Construction of Cloud Computing Task Scheduling Model Based on Simulated Annealing Hybrid Algorithm

Kejin Lv, Tianxu Huang*
College of Information Engineering, Guangxi City Vocational University, Chongzuo, 532200, China

***Abstract*—With the development of cloud computing technology, effective task scheduling can help people improve work efficiency. Therefore, this study presented a hybrid algorithm on the grounds of simulated annealing and taboo search to optimize task scheduling in cloud computing. This study presented a hybrid algorithm for optimizing the cloud computing task scheduling model. The model used simulated annealing algorithm and taboo search algorithm to convert the objective function into an energy function, allowing atoms to quickly arrange in terms of a certain rule for obtaining the optimal solution. The study analyzed the model through simulation experiments, and the experiment showed that the optimal value of the hybrid algorithm in high-dimensional unimodal testing was 7.15E-247, far superior to the whale optimization algorithm's 3.99E-28 and the grey wolf optimization algorithm's 1.10E-28. The completion time of the hybrid algorithm decreased with the growth of virtual machines, and the shortest time was 8.6 seconds. However, the load balancing degree of the hybrid algorithm increased with the growth of virtual machines. The final results indicated that the proposed hybrid algorithm exhibits high efficiency and superior performance in cloud computing task scheduling, especially when dealing with large-scale and complex optimization problems.**

Keywords*—Simulated annealing algorithm; taboo search optimization algorithm; cloud computing; task scheduling; completion time; load balancing degree

I. Introduction

As the boost of Cloud Computing (CC) technology, it has been the preferred platform for modern enterprises and research institutions to handle large-scale data and complex computing tasks [1]. In this context, an efficient CC Task Scheduling (TS) strategy is crucial for optimizing resource allocation, improving processing efficiency, and reducing operational costs [2]. TS, as a core issue in CC environments, directly affects the overall efficiency and user satisfaction of cloud services [3-4]. At present, many optimization algorithms have been proposed in the field of CC TS, but these algorithms still face many challenges when handling large-scale, multi-objective, and dynamically changing scheduling problems [5-6]. Therefore, to solve the problem of how to achieve high efficiency and economy in TS in CC while ensuring service quality, a hybrid optimization algorithm based on Simulated Annealing (SA) and Taboo Search (TS) is proposed. This algorithm combines the global search capability of SA and the efficient optimization characteristics of TS, and can solve multi-objective optimization problems in CC TS. The innovation of this research lies in effectively combining the advantages of two optimization algorithms for enhancing TS.

The main contribution of the research is to propose a hybrid optimization algorithm combining SA and TS to effectively solve the multi-objective optimization problem of CC TS, and provide practical and feasible solutions for the field of CC. This method can improve the efficiency and economy of TS while ensuring service quality. This algorithm outperforms traditional single optimization methods, especially in handling large-scale and dynamically changing scheduling tasks. The research mainly verifies the effectiveness of the model in improving the overall efficiency and user satisfaction of cloud services through performance analysis.

The research structure mainly includes six sections. Section II is for summarizing the research results of scholars around the world on SA and CC TS. Section III is for building a CC TS model and analyze the application of SA and TS algorithms in the model. Section IV analyzes the performance of the constructed model through testing functions and simulation experiments. Discussion is given in Section V. Finally, Section VI concludes the paper.

II. Related Works

As the boost of CC, a good scheduling algorithm can effectively help enhance the efficiency of CC. SA has been extensively utilized due to its powerful search capabilities. Zolfi K et al. studied the continuous form of multi-layer dynamic facility layout problem, using an approximate Optimal Solution (OS) method of SA metaheuristic algorithm, and running the proposed algorithm in MATLAB software. Through experimental results analysis, SA successfully found suitable solutions for each test case, and comparative experiments showed that SA has better solving ability [7]. Moradi N proposed a new population-based SA algorithm and applied it to solve the 0-1 knapsack problem. The calculation results indicated that the proposed population-based SA is the most effective optimization algorithm for KP01 among all SA based solvers, achieving the goal of putting projects with total profits into the backpack [8]. Abdel Asset M et al. proposed a hybrid version of the Harris Hawks optimization algorithm on the grounds of bitwise operations and simulated annealing (HHOBSA) for addressing feature selection problems. They compared and analyzed the proposed HHOBSA algorithm using 24 standard datasets and 19 manual datasets, and found

