International Journal for Housing Science and Its Applications
Publish August 3, 2025. Volume 46, Issue 3 Pages 507-518

https://doi.org/10.70517/ijhsa46336

Research on key technology of elastic scheduling of heterogeneous cloud resources based on quantitative computing methods

Yanxu Jin[1], Chenglin Li[2,*] and Fengbo Kong[3]
[1] Information Center, Yunnan Power Grid Co., Ltd., Wuhan University of Technology, Kunming, Yunnan, 650000, China
[2] Information Center, Yunnan Power Grid Co., Ltd., Zhejiang University of Technology, Kunming, Yunnan, 650000, China
[3] Information Center, Yunnan Power Grid Co., Ltd., Southwest University, Kunming, Yunnan, 650000, China

Corresponding authors: (e-mail: a644910687@163.com).

Abstract Heterogeneous cloud infrastructures in cloud computing resource provisioning can improve the fault tolerance of cloud computing environments, but they also bring about difficulties that bring about cloud computing resource allocation. In order to enhance the efficiency of heterogeneous resource scheduling and allocation, this paper proposes a heterogeneous cloud resource scheduling algorithm based on the Improved Competitive Particle Swarm Algorithm (ICSO), which utilizes the chaos optimization strategy to initialize the particle swarm, and introduces Gaussian variants to update the victory particle positions to improve the diversity of the populations and to enhance the global searching ability. Simulation experiments of heterogeneous cloud resource scheduling are set up to explore the optimization effect of heterogeneous cloud resource scheduling of ICSO algorithm. Comparing the CSO algorithm and the ACO algorithm, the ICSO algorithm in this paper has a lower GD value and obtains a larger HV value in all 15 experimental data, and is able to search for the optimal solution for heterogeneous cloud resource scheduling in about 100 generations.

Index Terms competitive particle swarm algorithm, chaotic optimization strategy, Gaussian variation, heterogeneous cloud resources, resource scheduling

I. Introduction

With the rapid development of cloud computing technology, the evolution of the technical architecture between different cloud computing vendors is not uniform, and it is difficult to achieve the sharing and collaborative scheduling of resources between different types of cloud resources and cloud platforms [1]. 2021 On May 24, 2021, it was jointly issued by the National Development and Reform Commission and other departments as the “Implementation Plan for the Arithmetic Hub of the National Integration of Big Data Center Collaboration and Innovation System,” which mentions that it is necessary to promote the data center Rational layout, supply and demand balance, green intensification and interconnection [2], [3]. Construct a new type of arithmetic network system integrating data centers, cloud computing, and big data, realize cascade scheduling of arithmetic at different levels, realize clusters and regional data centers within urban areas to carry out integrated scheduling, and realize integrated resource scheduling between multiple clouds, between clouds and data centers, and between clouds and networks [4]. Carry out the integrated co-innovation of arithmetic power with algorithms, data, and application resources [5]. In this context, the research on cloud resource elastic scheduling technology to address heterogeneous multi-cloud scenarios is proposed.

Many problems of resource scheduling in cloud environments are solved using heuristic algorithms. Compared to global search to find an optimal solution, heuristic algorithms can give an approximate optimal solution within a limited computational cost [6], [7]. There is a great variety of heuristic scheduling algorithms, the common ones are Min-Min & Max-Min (Min-Min & Max-Min) algorithms, Highest Earliest Fulfillment Time (HEFT) algorithms, and Rounds-Rolling (RR) algorithms [8]-[10]. The main advantage of the Min-Min algorithm is the short response time and the main disadvantage is that it causes an unbalanced load on the resources. In order to overcome the limitations of this algorithm, Chen, H et al. proposed the Max-Min (Max-Min) scheduling algorithm, which assigns tasks with longer processing time to resources with higher computational power at the beginning, which reduces the task processing time and response time and also improves the resource utilization [11]. Raeisi-Varzaneh et al. proposed an advanced Max-Min algorithm based on Cost-Aware for the task scheduling problem in cloud computing systems, which aims to improve the maximum completion time, waiting time, resource utilization, and cost as compared to traditional algorithms [12]. Ahmed, Z et al. A new clustering-based Max-Min scheduling algorithm for

cloud computing environment can effectively reduce the total maximal span of variable-length tasks and improve virtual machine utilization [13]. The above studies proposed various types of improved Max-Min algorithms for scenarios where there are more short tasks than long tasks, the main drawback is that the short tasks remain unexecuted and starvation occurs [14].

The HEFT scheduling algorithm schedules all tasks to the computing node that enables the earliest completion of the task, and the algorithm greatly improves the task completion rate. However, the algorithm is expensive to maintain resources and task information and ignores the communication overload problem during the scheduling process [15]. Many variants of the HEFT algorithm in subsequent studies have also been used to solve scheduling problems in cloud environments, such as the modified HEFT method proposed by Dubey, K et al. with the objective of minimizing the processing time of all the tasks to efficiently schedule the tasks to the running compute nodes but the algorithm does not take into account the resource utilization of the compute nodes, which creates an imbalance in the load [16]. Gupta, S et al. proposed an enhanced version of the HEFT algorithm for task scheduling in cloud computing environments and achieved better results in minimizing the total completion time of workflow tasks running on virtual machines [17]. Hai, T et al. proposed improved versions of three HEFT algorithms (MXCT, MNCT, AVBS) that are improved improvements in task scheduling in cloud environments improve resource scheduling toughness and efficiency compared to the basic HEFT algorithms [18]. Faragardi et al. proposed a novel resource allocation mechanism and workflow scheduling algorithm (GRP-HEFT) to minimize the completion time of a given workflow under the budget constraints of the hourly-based cost model of the LAAS cloud and outperforms other HEFT techniques [19].

