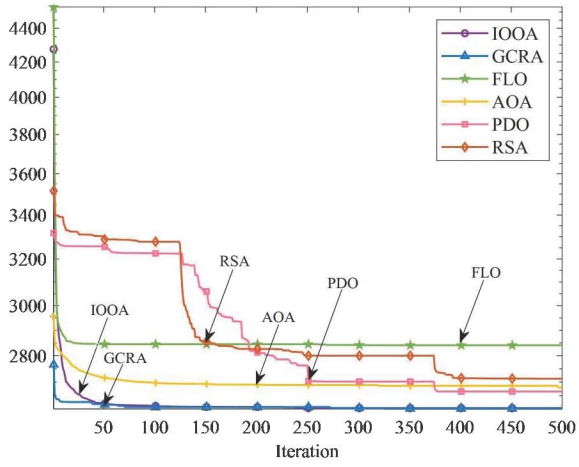
(h) F_8



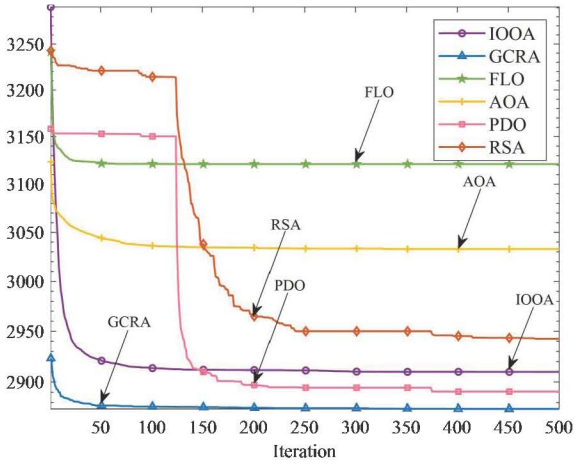
(i) F_9



(k) F_{11}

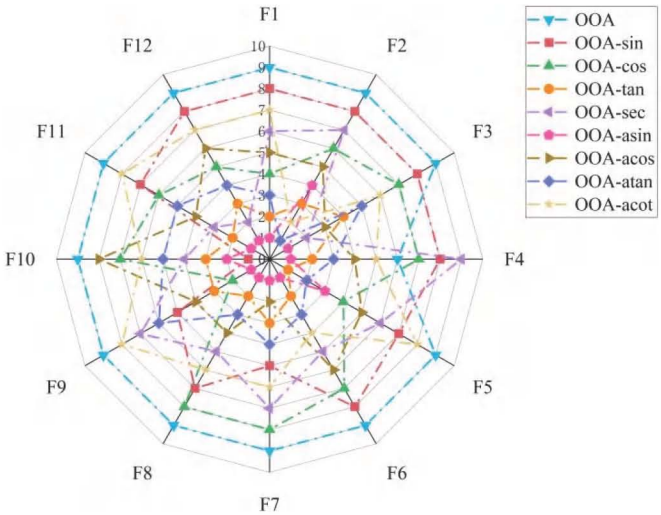


(j) F_{10}

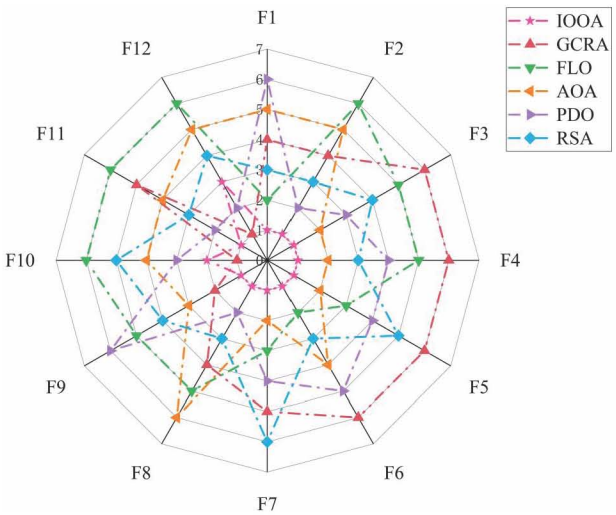


(l) F_{12}

Fig. 4 Convergence curves of various algorithms on CEC-2022.



(a) Ranking radar chart of OOA and its variant algorithms



(b) Ranking radar chart of IOOA and other algorithms

Fig. 5 Ranking radar chart of each algorithm on CEC-2022.

Fig. 5(a) is a radar chart ranking the average fitness values of each variant. It can be found from Fig. 5(a) that the asin has the smallest area, which means that the higher the ranking, the better the performance. The validity of the bidirectional search factor of asin trigonometric functions has been proved again. Fig. 5(b) is the average fitness value ranking radar chart of IOOA and other algorithms. It can be found that IOOA occupies the smallest area and has better performance compared with other algorithms. The validity of IOOA based on the bidirectional search factor of asin trigonometric functions has been proved.

V. IOOA Scheduling Optimization in Small-scale and Large-scale Task Scenarios

This section uses IOOA to solve cloud computing task scheduling optimization problems in small-scale and large-scale. In the simulation experiment process of task scheduling optimization, the maximum number of iterations $T = 500$, the population of each algorithm $P = 50$, the number of virtual machines $VMS = 40$. The following table IV describes the VM and task configuration parameters.