from the relevant outcomes that the HHOBSA algorithm possesses more excellent performance compared to others [9]. Tanha M et al. proposed a new theorem and applied it to generate an initial population of semiconductors. In genetic algorithms (GA) with global trends, it performed crossover operators to explore the search space. After obtaining the appropriate solution, it would randomly call one of the three novel neighbor operators to potentially enhance the given solution. The relevant outcomes showed that relative to other comparative algorithms, the proposed hybrid algorithm has advantages of 10.17%, 9.31%, 7.76%, and 8.21% in terms of production span, plan length ratio, acceleration, and efficiency, respectively [10]. Fontes D and other researchers proposed a Hybrid Particle Swarm Optimization Simulated Annealing Algorithm (PSOSA) to solve job shop scheduling problems. This method integrated the search capability of particle swarm optimization (PSO) with the local search advantage of SA to handle the integrated scheduling of production and transportation in manufacturing systems. Extensive computational experiments validated the effectiveness of PSOSA on 73 benchmark instances. The results showed that the algorithm outperforms existing technologies in shortening manufacturing cycles and exit times, and demonstrated a high degree of robustness [11].

There are also many scholars who have conducted different analyses on CC TS models. Fu X et al. studied the process of cloud TS and presented a PSO genetic hybrid algorithm with phagocytic effect. Firstly, it divided each generation of particle swarm and used the phagocytic mechanism and GA's cross mutation to change the position of particles in the subpopulation. Then, a feedback mechanism was utilized for ensuring that the particle population always moves in the direction of the OS. Through the simulation, the algorithm markedly enhanced the overall Completion Time (CT) of cloud tasks and possessed higher convergence accuracy [12]. Hamed A Y et al. presented a TS algorithm on the grounds of GA. The goal of this algorithm was for minimizing the CT and execution cost of tasks, and maximized resource utilization. The outcomes showcased that the proposed method can find the OS for CT, execution cost, and resource utilization [13]. Pirozmand et al. proposed a two-step hybrid method for scheduling tasks that perceive energy and time, called GA and energy aware scheduling heuristic on the grounds of GA. The first step included determining the priority of the task, and the second step included assigning the task to the processor. They determined the priority of the task and generated the primary chromosome, and used an energy aware scheduling heuristic model for assigning the task to the processor. The simulation showcases that the proposed algorithm can outperform others [14]. Bezdan T et al. presented a hybrid bat algorithm on the grounds of multi-objective TS, and conducted experiments on the CloudSim toolkit utilizing standard parallel workloads and synthetic workloads. It compared the obtained results with other similar meta heuristic techniques evaluated in the same situation. The simulation showcased the enormous potential of their proposed method [15]. Khan M and other scholars proposed a TS method based on hybrid optimization algorithms, aiming at effectively scheduling jobs in CC environments and minimizing waiting time. This method combined the advantages of ant colony algorithm and PSO, improving task allocation and resource utilization. Through simulation testing, the scheduling strategy showed better performance than traditional methods in multiple parameters such as total production time, execution time, waiting time, efficiency, and utilization. This study highlighted the practicality and efficiency of hybrid optimization strategies in handling large-scale CC resource allocation [16].

In summary, although the above studies have achieved good results in their respective application scenarios, these research methods still have certain limitations. Most studies only focus on a single algorithm, lacking consideration for the diverse and dynamic characteristics of CC environments. Therefore, this study constructs a CC TS model from the perspective of combining multiple algorithms. By combining the advantages of SA's extensive search ability and TS's fast search, a CC TS model on the grounds of SA hybrid algorithm is proposed.

III. Construction of CC TS Model with Improved SA Algorithm

This study focuses on the TS problem of CC. Firstly, a scheduling model for CC will be constructed, and the basic principles of CC TS will be analyzed. Aiming at addressing the issues of low efficiency and high resource consumption in TS, this study aims to optimize and improve the CC TS model using the search capability of SA. Meanwhile, it adopts TS algorithm for adopting the convergence of the TS model. This is to build an efficient and reasonable scheduling algorithm that reduces costs while improving user satisfaction.

A. Construction of CC Scheduling Model

With the advent of the information age, the amount of data information is constantly increasing, and the demand for server integration is increasing. Many high-performance storage technologies have emerged. These technologies have driven the advancement of virtualization technology, and with the continuous development and integration of various technologies, CC with stronger computing power and a wider range of application services has emerged. An example of CC is showcased in Fig. 1. In Fig. 1, CC is described as a multi-layered architecture that includes an infrastructure layer, a platform layer, and an application layer. The Infrastructure as a Service (IaaS) layer provides virtualized physical computing resources such as servers, storage, and network facilities. Platform as a Service (PaaS) provides development tools and runtime environments that enable developers to build and deploy applications. The application layer Software as a Service (SaaS) provides software applications directly to end users. The figure also shows that how cloud services are provided, namely the concepts of public cloud, private cloud, and hybrid cloud. In addition, the dynamic allocation process of CC resources is also reflected in the figure. Through this model, CC can maximize the utilization of resources, optimize computing power, and reduce the operating costs of enterprises.