The RR algorithm defines a circular queue where each task is executed in turn.The RR algorithm defines a constant amount of time quantum, if the task is not completed within a quantum, it returns to the circular queue to wait for the next round of scheduling [20]. The main advantage of this algorithm is that the tasks are all executed in turn without waiting for the completion of previous tasks, i.e., the tasks can be scheduled fairly. However, if the circular queue is full or the workload is too heavy then it takes a lot of time to complete all the tasks, also it is difficult to choose the right amount of constant time in this algorithm [21]. Samal and Mishra proposed a weighted RR algorithm to solve the load balancing problem and improve the system response time [22]. Ghazy, N et al. proposed an improved round robin algorithm (ARRA) for task scheduling in cloud computing aiming to optimize metrics such as average waiting time, turnaround time and response time [23]. Zohora et al. proposed an enhanced RR algorithm for task scheduling in cloud computing systems, which dynamically adjusts the quantum time based on the number of available processes, and their burst lengths with high performance [24]. Although the above improved RR algorithm further reduces the system response time and improves other quality of service parameters, it still fails to balance the workload of the cloud data center and thus fails to ensure resource utilization.

For the problem of resource elastic scheduling in cloud environment, resource utilization is very important indicator and there are many researches from the point of view of resource load balancing. Zou, C et al. proposed load balancing of resources among nodes based on resource monitoring through VM migration technique, which relies on the accurate monitoring of the resources of each node of the cluster [25]. Lahande et al. studied the load balancing mechanism in cloud computing and proposed a resource scheduling algorithm based on a reinforcement learning model to optimize cloud resource utilization and provide the best quality of service (QoS) [26]. Jangra and Mangla proposed an efficient load balancing framework for resource scheduling in healthcare deployment in cloud based environment which is energy efficient, low latency and maximum completion time and high throughput as compared to existing techniques [27]. PushpaLatha et al. proposed a cost-effective load balancing scheme for cloud computing, which significantly improves the resource utilization and performance of cloud computing by using load balancing techniques and compression techniques for task scheduling [28].

In practical application scenarios, users submit many tasks to cloud computing providers. These tasks can be abstracted into random independent tasks, i.e., the arrival time obeys a random distribution. These tasks have different workload sizes and different resource demands, and when submitting tasks, users often expect the cloud service provisioning platform to complete the tasks quickly, accurately and efficiently. Therefore, how to minimize the total delay of tasks becomes an important research element in elastic scheduling of heterogeneous cloud resources.

Aiming at the heterogeneous cloud resource scheduling problem in large-scale cloud computing environment, this paper firstly constructs a mathematical model and a resource cost model for cloud computing task scheduling, on the basis of which a heterogeneous cloud resource scheduling algorithm based on the Improved Competitive Particle Swarm Optimization (ICSO) algorithm is proposed. Adaptive Gaussian variation is introduced into the CSO algorithm, which is used to update the position of the winning particles, while a chaotic initialization strategy is introduced to initialize the initial population. The adaptivity function containing task completion time, power consumption and load balancing degree is established to improve the convergence speed and optimization

efficiency, and to improve the search accuracy of the algorithm. The optimization performance of the ICSO algorithm in this paper is tested through algorithm simulation, and simulation experiments of heterogeneous cloud resource scheduling are carried out using WorkflowSim simulation platform to verify the optimization effect of the ICSO-based heterogeneous cloud resource scheduling algorithm proposed in this paper.

II. Cloud Computing Task Scheduling Model

Cloud computing is considered to be the most prominent technology in the IT field, which brings great convenience to users by virtue of its on-demand and pay-as-you-go advantages [29]. Resource allocation in cloud computing belongs to the large-scale multi-task scheduling problem, which has always been a hot research topic and is one of the most important and critical problems in cloud computing.

In the field of cloud computing, considering the differences in the demand of different tasks for different resources, it is necessary to briefly simulate the details of the task costs and explain the connection between different resource costs and user budget costs. To address this issue, this paper proposes a resource overhead model.

The resource overhead model is decomposed into 3 parts: CPU, memory, and bandwidth. Based on the resource definition, a multi-objective optimal scheduling model is proposed on the basis of the resource cost model to achieve multi-objective optimal scheduling in cloud computing environment.

II. A.Mathematical modeling

In this paper, it is assumed that there is n task and these tasks should be processed on m computing resources (also known as virtual machine nodes), with the ultimate goal of minimizing the execution time and optimizing the resource utilization of CPU, memory and bandwidth. When the number of tasks is less than the resource nodes, tasks are assigned according to a first-come-first-served algorithm; if the number of tasks is greater than the resource nodes, they are assigned according to a scheduling algorithm. In addition, tasks are not allowed to migrate between resources, i.e., a task can only be assigned to one computing resource node. To formulate the problem, task set $T_i = \{t_1, t_2, t_3, \cdots, t_n\}$ is defined as n independent tasks and $R_i = \{r_1, r_2, r_3, \cdots, r_n\}$ is a resource node. Thus, the relationship between computational resources and tasks is represented as follows:

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mn} \end{pmatrix} \tag{1}$$

where: element P_{ji} represents the correspondence between a computing resource node R_j and a task T_i. P_{ji} is 1 when task T_i is running on compute resource R_j and vice versa, i.e., $P_{ji} \in \{0,1\}$ is 0.

II. B.Resource Cost Model

In cloud computing, tasks and resources are contradictory. The cost of different resources is different, which leads to the difference in task cost. To address this issue, this paper proposes a resource cost model which combines CPU processing power, memory and network bandwidth to give B_d 3 optimization objectives for total task elapsed time $RT_{\max}$, power consumption E and system load degree.