TABLE IV. SUMMARY TABLE OF EXPERIMENTAL DATA RESULTS

F		IOOA	GCRA	FLO	AOA	PDO	RSA
F_1	Ave	**4.391E+03**	1.025E+04	8.919E+03	1.201E+04	1.487E+04	9.396E+03
	Std	1.869E+03	2.347E+03	**1.440E+03**	4.283E+03	7.464E+03	2.294E+03
	Best	**1.905E+03**	6.064E+03	5.199E+03	4.016E+03	5.869E+03	6.368E+03
F_2	Ave	**8.159E+02**	1.240E+03	1.730E+03	1.679E+03	1.061E+03	1.124E+03
	Std	**3.595E+02**	6.063E+02	8.059E+02	6.854E+02	3.758E+02	7.037E+02
	Best	**4.970E+02**	6.376E+02	5.641E+02	6.862E+02	5.623E+02	5.629E+02
F_3	Ave	**6.348E+02**	6.570E+02	6.526E+02	6.371E+02	6.492E+02	6.505E+02
	Std	1.182E+01	**6.714E+00**	1.157E+01	7.214E+00	1.153E+01	9.451E+00
	Best	**6.116E+02**	6.442E+02	6.326E+02	6.228E+02	6.263E+02	6.307E+02
F_4	Ave	**8.359E+02**	8.612E+02	8.562E+02	8.372E+02	8.525E+02	8.511E+02
	Std	**7.923E+00**	8.748E+00	9.098E+00	9.033E+00	1.157E+01	8.491E+00
	Best	8.218E+02	8.388E+02	8.379E+02	**8.209E+02**	8.322E+02	8.378E+02
F_5	Ave	**1.207E+03**	1.901E+03	1.476E+03	1.383E+03	1.480E+03	1.547E+03
	Std	1.726E+02	2.293E+02	1.941E+02	**1.518E+02**	2.321E+02	1.801E+02
	Best	**9.208E+02**	1.419E+03	1.117E+03	1.117E+03	1.185E+03	1.176E+03
F_6	Ave	**5.512E+03**	9.197E+08	5.506E+07	1.341E+08	2.587E+08	1.061E+08
	Std	**1.136E+04**	5.810E+08	7.031E+07	4.075E+08	2.712E+08	1.194E+08
	Best	**1.906E+03**	1.108E+07	3.830E+05	2.052E+03	3.163E+06	1.909E+07
F_7	Ave	**2.062E+03**	2.135E+03	2.115E+03	2.102E+03	2.116E+03	2.143E+03
	Std	**1.664E+01**	3.552E+01	2.830E+01	2.738E+01	2.669E+01	2.664E+01
	Best	**2.031E+03**	2.083E+03	2.044E+03	2.047E+03	2.071E+03	2.096E+03
F_8	Ave	**2.227E+03**	2.272E+03	2.277E+03	2.294E+03	2.262E+03	2.269E+03
	Std	**4.019E+00**	1.971E+01	6.922E+01	7.857E+01	2.804E+01	4.141E+01
	Best	**2.215E+03**	2.241E+03	2.227E+03	2.227E+03	2.228E+03	2.243E+03
F_9	Ave	**2.706E+03**	2.731E+03	2.784E+03	2.746E+03	2.795E+03	2.748E+03
	Std	**2.639E+01**	5.578E+01	4.934E+01	6.608E+01	7.603E+01	5.157E+01
	Best	2.655E+03	2.635E+03	2.702E+03	**2.601E+03**	2.615E+03	2.676E+03
F_{10}	Ave	2.605E+03	**2.602E+03**	2.841E+03	2.680E+03	2.665E+03	2.712E+03
	Std	7.374E+01	**6.029E+01**	2.537E+02	1.736E+02	1.463E+02	1.633E+02
	Best	**2.501E+03**	2.548E+03	2.553E+03	2.523E+03	2.510E+03	2.514E+03
F_{11}	Ave	**3.229E+03**	3.538E+03	3.698E+03	3.528E+03	3.270E+03	3.331E+03
	Std	4.208E+02	**1.550E+02**	4.569E+02	4.281E+02	4.027E+02	3.993E+02
	Best	2.811E+03	3.097E+03	2.952E+03	2.859E+03	**2.792E+03**	2.880E+03
F_{12}	Ave	2.910E+03	**2.874E+03**	3.121E+03	3.033E+03	2.891E+03	2.942E+03
	Std	3.062E+01	**2.812E+00**	1.183E+02	5.826E+01	1.242E+01	6.411E+01
	Best	**2.868E+03**	2.870E+03	2.904E+03	2.935E+03	2.875E+03	2.873E+03
Friedman		**1.25**	4.17	4.50	3.67	3.58	3.83
Rank		**1**	5	6	3	2	4

TABLE V. PARAMETER SETUP OF VMS AND TASKS

Parameter	Range of VM values	Range of Task values
CPU (E)	[200,500]	[10,50]
Memory (S)	[100,500]	[50,100]
Resource (C)	[100,250]	[20,50]

TABLE VI. SUMMARY OF THE DIFFERENT COSTS OF EACH ALGORITHM

100 tasks						
Cost	IOOA	WOA	RSO	AOA	SOA	ACO
Total	**0.2188**	0.2609	0.2967	0.2620	0.2452	0.3012
Price	**0.1330**	0.1531	0.1726	0.1525	0.1454	0.1948
Time	0.3486	0.3400	0.3614	0.3319	**0.3198**	0.3833
Load	**0.1749**	0.2897	0.3310	0.3016	0.2704	0.3255
500 tasks						
Cost	IOOA	WOA	RSO	AOA	SOA	ACO
Total	**0.2224**	0.2886	0.3023	0.2848	0.2720	0.2986
Price	**0.1356**	0.1786	0.1851	0.1764	0.1593	0.1860
Time	0.3656	0.3697	0.3802	0.3694	**0.3509**	0.3735
Load	**0.1658**	0.3177	0.3300	0.3086	0.3058	0.3364