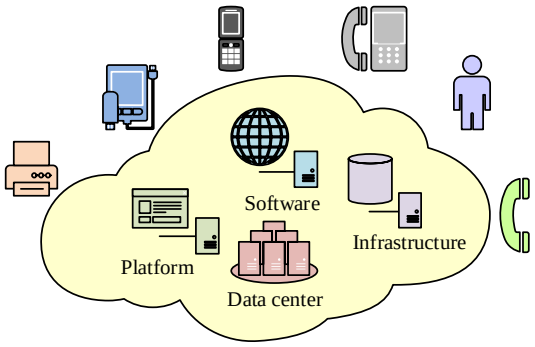


Fig. 1. CC example diagram.

Fig. 1 shows that CC is an Internet-based computing method, which offers shared processing resources and data for computers or other devices, and is a configurable computing resource that can be accessed as needed. CC is the advancement of parallel, distributed, and grid computing, which is a comprehensive evolution of concepts. Therefore, CC has advantages such as large-scale, high reliability, virtualization, high scalability, on-demand services, universality, and low cost, while also having disadvantages such as high dependence on networks and data security issues. In a cloud environment, TS first abstracts different types of hardware resources into Virtual Machines (VMs) using virtualization, then deploys tasks to these VMs, and finally the VMs execute these tasks. The TS model on the grounds of CC generally consists of two layers of scheduling. The first layer mainly solves the problem of matching VM resources with user tasks. The second layer mainly solves the problem of how to match VMs and physical machines [13]. The specific scheduling model of CC is showcased in Fig. 2.

Fig. 2 shows that the model has n tasks, m VMs, and k physical machines. The first layer of job level scheduling focuses on how to map tasks to VMs. The second layer of facility level scheduling focuses on how to allocate VMs to physical machines. When scheduling resources in CC, it is necessary to ensure that tasks are executed before the deadline, while also balancing the system's load and improving resource utilization. Therefore, it is crucial for introducing an efficient resource scheduling algorithm into the CC scheduling model.

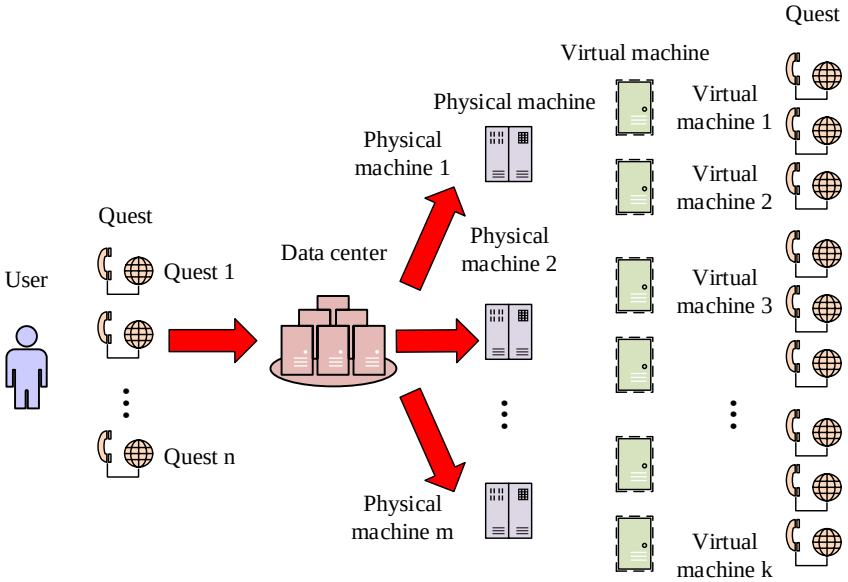


Fig. 2. CC scheduling model.

B. A CC Scheduling Model on the Grounds of SA Algorithm

SA essentially simulates the annealing process in thermodynamic systems, using the objective function as an energy function to slowly cool high-temperature objects and minimize the energy state of their internal molecules [14-16]. In SA, the atoms inside an object have multiple discrete states, each with corresponding state energy. After cooling, they reach thermal equilibrium, and the atoms are arranged according to a certain rule to reach a high-density, low-energy stable state. At this time, the stable state is equivalent to the global OS. SA will jump out of the local OS with a certain probability, which is directly relevant to the current state, temperature, and energy. The transition of annealing probability is shown in Fig. 3.