II. B. 1) Total time spent on tasks $RT_{\max}$

The processing time of each task varies depending on the CPU computing power of each computing resource node. The processing time of the i st task on compute resource node j can be expressed as:

$$RT_{ji} = \frac{L_i}{R_{Cj}} \tag{2}$$

where: L_i is the length of task i, and R_{Cj} denotes the CPU computing power of compute resource node j, the completion time for compute resource node j to process all tasks on the current node is:

$$RT_j = \sum_{i=1}^{n} P_{ji} RT_{ji} \tag{3}$$

Since the total system elapsed time $RT_{\max}$ is the maximum value of the time consumed by each computing resource to accomplish their respective tasks, the formula is as follows:

$$RT_{\max} = MAX(RT_j), 1 \le j \le m \tag{4}$$

II. B. 2) Power consumption E

The power consumption required to complete the task is related to the CPU utilization, and the system power consumption is the sum of the power consumption of each computing resource. According to the energy consumption formula $E = P \times T$, the total power consumption E of the system is:

$$E = \sum_{j=1}^{m} R_{OCj} RT_j \tag{5}$$

where: R_{OCj} and RT_j denote the CPU utilization and task completion time of computing resource j, respectively. Where, R_{OCj} The computational formula can be expressed as:

$$R_{OCj} = \frac{\sum_{i=1}^{n} P_{ji} T_{Ci}}{R_{Cj}} \times 100\% \tag{6}$$

II. B. 3) System loadability B_d

The load degree of a computing resource node during a task being executed is mainly determined by the number of tasks assigned to the current node and the computing power of the node. Take R_{Gj}, R_{Mj}, R_{Bj} to denote the availability of CPU, memory and bandwidth on compute node j respectively; T_{Ci}, T_{Mi}, T_{Bi} to denote the number of resources required for task i relative to CPU, memory and bandwidth respectively. Then the utilization of compute node j is:

$$R_{Oj} = \frac{k_1 \sum_{i=1}^{n} P_{ji} T_{Ci} + k_2 \sum_{i=1}^{n} P_{ji} T_{Mi} + k_3 \sum_{i=1}^{n} P_{ji} T_{Bi}}{R_{Gj} + R_{Mj} + R_{Bj}} \times 100\% \tag{7}$$

$$k_1 + k_2 + k_3 = 1 \tag{8}$$

where: k_1, k_2, k_3 denotes the weights of CPU, memory and bandwidth respectively.

Reflecting the load of the system in terms of the standard deviation coefficient of utilization between each computing resource, the system load degree B_d is:

$$R_{Oavg} = \frac{\sum_{j=1}^{m} R_{Oj}}{m} \times 100\% \tag{9}$$

$$B_d = \sqrt{\frac{\sum_{j=1}^{m} (R_{Oj} - R_{Oavg})^2}{m-1}} \tag{10}$$

In equation (9), R_{Oavg} is the average resource utilization of the whole system.

III. Heterogeneous Cloud Resource Scheduling Algorithm Based on ICSO

To improve the security of cloud computing environments, developers often leverage infrastructure heterogeneity to increase the difficulty of detection by attackers and improve the security of cloud computing environments. However, at the same time, heterogeneous cloud infrastructures can bring challenges in cloud computing resource allocation. For this reason, this chapter will propose a heterogeneous cloud resource scheduling algorithm based on the Improved Competitive Particle Swarm Algorithm (ICSO) to improve the efficiency of resource scheduling and allocation.

III. A. Task Scheduling Strategy

III. A. 1) Competitive particle swarm algorithm

Particle Swarm Optimization (PSO) algorithm, as one of the most classical swarm intelligence algorithms, is widely used to solve single objective optimization problems (SOPs) due to its simple implementation and fast convergence

[30]. In each iteration process firstly, the population is randomly divided equally into 2 groups, and the two groups of particles are compared competitively two by two, according to the size of the adaptation value are the winner and the loser, the winner will directly enter the next generation, and the loser learns from the winner and updates its own position and speed according to Eq. (11) and Eq. (12):

$$v_L(t+1) = r_1(t) \times v_L + r_2(x_w(t) - x_L(t)) + \varphi r_3(\overline{x}(t) - x_L(t)) \tag{11}$$

$$x_L(t+1) = x_L(t) + v_L(t+1) \tag{12}$$

where: $x_w(t), x_L(t)$ denotes the position vectors of the winner and the loser, respectively; $v_L(t+1)$ denotes the velocity vector of the loser; t is the number of iterations; $r_1(t), r_2(t), r_3(t) \in [0,1]^p$ is three random vectors obeying a uniform distribution and having the same dimension as the solution vector; φ is a parameter used to control the effect of $\overline{x}(t)$ on the update of the loser's position; $\overline{x}(t)$ has two meanings, one denotes the average position of all particles, which is global in nature, and the other denotes the local average position of particles in the domain, which is local in nature. In this paper $\overline{x}(t)$ the global average position is used.

III. A. 2) Improved Competitive Particle Swarm Algorithm

In order to enhance the diversity of the population and balance the exploration and development of the population, this paper draws on the idea of variation in genetic algorithm and introduces adaptive Gaussian variation in the CSO algorithm, which is used to update the position of the winning particles, and at the same time introduces a chaotic initialization strategy to initialize the initial population [31].

1) Chaotic initialization strategy

In this paper, a chaotic optimization strategy is used to initialize the particle population. In this paper, the method of Logistic mapping is selected to generate the initial chaotic sequence. The expression of this mapping is:

$$x_{n+1} = \mu x_n (1 - x_n) \tag{13}$$

where: x_n is the chaos variable; parameter $\mu \in (0,4]$; n is the order number of the chaos variable, $n = 1,2,3,\cdots,m$. In the mapping image when $3.5699 < \mu \le 4$, the system is in a chaotic state, in this paper μ takes the value of 4.