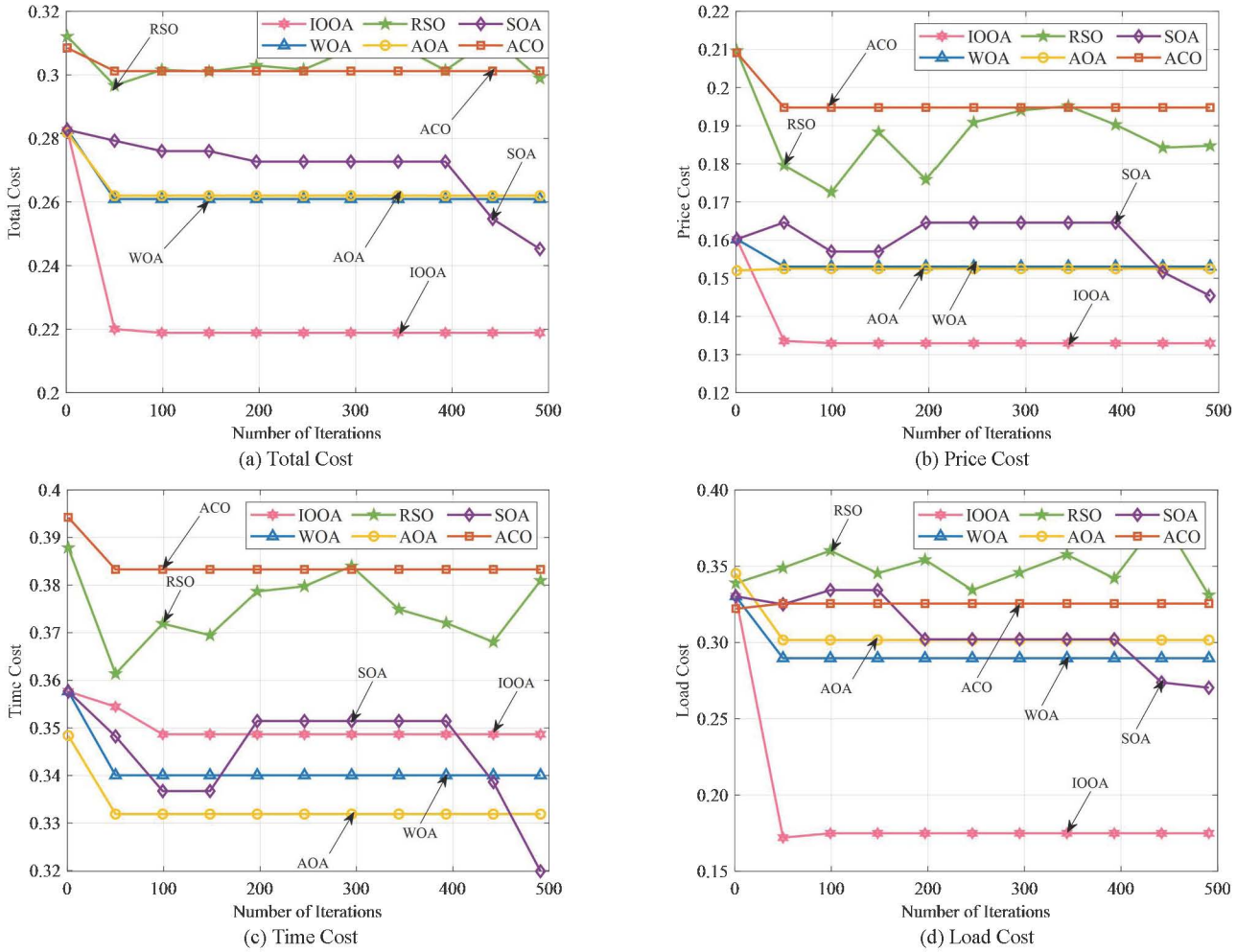
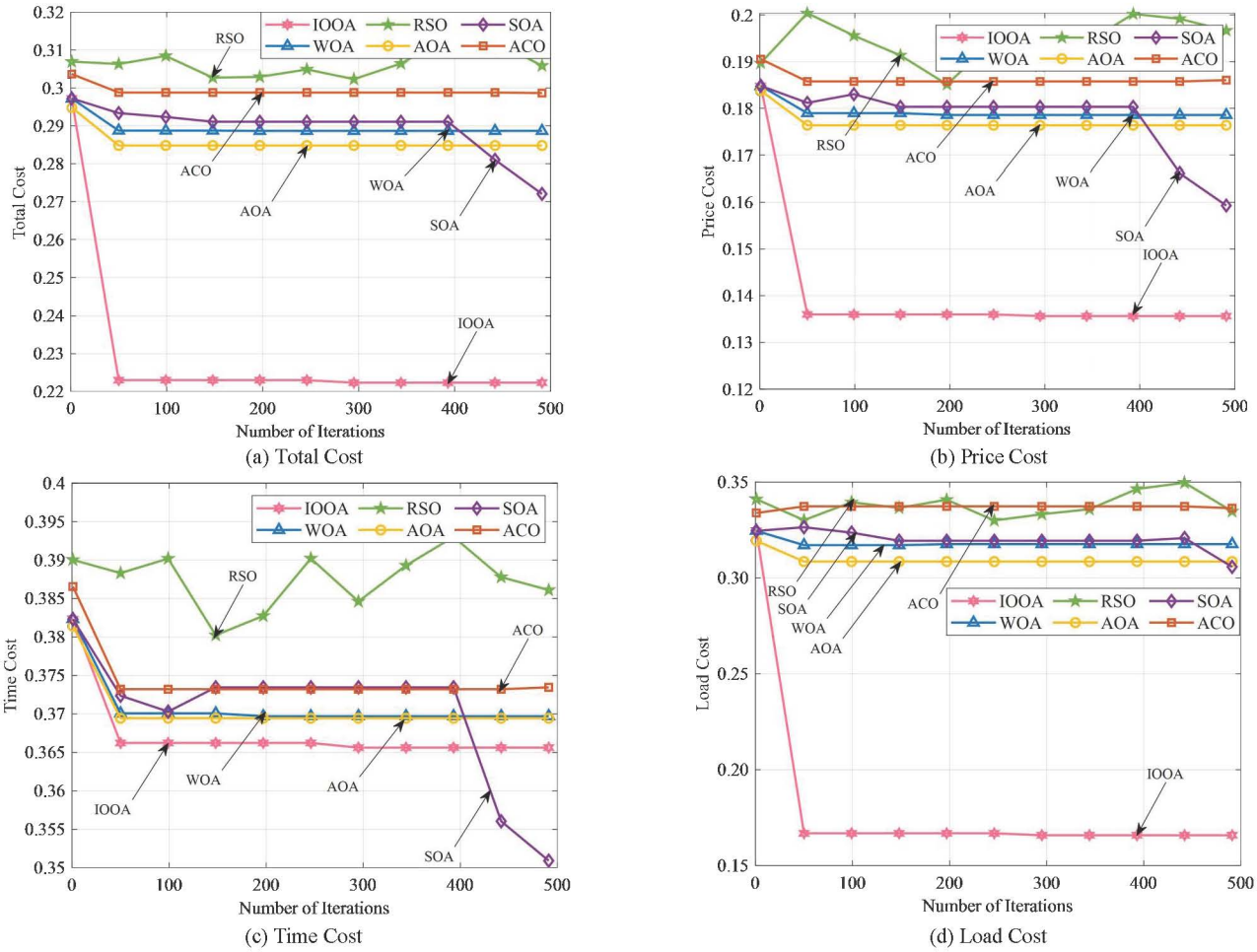


Fig. 6 The change trend of each algorithm when the number of tasks is 100.

A. *Small-scale Task Scenario*

The following are the experimental data and results in a small-scale task scenario. To ensure the fairness of the experiment, in a small-scale task scenario, the performance of various algorithms in terms of total cost, price cost, time cost and load cost is considered when the number of tasks is 100 and 500 respectively. Fig. 6 shows the change trends of different algorithms in total cost, price cost, time cost and load cost respectively when the number of tasks is 100. Fig. 7 shows the variation trend of each cost of different algorithms when the number of tasks is 500. Table VI is a summary of the different costs of IOOA and other SI algorithms when the number of tasks is 100 and 500. For ease of viewing, the optimal values of the data are highlighted in bold.



(a) Total Cost (b) Price Cost

(c) Time Cost (d) Load Cost

Fig. 7 The change trend of each algorithm when the number of tasks is 500.

As can be seen from Fig. 6(a), IOOA has the smallest total cost and can quickly reach the minimum total cost. Compared with other algorithms, IOOA has the best performance and the difference is obvious. In Fig. 6(b), compared with other algorithms, IOOA is still at the bottom of the convergence curve, and the price cost is much lower than other algorithms. As you can see from Fig. 6(c), late in the iteration, SOA suddenly drops off and gains an advantage in terms of time cost. Then, in that order, AOA, WOA, and IOOA. IOOA came in fourth. In Fig. 6(d), the convergence curve of IOOA is much lower than that of other algorithms. This means that IOOA spends the least on load costs. As can be seen from Fig. 7(a), IOOA has far opened up the gap between other algorithms. This illustrates the significant advantages of IOOA in terms of total cost, rapid convergence and minimal total cost spent. In Fig. 7(b), IOOA has the lowest price and cost, and compared with other algorithms, IOOA has the best performance with obvious differences. As you can see in Fig. 7(c), a similar situation occurs as in Fig. 6(c), where SOA suddenly drops off late in the iteration and gains an advantage in terms of time cost. But the difference is that at 500 tasks, IOOA comes in second in terms of cost of time spent. As the number of tasks increases, IOOA's time cost decreases. In Fig. 7(d), the convergence curve of IOOA is still much lower than other algorithms, which once again proves the strength of IOOA in load cost. As can be seen from Table VI, compared with other algorithms, IOOA always spends the least on total cost, price cost and load cost when the number of tasks is 100 or 500. At 100 tasks, IOOA loses 0.0288 more in time cost than SOA. At 500 tasks, IOOA loses 0.0147 more in time cost than SOA. Overall, although the IOOA time cost does not reach the minimum value, the difference is not large. At the same time, considering the performance of IOOA on the other three costs, IOOA is still an effective method for scheduling optimization in small-scale task scenarios.