In this paper, we initialize the particle swarm positions by using chaotic licentious generation of oxen, and the specific steps are as follows:

(1) For the n initial particle positions in the D-dimensional space, a D-dimensional vector is first randomly generated as the first chaotic vector, i.e., $r_1 \in [0,1]^p$;

(2) Perform $n-1$ iterations of each dimension of r_1 using Eq. (13) to generate $n-1$ chaotic vectors $r_2, r_3, \cdots, r_n$;

(3) Mapping the generated n chaotic vectors to the solution search space according to Eq. (14):

$$x_i = x_{\min} + (1 + r_i)\frac{x_{\max} - x_{\min}}{2} \tag{14}$$

where: $x_{\min}, x_{\max}$ is the upper and lower limits of the search space, respectively; x_i is the position information of the i rd chaotic initialized particle.

2) Adaptive Gaussian Variation

Gaussian variation is introduced into the PSO algorithm to improve the diversity of particles in the solution process, and a multi-objective particle swarm optimization algorithm based on Gaussian variation is proposed [32]. In this paper, Gaussian variation is introduced into the CSO algorithm to update the position of the winning particle, and the variation operation is as follows:

$$\begin{cases} x_{t+1} = x_t + c \times \gamma, rand(0,1) \le P_m \\ x_{t+1} = x_t, rand(0,1) > P_m \end{cases} \tag{15}$$

where: c is the variation step; γ is the random variable obeying Gaussian distribution Gauss(0, 1); P_m is the variation probability. The formula for P_m is:

$$P_m = P_{m,\min} + \frac{(P_{m,\max} - P_{m,\min})k}{N} \tag{16}$$

where: $P_{m,\min}$ is the minimum variation rate; $P_{m,\max}$ is the maximum variation rate; k is the current iteration number; N is the maximum iteration number. As seen from the update formula, the variance probability increases linearly with the increase in the number of iterations. Therefore, the update of the winner after the introduction of adaptive Gaussian variation is shown in equation (17):

$$\begin{cases} x_w(t+1) = x_w(t) + \delta \times Gauss(0,1) \times \\ |x_w(t) - \bar{x}(t)|, rand(0,1) \le P_m \\ x_w(t+1) = x_w(t), rand(0,1) > P_m \end{cases} \tag{17}$$

where: δ is the multiplication operator; $\bar{x}(t)$ is the average position of all particles, and the difference between the current position of the victory particle and the average position of the global particles is used as the variation step; rand(0, 1) is a random number obeying the uniform distribution of [0, 1].

III. B. Design of ICSO-based cloud resource scheduling algorithm

III. B. 1) Particle coding and fitness function design

Let the number of tasks m and the number of virtual resource nodes n. In this study, each particle represents a task allocation scheme, assuming that the solution vector represented by the particle $P = \{p_1, p_2, \cdots, p_n\}$, N denotes the dimension of vector P, N_i denotes the value of the j th dimension of vector P, and $N_j = i$ denotes the assignment of task j to virtual machine i.

Since this paper involves a multi-objective optimization process, the fitness function is associated with the total time taken for task completion $T_{\max}$, power consumption for task completion E, and load balancing B_{degree} 3 objectives, it is necessary to transform the multi-objective problem into a single-objective problem first. Firstly, each optimization objective parameter is normalized as shown in equation (18):

$$F = \frac{f(x) - f(x)_{\min}}{f(x)_{\max} - f(x)_{\min}} \tag{18}$$

where: F represents the normalized value; $f(x)$ denotes the size of an optimization objective parameter in the current system; $f(x)_{\max}, f(x)_{\min}$ denotes the maximum and minimum values of this objective parameter, respectively. The multi-objective optimization problem is transformed into a single-objective optimization problem by weighting the fitness value of a single optimization objective. The fitness function can be expressed as:

$$f_i = w_1 F_T + w_2 F_E + w_3 F_B \tag{19}$$

where: f_i denotes the fitness value of the i nd particle; w_1, w_2, w_3 denotes the weights corresponding to the total time $T_{\max}$ spent on task completion, power consumption for task completion E and load balancing B_{degree} 3 objectives, and $w_1 + w_2 + w_3 = 1$ respectively.

III. B. 2) Improved task scheduling for competitive particle swarm algorithms

The ICSO algorithm proposed in this paper is applied to the heterogeneous resource scheduling problem, and the specific implementation steps of the scheduling algorithm are:

Step 1, one-to-one correspondence between the cloud computing task scheduling scheme and the particle positions in the ICSO algorithm, and the best position of the particle is the best task scheduling scheme;

Step 2, initialize the parameters. Given the task set data, virtual machine parameters, the maximum number of iterations of the algorithm Maxcycle, the population size m, and the mutation step;

Step 3, Chaos initialization of the initial position of the particle swarm;

Step 4, calculating the fitness value of each particle;

Step 5, the population is randomly compared in a two-by-two competition and divided into a winner Winner and a loser Loser based on the size of the adaptation value;

Step 6, update the velocity as well as the position of the losing particle and update the position and velocity of the winning particle. Calculate the adaptation value of the updated particle and update the global optimal value and optimal solution;

Step 7, when the maximum number of iterations is reached, the algorithm ends and goes to step 8, otherwise continue with step 5;

Step 8, get the optimal particle position, that is, get the optimal task scheduling scheme.