B. Large-scale Task Scenario

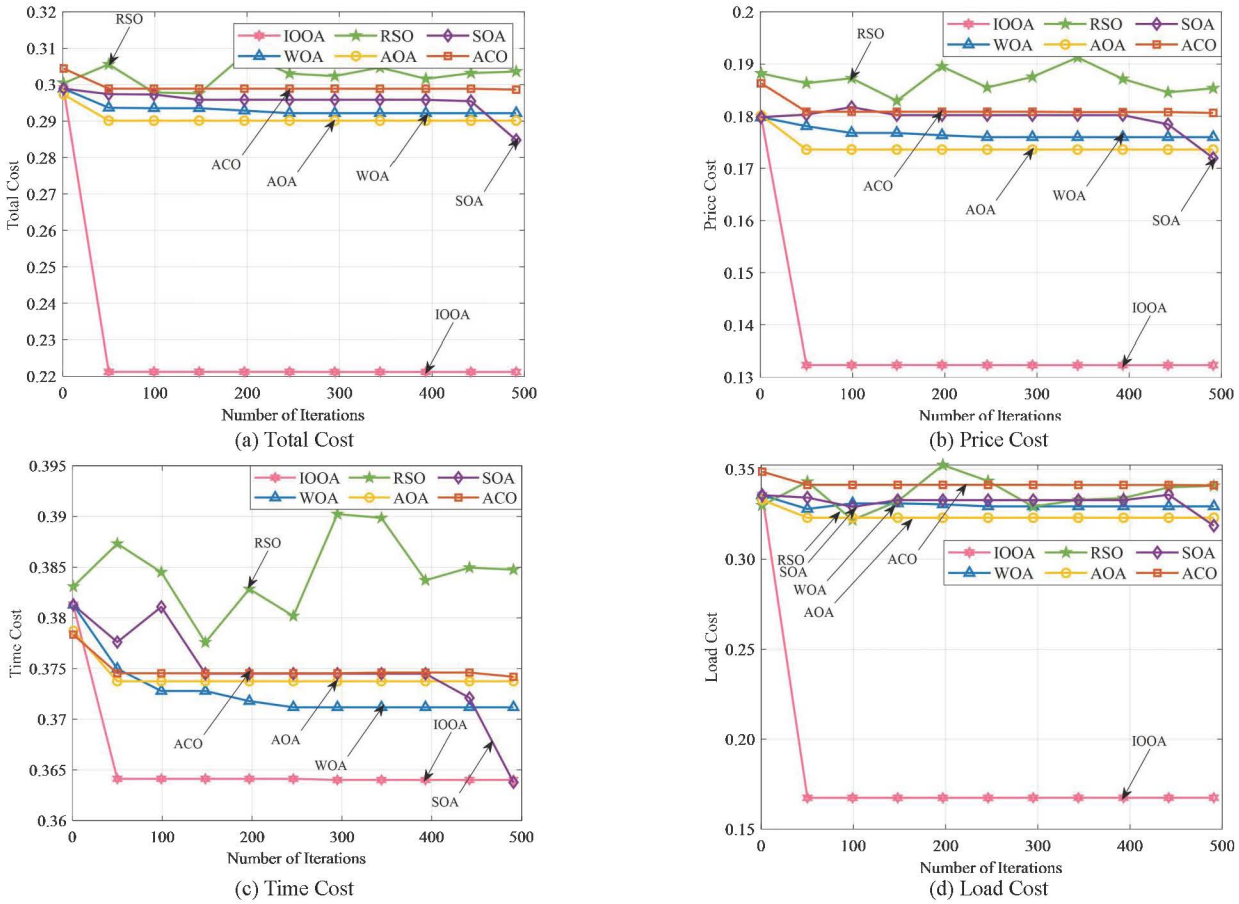
In a large-scale task scenario, the performance of various algorithms in terms of total cost, price cost, time cost and load cost is considered when the number of tasks is 1000 and 5000 respectively. Fig. 8 shows the change trends of different algorithms in total cost, price cost, time cost and load cost respectively when the number of tasks is 1000. Fig. 9 shows the variation trend of each cost of different algorithms when the number of tasks is 5000. Table VII is a summary of the different costs of each algorithm when the number of tasks is 1000 and 5000, and the optimal value is marked in bold. As can be seen from Fig. 8(a), the convergence speed of IOOA is much faster than other algorithms, and the total cost is the least. SOA comes in second, but the difference is huge. In Fig. 8(b), IOOA leads by a wide margin and is far lower than other algorithms in terms of price cost. As can be seen from Fig. 8(c), IOOA quickly reaches the minimum time cost value in the early iteration phase. But at the end of the iteration, the SOA suddenly dips down and is close to the IOOA value. Behind the two is WOA. In Fig. 8(d), IOOA reaches the minimum load cost with very clear speed and trend, and other

algorithms differ greatly from IOOA. It can be seen from Fig. 9(a) that IOOA has a fast convergence speed and its convergence curve is much better than other comparison algorithms. Once again, IOOA performs well in terms of total cost. In Fig. 9(b), IOOA is significantly different from other comparison algorithms and costs the least. As you can see from Fig. 9(c), IOOA spends the least in terms of time cost at 5000 tasks. This is different from 100, 500 and 1000 tasks, as the number of tasks increases, you can see that IOOA shows more and more performance, and less and less loss in time cost. In Fig. 9(d), IOOA is far apart from other comparison algorithms, which fully demonstrates the advantages of IOOA in terms of load cost. Fig. 10 is a bar chart showing the total, price, time and load cost of each SI algorithm under different number of tasks. Through the bar chart, the performance of each algorithm can be viewed and compared more intuitively. As can be seen in Fig. 10, IOOA has outstanding performance in terms of total, price and load cost.

As can be seen from Table VII, compared with other algorithms, IOOA spends the least on total cost, price cost and load cost when the number of tasks is 1000. In terms of time cost, IOOA's time cost is only 0.0002 more than SOA's, which is almost the same. At 5,000 tasks, IOOA is optimal in terms of all costs. Overall, IOOA's performance in total cost, price cost and load cost is very stable, and it is an effective method. As the number of tasks increases, the time cost of IOOA gradually increases. IOOA is a powerful method for scheduling optimization in large-scale task scenarios.

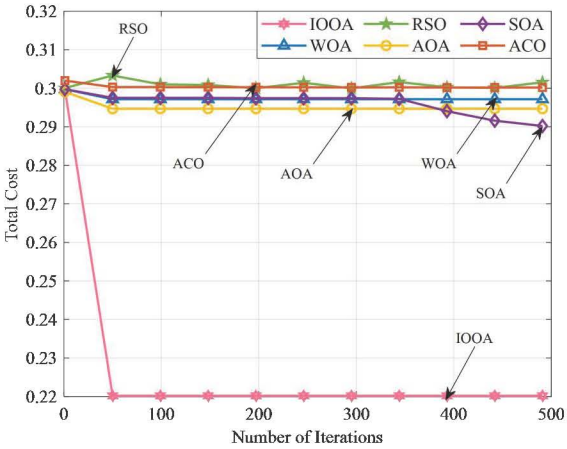
TABLE VII. SUMMARY OF THE DIFFERENT COSTS OF EACH ALGORITHM

1000 tasks						
Cost	IOOA	WOA	RSO	AOA	SOA	ACO
Total	**0.2213**	0.2922	0.2976	0.2876	0.2848	0.2986
Price	**0.1324**	0.1760	0.1829	0.1779	0.1720	0.1806
Time	0.3640	0.3712	0.3776	0.4086	**0.3638**	0.3742
Load	**0.1675**	0.3294	0.3218	0.2764	0.3187	0.3411
5000 tasks						
Cost	IOOA	WOA	RSO	AOA	SOA	ACO
Total	**0.2202**	0.2972	0.3000	0.2870	0.2901	0.3002
Price	**0.1296**	0.1822	0.1810	0.1746	0.1748	0.1816
Time	**0.3691**	0.3794	0.3808	0.4116	0.3751	0.3806
Load	**0.1619**	0.3299	0.3343	0.2749	0.3203	0.3384

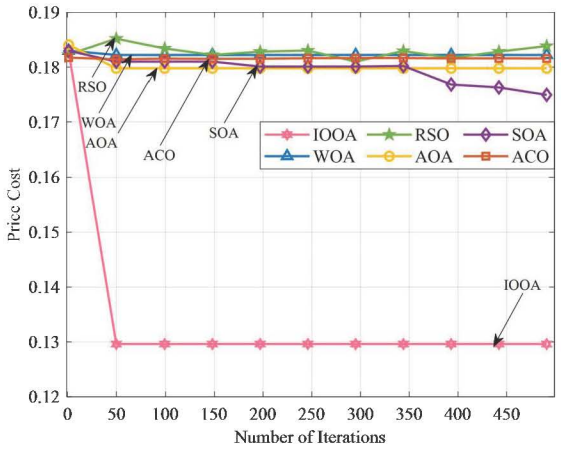


(a) Total Cost (b) Price Cost (c) Time Cost (d) Load Cost

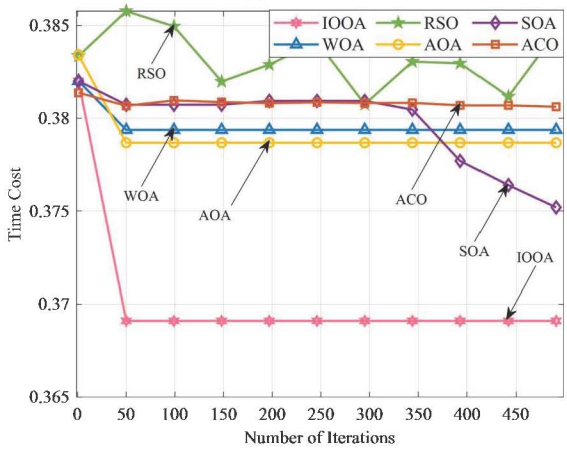
Fig. 8 The change trend of each algorithm when the number of tasks is 1000.