III. C. ICSO algorithm simulation experiment

In this chapter, the simulation and comparison experiments of the proposed ICSO algorithm with CSO algorithm and ACO algorithm will be conducted on the cloud computing platform to test the optimization performance of the ICSO algorithm in this paper.

III. C. 1) Experimental setup

Five functions such as Sphere function, Rosenborock function, Rastrigin function, Griewank function, and Ackley function are used to differentiate in the process of comparison.

In the experimental run, the average, standard deviation, maximum and minimum values of each algorithm for 30 runs for different functions are given, and the upper limit of the number of iterations is set to 10000 times, if the method convergence meets the requirements, the algorithm execution is finished, and if the convergence does not reach the expected effect, then the algorithm is automatically terminated after 10000 times of execution or after reaching the preset accuracy.

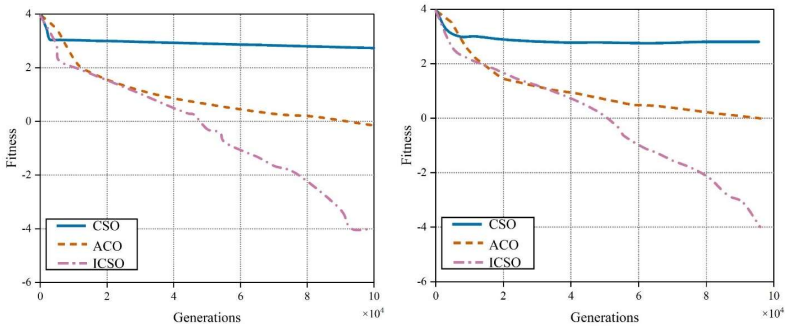
III. C. 2) Experimental results and analysis

The experimental results obtained by the ICSO algorithm proposed in this paper with CSO algorithm and ACO algorithm under the action of five functions are shown in Table 1. From the table, it can be seen that the ICSO algorithm has a significantly higher convergence speed, and compares with the CSO algorithm and ACO algorithm in terms of high convergence accuracy and more stable performance.

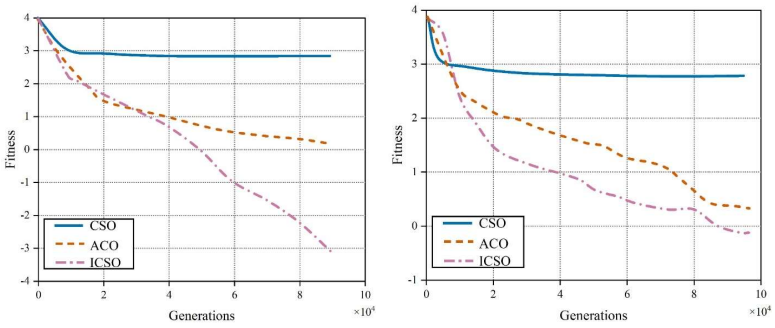
Table 1: The performance of algorithm

Fucntions for category	-	ICSO	CSO	ACO
Sphere	Average	2.72E-04	4.16E+04	1.59E+05
	Best	4.18E-05	7.23E+02	5.33E+05
	Worst	3.32E-02	2.72E+03	2.62E+04
	Std	6.23E-05	5.33E+01	3.83E+02
Rosenborock	Average	5.23E-05	2.34E-03	2.01E+01
	Best	1.34E-07	6.38E+00	7.43E+00
	Worst	3.22E+00	6.20E+01	6.32E+01
	Std	6.91E-02	1.43E-01	1.23E+01
Rastrigin	Average	2.24E-07	1.46E+03	5.23E+02
	Best	3.42E-07	8.35E+03	8.13E+02
	Worst	2.34E+02	1.02E+04	1.23E+03
	Std	5.43E-05	4.68E+02	3.42E+02
Griewank	Average	7.34E-05	4.33E+00	5.54E+01
	Best	1.09E-06	3.35E+00	1.33E+01
	Worst	1.65E-05	4.74E+01	6.78E+02
	Std	3.33E+00	4.23E+01	7.23E-01
Ackley	Average	2.79E-02	6.34E+01	5.78E+02
	Best	3.46E-04	7.00E+20	9.88E+00
	Worst	7.33E-01	8.34E+03	8.76E+02
	Std	3.56E-02	1.10E+05	7.57E+02

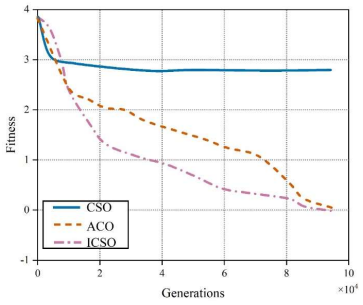
The convergence of the ICSO algorithm proposed in this paper with the CSO algorithm and the ACO algorithm with respect to the five functions running on the dataset is specifically shown in Fig. 1. Figures (a)~(e) correspond to the functions Sphere function, Rosenborock function, Rastrigin function, Griewank function, and Ackley function in that order. From the overall situation to analyze, all algorithms can achieve convergence within 10,000 iterations, but in comparison, the ICSO algorithm proposed in this paper reflects better adaptability and performance on various functions.ICSO algorithm shows strong convergence effect, on average, less than 1,000 iterations begin to converge, and it can ensure excellent convergence state.



(a)Function Sphere (b)Function Rosenborock



(c)Function Rastrigin (d)Function Griewank



(e)Function Ackley

Figure 1: The covergence on 5 Functions

IV. Heterogeneous Cloud Resource Scheduling Simulation Experiments

In order to verify the optimization effect of the heterogeneous cloud resource scheduling algorithm based on ICSO proposed in this paper for heterogeneous cloud resource scheduling, the WorkflowSim simulation platform is used as the experimental environment to encode the implementation of the ICSO algorithm proposed in this chapter, to extend and invoke the simulation program interfaces to implement the proposed heterogeneous cloud resource scheduling process, and to conduct the comparative experiments with the CSO algorithm and the ACO algorithm.