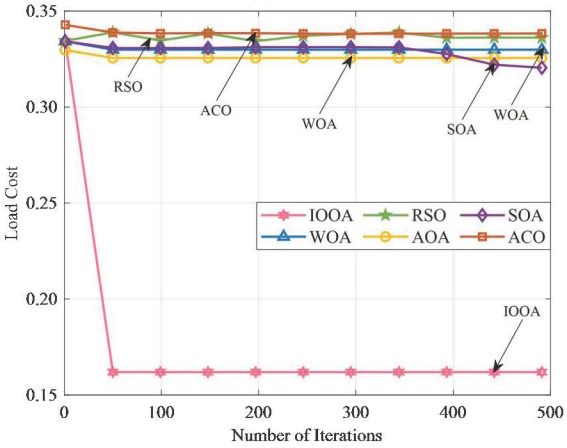
(a) Total Cost



(b) Price Cost

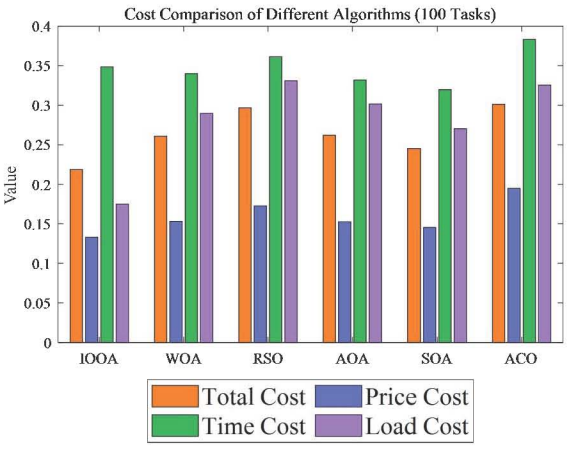


(c) Time Cost

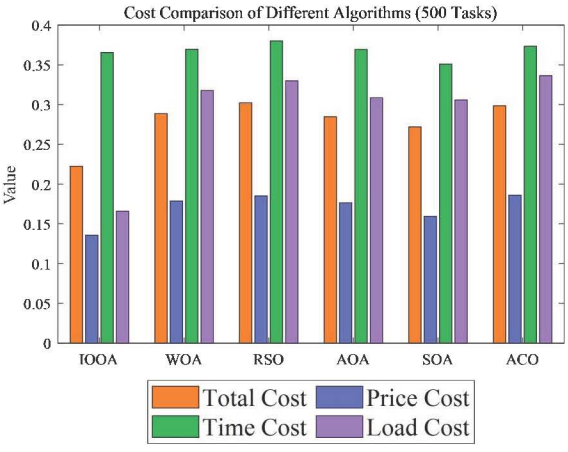


(d) Load Cost

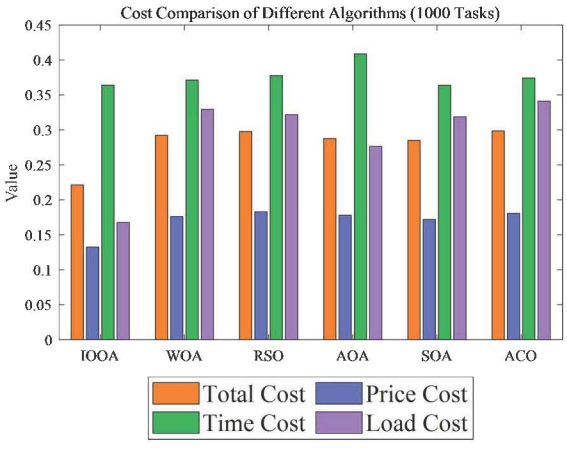
Fig. 9 The change trend of each algorithm when the number of tasks is 5000.



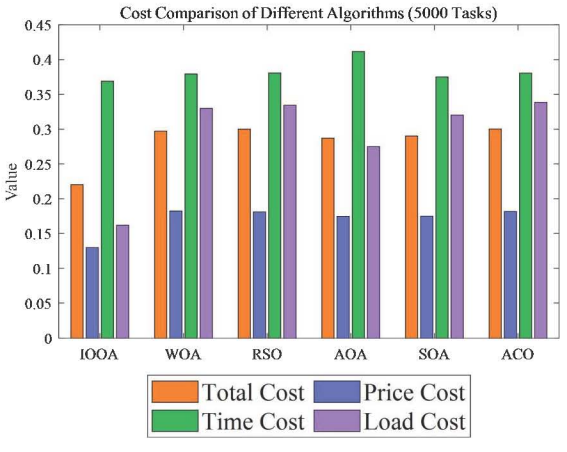
(a) 100 tasks



(b) 500 tasks



(c) 1000 tasks



(d) 5000 tasks

Fig. 10 Bar chart comparison of the costs of each algorithm in different numbers of tasks.

VI. Conclusion

Aiming at cloud computing task scheduling optimization, an Osprey optimization algorithm based on bidirectional search factor is proposed in this paper. The original OOA algorithm has the problem that the convergence speed is slow and the better value cannot be found. Through different types of trigonometric functions, a total of 8 bidirectional search factors are designed. The original OOA algorithm only considers single-direction search at the location update, but each search factor proposed in this paper has its own unique bias and advantages, considering both positive and negative directions. The direction and amplitude of random search are controlled according to the probability advantage of corresponding trigonometric function, which increases the diversity of search, speeds up the convergence speed and avoids the algorithm falling into local optimal. The proposed strategy is tested by CEC-2022, and it is found that IOOA based on asin trigonometric function bidirectional search factor has the best effect. Finally, the proposed algorithm is extended to cloud computing task scheduling optimization problem. Compared with other algorithms in small-scale and large- scale task scenarios, IOOA is proved to be an effective solution to cloud computing task scheduling optimization. In this study, OOA is improved and an enhanced version algorithm (IOOA) is proposed. For future research, it can be expanded from the following directions:

1) For the strategy part, the bidirectional search factor strategy can be considered to replace the chaotic sequence for population initialization, and the influence of population initialization by trigonometric function on the performance of the algorithm can be verified.

2) For cloud computing task scheduling optimization problem, although IOOA achieves excellent performance in cloud computing task scheduling optimization problem, multi-objective optimization can be considered in the future to explore the strength of the algorithm in multi-objective optimization performance.

3) For other applications, IOOA can be considered to solve problems in other fields and explore the universality of IOOA.

References

[1] A. Benlian, W. J. Kettinger, and A. Sunyaev. “The transformative value of cloud computing: A decoupling, platformization, and recombination theoretical framework”, *Journal of Management Information Systems*, vol.35, no. 3, pp. 719-739, 2018.

[2] W. T. Tsai, X. Y. Bai, and Y. Huang. “Software-as-a-service (SaaS): perspectives and challenges”, *Science China Information Sciences*, vol.57, pp. 1-15, 2014.

[3] X. Xu, S. Zang, and M. Bilal. “Intelligent architecture and platforms for private edge cloud systems: A review”, *Future Generation Computer Systems*, vol.160, pp. 457-471, 2024.

[4] A. J. Ferrer, J. M. Marquès, and J. Jorba. “Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing”, *ACM Computing Surveys (CSUR)*, vol.51, no. 6, pp. 1-36, 2019.

[5] C. Yang, Q. Huang, and Z. Li. “Big data and cloud computing: innovation opportunities and challenges”, *International Journal of Digital Earth*, vol.10, no. 1, pp. 13-53, 2017.

[6] W. Khallouli, and J. Huang. “Cluster resource scheduling in cloud computing: literature review and research challenges”, *The Journal of Supercomputing*, vol.78, no. 5, pp. 6898-6943, 2022.

[7] P. Majumdar, and S. Mitra. “Salp swarm algorithm using lens opposition-based learning and local search for constrained optimization problems”, *Iran Journal of Computer Science*, pp. 1-28, 2025.

[8] A. Z. Khan, B. M. Rao, and J. V. N. Ramesh. “Self-organizing neural networks integrated with artificial fish swarm algorithm for energy-efficient cloud resource management”, *International Journal of Advanced Computer Science & Applications*, vol.16, no. 2, 2025.

[9] D. G. Zhang, S. Li, and J. Zhang. “Novel offloading approach of computing task for internet of vehicles based on particle swarm optimization strategy”, *Cluster Computing*, vol.28, no. 3, pp. 156, 2025.

[10] K. A. Tabary, H. Motameni, and B. Barzegar. “QoS aware task scheduling using hybrid genetic algorithm in cloud computing”, *IEEE Access*, vol.13, pp. 51603-51616, 2024.

[11] Y. Wang, P. Zhang, and B. Wang. “A hybrid PSO and GA algorithm with rescheduling for task offloading in device-edge-cloud collaborative computing”, *Cluster Computing*, vol.28, no. 2, pp. 101, 2025.

[12] X. Xiao, G. Tan, and C. Gao. “A multi-strategy improved tuna swarm optimization algorithm for cloud task scheduling”, *2024 5th International Symposium on Computer Engineering and Intelligent Communications (ISCEIC). IEEE,* pp. 174-179, 2024.

[13] A. Talha, and A. Bouayad. “Quasi-opposition remora optimizer based nelder-mead algorithm for tasks scheduling in cloud”, *Cluster Computing*, vol.28, no. 1, pp. 57, 2025.

[14] M. Dehghani, and P. Trojovský. “Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems”, *Frontiers in Mechanical Engineering*, vol.8, pp. 1126450, 2023.

[15] R. Mandipudi, and C. S. Kotikalapudi. “COLO: combined osprey and lyrebird optimization for optimal antenna selection for massive MIMO system”, *Computers and Electrical Engineering*, vol.123, pp. 110245, 2025.

[16] N. Ramshankar, J. A. Shathik, and K. Raju. “Integrated deep learning and blockchain-based framework for cloud manufacturing with improved customer satisfaction”, *Knowledge and Information Systems*, pp. 1-34, 2025.

[17] S. M. Satre, and B. Joshi. “Quantum image cryptography based on discrete chaotic maps”, *Quantum Machine Intelligence*, vol.7, no. 1, pp. 30, 2025.

[18] N. Rudrapogu, N. Hemalatha, and S. B. Warkad. “A novel approach to smart grid energy management using the osprey optimization algorithm”, *2025 4th International Conference on Sentiment Analysis and Deep Learning (ICSADL). IEEE*, pp. 622-627, 2025.

[19] S. R. Alotaibi, H. A. Mengash, and M. Maray. “Integrating explainable artificial intelligence with advanced deep learning model or crowd density estimation in real-world surveillance systems”, *IEEE Access*, vol.13, pp. 20750-20762, 2025.

[20] A. Younesi, E. Oustad, and M. Abolnejadian. “MoTiCPS: energy optimization on multi-objective task scheduling in IoT-integrated cyber-physical systems”, *IEEE Transactions on Sustainable Computing*, pp. 1-15, 2025.

[21] J. MidhulaSri, C. V. Ravikumar. “Offloading computational tasks for MIMO-NOMA in mobile edge computing utilizing a hybrid pufferfish and osprey optimization algorithm”, *Ain Shams Engineering Journal*, vol.15, no. 12, pp. 103136, 2024.

[22] Y. Zhang, B. Liu, and Y. Gong. “Application of machine learning optimization in cloud computing resource scheduling and management”, *Proceedings of the 5th International Conference on Computer Information and Big Data Applications.* pp. 171-175, 2024.

[23] J. O. Agushaka, A. E. Ezugwu, and A. K. Saha. “Greater cane rat algorithm (GCRA): a nature-inspired metaheuristic for optimization problems”, *Heliyon*, vol.10, no. 11, 2024.

[24] I. A. Falahah, O. Al-Baik, and S. Alomari. “Frilled lizard optimization: a novel bio-inspired optimizer for solving engineering applications”, *Computers, Materials & Continua*, vol.79, no. 3, pp. 3631-3678, 2024.

[25] L. Abualigah, A. Diabat, and S. Mirjalili. “The arithmetic optimization algorithm”, *Computer Methods in Applied Mechanics and Engineering*, vol.376, pp. 113609, 2021.

[26] A. E. Ezugwu, J. O. Agushaka, and L. Abualigah. “Prairie dog optimization algorithm”, *Neural Computing and Applications*, vol.34, no. 22, pp. 20017-20065, 2022.

[27] L. Abualigah, M. Abd Elaziz, and P. Sumari. “Reptile search algorithm (RSA): a nature-inspired meta-heuristic optimizer”, *Expert Systems with Applications*, vol.191, pp. 116158, 2022.