IV. A. Comparative analysis of generation distances

Generation distance (GD) is a performance metric used to evaluate the proximity between the approximate solution sought in the algorithm and the real Petro front.The smaller the value of GD, the better the convergence of the algorithm. The GD values of ICSO algorithm, CSO algorithm and ACO algorithm are shown in Table 2. As can be seen from the table, the ICSO proposed in this paper has lower GD values on all 15 experimental data. The real Petro frontier set in the experiments is obtained by integrating the Petro-optimal solutions selected by the three algorithms in many experiments and ranking the top 100 approximations in a non-dominated order. The algorithms are all iterated 300 times, and it is obvious that CSO algorithm and ACO algorithm are more difficult for the convergence performance under the same conditions, inferior to the ICSO algorithm proposed in this paper. It can

be concluded that ICSO algorithm shows better convergence performance than the comparison algorithm in the above test data.

Table 2: Comparison of GD

Experimental data	ICSO	ACO	CSO
Cybershake50	0.020019	0.019358	0.006457
Cybershake100	0.024635	0.015028	0.003811
Cybershake1000	0.016789	0.019677	0.002122
Epigenomics46	0.019498	0.011951	0.003659
Epigenomics100	0.017209	0.012776	0.002403
Epigenomics997	0.01813	0.013853	0.002055
Inspiral50	0.021513	0.016945	0.009502
Inspiral100	0.026483	0.017934	0.006343
Inspiral1000	0.023822	0.01587	0.007471
Montage50	0.024685	0.012105	0.006306
Montage100	0.02022	0.013058	0.006481
Montage1000	0.024729	0.015372	0.00452
Sipht60	0.015468	0.013358	0.009413
Sipht100	0.025547	0.01588	0.007935
Sipht1000	0.017657	0.012561	0.006054

IV. B. Comparative analysis of super volume

Hypervolume (HV) is used to measure the quality of the nondominated solutions and also to assess the convergence and diversity of the solution set. The test results of the ICSO algorithm proposed in this paper with CSO algorithm and ACO algorithm on 15 workflow data are specifically shown in Table 3. In the test results, the ICSO algorithm dominates in the majority of the test results and obtains a larger HV value. It can be concluded that the ICSO algorithm proposed in this paper shows better diversity and convergence on the solution set than the comparison algorithms.

Table 3: Comparison of HV

Experimental data	ICSO	ACO	CSO
Cybershake50	0.434	0.613	0.6701
Cybershake100	0.4563	0.5045	0.5292
Cybershake1000	0.1048	0.2472	0.2269
Epigenomics46	0.5448	0.7174	0.7432
Epigenomics100	0.5013	0.6316	0.684
Epigenomics997	0.1847	0.361	0.3628
Inspiral50	0.3613	0.5196	0.5542
Inspiral100	0.2557	0.3187	0.3703
Inspiral1000	0.1088	0.1644	0.1877
Montage50	0.4829	0.6494	0.6766
Montage100	0.2156	0.3337	0.3618
Montage1000	0.0547	0.1677	0.1884
Sipht60	0.3372	0.3514	0.3965
Sipht100	0.3437	0.5061	0.535
Sipht1000	0.1231	0.1073	0.162

IV. C. Comparative analysis of optimal solution change curves

The optimal solution change curves of the ICSO algorithm proposed in this paper are compared with those of the CSO algorithm and the ACO algorithm, as shown in Fig. 2. As can be seen from the figure, compared with CSO algorithm and ACO algorithm, ICSO algorithm has an obvious advantage in convergence performance. In about 100 generations, ICSO algorithm finds the optimal solution for heterogeneous cloud resource scheduling, which is much faster than 250 and 300 generations of CSO and ACO algorithms. Comparison results show that ICSO

algorithm prevents the local optimal solution from appearing and obtains a better heterogeneous cloud resource scheduling scheme.

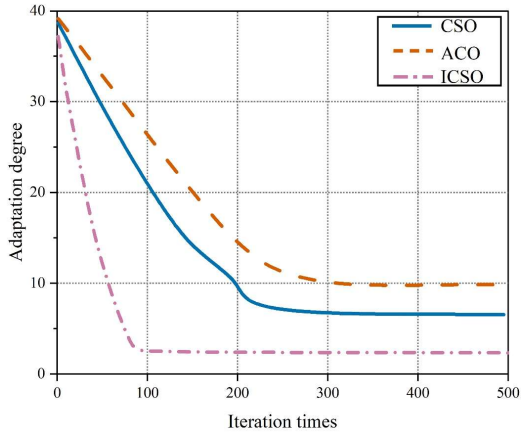


Figure 2: Convergence performances

IV. D. Comparative analysis of task completion times

The 200 heterogeneous cloud resource scheduling tasks are randomly assigned to 10 resource nodes, and the change curves of the task completion time of the ICSO algorithm proposed in this paper and the CSO algorithm and the ACO algorithm are specifically shown in Fig. 3. From the figure, it can be seen that with the increase of the number of nodes, the competition for resources between the tasks for resources weakened, the task completion time of CSO and ACO algorithms decreased, and the ICSO algorithm completes the task in relatively less time, and the comparison results show that the ICSO algorithm has a certain degree of superiority.

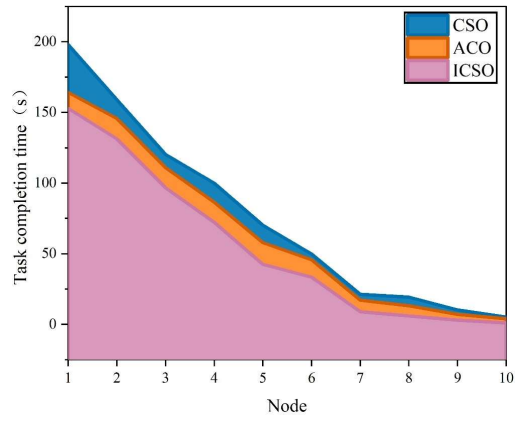


Figure 3: Completion time

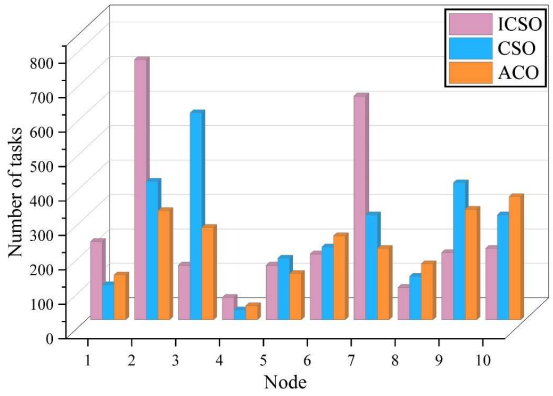


Figure 4: Load distribution on different nodes

IV. E. Comparative Resource Load Analysis

For 10 nodes, the heterogeneous cloud resource load of the proposed ICSO algorithm with CSO algorithm and ACO algorithm in this paper is specifically shown in Fig. 4. From the figure, it can be seen that the task load on different nodes varies mainly because of the difference in processing capability between them, compared to CSO algorithm and ACO algorithm, ICSO algorithm can better equalize the task load on each node, while the nodes with strong processing capability of CSO algorithm and ACO algorithm are assigned with fewer tasks, while the nodes with poor processing capability are task-heavy, and the load is extremely unbalanced on each node.

V. Conclusion

This paper proposes an improved competitive particle swarm optimization (ICSO) algorithm to solve the heterogeneous cloud resource scheduling problem and enhance the efficiency of heterogeneous cloud resource scheduling and allocation. Algorithm simulation is carried out on the cloud computing platform to test the optimization performance of the ICSO algorithm in this paper. In five function tests, including Sphere function, Rosenborock function, Rastrigin function, Griewank function, and Ackley function, the ICSO algorithm converges significantly faster than the CSO algorithm, and the ACO algorithm has a high convergence accuracy and more stable performance.

Using the WorkflowSim simulation platform as the experimental environment, heterogeneous cloud resource scheduling simulation experiments are carried out to verify the resource scheduling optimization effect of the heterogeneous cloud resource scheduling algorithm based on ICSO in this paper. The ICSO algorithm in this paper has lower GD values on all 15 experimental data, while the CSO algorithm and the ACO algorithm have a more difficult convergence performance under the same conditions, and their performance is inferior to that of the ICSO algorithm. In the hypervolume comparison analysis, the ICSO algorithm obtains a larger HV value and has an advantage in the test results, showing better diversity and convergence. Comparing the optimal solution change curves of the algorithms, the ICSO algorithm in this paper obtains the optimal solution for heterogeneous cloud resource scheduling in about 100 generations, which is much faster than CSO algorithm and ACO algorithm. By randomly assigning 200 heterogeneous cloud resource scheduling tasks to 10 resource nodes, the ICSO algorithm in this paper takes relatively less time to complete the tasks, and is able to better equalize the task load of each node.

Overall, the ICSO-based heterogeneous cloud resource scheduling algorithm proposed in this paper can be well applied to the heterogeneous resource scheduling problem in large-scale cloud computing environments to improve the efficiency of heterogeneous cloud resource scheduling and allocation.

References

[1] Amer, A. A., Talkhan, I. E., Ahmed, R., & Ismail, T. (2022). An optimized collaborative scheduling algorithm for prioritized tasks with shared resources in mobile-edge and cloud computing systems. Mobile Networks and Applications, 27(4), 1444-1460.

[2] Chen, G., Qi, J., Sun, Y., Hu, X., Dong, Z., & Sun, Y. (2023). A collaborative scheduling method for cloud computing heterogeneous workflows based on deep reinforcement learning. Future Generation Computer Systems, 141, 284-297.

[3] Liu, S., & Wang, N. (2020). Collaborative optimization scheduling of cloud service resources based on improved genetic algorithm. IEEE Access, 8, 150878-150890.

[4] Zhan, Z. H., Liu, X. F., Gong, Y. J., Zhang, J., Chung, H. S. H., & Li, Y. (2015). Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Computing Surveys (CSUR), 47(4), 1-33.

[5] Khallouli, W., & Huang, J. (2022). Cluster resource scheduling in cloud computing: literature review and research challenges. The Journal of supercomputing, 78(5), 6898-6943.

[6] Visalaxi, G., & Muthukumaravel, A. (2024). Towards Quantum Computing-Inspired Evolutionary Algorithm for Optimized Cloud Resource Management. International Journal of Intelligent Engineering & Systems, 17(3).

[7] Mishra, K., Pradhan, R., & Majhi, S. K. (2021). Quantum-inspired binary chaotic salp swarm algorithm (QBCSSA)-based dynamic task scheduling for multiprocessor cloud computing systems. The Journal of Supercomputing, 77, 10377-10423.

[8] Sharma, N., Tyagi, S., & Atri, S. (2017). A comparative analysis of min-min and max-min algorithms based on the makespan parameter. International Journal of Advanced Research in Computer Science, 8(3), 1038-1041.

[9] Samadi, Y., Zbakh, M., & Tadonki, C. (2018, July). E-HEFT: enhancement heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing. In 2018 International Conference on High Performance Computing & Simulation (HPCS) (pp. 601-609). IEEE.

[10] Pradhan, P., Behera, P. K., & Ray, B. N. B. (2016). Modified round robin algorithm for resource allocation in cloud computing. Procedia Computer Science, 85, 878-890.

[11] Chen, H., Wang, F., Helian, N., & Akanmu, G. (2013, February). User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing. In 2013 national conference on parallel computing technologies (PARCOMPTECH) (pp. 1-8). IEEE.

[12] Raeisi-Varzaneh, M., Dakkak, O., Fazea, Y., & Kaosar, M. G. (2024). Advanced cost-aware Max–Min workflow tasks allocation and scheduling in cloud computing systems. Cluster computing, 27(9), 13407-13419.

[13] Ahmed, Z., Ashrafi, A. F., & Mahbub, M. (2017). Clustering based Max-Min Scheduling in Cloud Environment. International Journal of Advanced Computer Science and Applications, 8(9).

[14] Jawade, P., Borkar, G. M., & Ramachandram, S. (2022). Confinement forest‐based enhanced min‐min and max‐min technique for secure multicloud task scheduling. Transactions on Emerging Telecommunications Technologies, 33(9), e4515.
[15] Ahmed, S., & Omara, F. A. (2022). An Enhanced Workflow Scheduling Algorithm for Cloud Computing Environment. International Journal of Intelligent Engineering & Systems, 15(6).
[16] Dubey, K., Kumar, M., & Sharma, S. C. (2018). Modified HEFT algorithm for task scheduling in cloud environment. Procedia Computer Science, 125, 725-732.
[17] Gupta, S., Iyer, S., Agarwal, G., Manoharan, P., Algarni, A. D., Aldehim, G., & Raahemifar, K. (2022). Efficient prioritization and processor selection schemes for heft algorithm: A makespan optimizer for task scheduling in cloud environment. Electronics, 11(16), 2557.
[18] Hai, T., Zhou, J., Jawawi, D., Wang, D., Oduah, U., Biamba, C., & Jain, S. K. (2023). Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes. Journal of Cloud Computing, 12(1), 15.
[19] Faragardi, H. R., Sedghpour, M. R. S., Fazliahmadi, S., Fahringer, T., & Rasouli, N. (2019). GRP-HEFT: A budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds. IEEE Transactions on Parallel and Distributed Systems, 31(6), 1239-1254.
[20] Belgaum, M. R., Musa, S., Mazliham, M. S., & Alam, M. (2019). A behavioral study of task scheduling algorithms in cloud computing. International Journal of Advanced Computer Science and Applications, 10(7).
[21] Elmougy, S., Sarhan, S., & Joundy, M. (2017). A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique. Journal of Cloud computing, 6, 1-12.
[22] Samal, P., & Mishra, P. (2013). Analysis of variants in round robin algorithms for load balancing in cloud computing. International Journal of computer science and Information Technologies, 4(3), 416-419.
[23] Ghazy, N., Abdelkader, A., Zaki, M. S., & Eldahshan, K. A. (2023). An ameliorated Round Robin algorithm in the cloud computing for task scheduling. Bulletin of Electrical Engineering and Informatics, 12(2), 1103-1114.
[24] Zohora, M. F., Farhin, F., & Kaiser, M. S. (2024). An enhanced round robin using dynamic time quantum for real-time asymmetric burst length processes in cloud computing environment. PloS one, 19(8), e0304517.
[25] Zou, C., Lu, Y., Zhang, F., & Sun, S. (2012, October). Load-based controlling scheme of virtual machine migration. In 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems (Vol. 1, pp. 209-213). IEEE.
[26] Lahande, P. V., Kaveri, P. R., Saini, J. R., Kotecha, K., & Alfarhood, S. (2023). Reinforcement learning approach for optimizing cloud resource utilization with load balancing. IEEE Access, 11, 127567-127577.
[27] Jangra, A., & Mangla, N. (2023). An efficient load balancing framework for deploying resource schedulingin cloud based communication in healthcare. Measurement: Sensors, 25, 100584.
[28] PushpaLatha, K., Shaji, R. S., & Jayan, J. P. (2014). A cost effective load balancing scheme for better resource utilization in cloud computing. Journal of Emerging Technologies in Web Intelligence, 6(3), 280-290.
[29] Yiru Zhang. (2024). Efficient Storage and Access Management System for Network Databases Driven by Cloud Computing Technology. Academic Journal of Engineering and Technology Science,7(6).
[30] Dandan Wang,Jian Guan,Hongyan Liu,Hanwen Zhang,Qi Wang,Lijian Zhang & Jingzheng Dong. (2025). Low-Carbon Control of Integrated Energy by Combining Cuckoo Search Algorithm and Particle Swarm Optimization Algorithm. Sustainability,17(7),3206-3206.
[31] A.P. Aseel W. Ali,Lecturer Riyadh Zaghlool Mahmood & A.P. Hanan H. Ali. (2024). Pulling color from a colored image for coloring gray imagesusing CSO algorithm. PRZEGLĄD ELEKTROTECHNICZNY,(1).
[32] Pengfei Qian,Chenwei Pu,Lei Liu,Hui Luo,Jie Wu,Yifan Jia... & Luis Miguel Ruiz Páez. (2024). Ultra-high-precision pneumatic force servo system based on a novel improved particle swarm optimization algorithm integrating Gaussian mutation and fuzzy theory. ISA transactions,152,453-466.