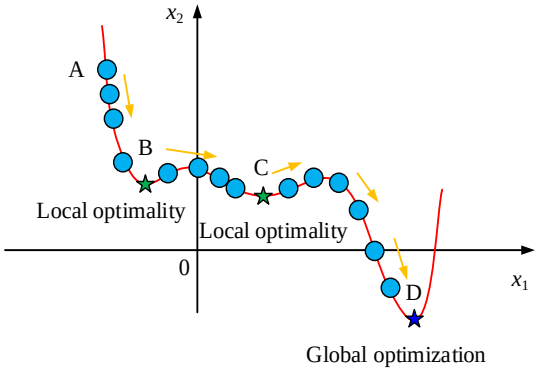


Fig. 3. Annealing probability transition diagram.

In Fig. 3, x_1 serves as the number of iterations and x_2 serves as the energy of the object. The research assumes that the initial state is point A, and as the iterations grows, the OS of the algorithm gradually updates to point B. At this point, the energy at point B is lower compared to point A, indicating that point B is closer to the OS. Therefore, the OS is directly transferred to point B. After reaching point B, the model continues to iterate and update, and the energy value increases. At this point, following the gradient descent rule does not allow the search to continue forward, and the algorithm will jump out of the local OS on the grounds of the state. Through repeated iterations, the final algorithm's OS will stabilize at point D. In SA, if time conditions permit, the higher the initial temperature set, the larger the search algorithm can perform, and the initial solution can be represented by Formula (1).

$$\exp(-\frac{\Delta f}{T_0}) \approx 1 \tag{1}$$

In Formula (1), Δf represents the difference in fitness function. When the algorithm has a large initial temperature, it can give the algorithm enough opportunities for jumping out of the local OS and achieve quasi equilibrium during iteration. But if the initial temperature is set too high, it will greatly increase the iterations and reduce its efficiency. Therefore, it is necessary to choose an appropriate initial temperature and ensure that the algorithm can obtain an approximate OS within the appropriate time. The speed of temperature update will affect the number of iterations and accuracy. The study uses temperature update as showcased in Formula (2).

$$T(k+1) = \alpha T(k) \tag{2}$$

In Formula (2), α represents the cooling parameter, which ranges from 0.5 to 0.99, $T(k)$ represents the current temperature, and $T(k+1)$ represents the temperature at the next time step. The expression of $T(k)$ is shown in Formula (3).

$$T(k) = \frac{(L-k)*T_0}{L} \tag{3}$$

In Formula (3), L serves as the total iterations, k serves as the current number of iterations, and T_0 serves as the initial temperature. The temperature update function can simply control the rate of temperature decrease, ensuring that the difference between control parameters remains unchanged. SA also includes Markov chain length and termination criteria. The length of the Markov chain represents the transformation interval generated by the Metropolis criterion during iteration, and the finite sequence Markov chain specifies the range of the algorithm's search space. The ending criterion of SA is generally set to three conditions, which are: when the temperature drops to a very small positive number and the temperature that has already dropped reaches a given constant. The current temperature has fallen into a local optimum and cannot escape from it. The local OS obtained by the algorithm is superior to the optimal value [17].

In CC scheduling, the optimization objectives of this study are system CT, system load balancing, and system execution cost. In the system CT, the user sends a task request, recording the task length as $Task_i$, the VM processing speed as $MIPS_j$, the TS and waiting time as $time_s$. The execution time of each task is $time_{ij}$, and the time required for each VM for executing all sub tasks assigned to it is T_j. The maximum CT of the system is shown in Formula (4) [18].

$$\begin{cases} Makespan = time_s + \max(T_j) \\ T_j = \sum_{i=1}^{n}\sum_{j=1}^{m} time_{ij} * x_{ij} \\ time_{ij} = Task_i / MIPS_j \end{cases} \tag{4}$$

Because the task requirements of users may involve computer related resources, the workload of VMs is represented by Formula (5) [19-20].

$$W_j = 1 - \frac{1}{k}\sum_{r=1}^{k}\frac{capacity_{jr} - sum(requested_r)}{capacity_{jr}} \tag{5}$$

In Formula (5), W serves as the workload of the VM, $capacity_{jr}$ serves as the total capacity of VM resources, and $request_r$ represents the total demand for VMs in all tasks. It simplifies the total workload of the VM through Formula (5), as shown in Formula (6).

$$W_{aj} = \sum_{i=1}^{n} W_j * x_{ij} \tag{6}$$

The load balancing degree of the system is obtained through Formula (6), and its expression is shown in Formula (7).

$$\begin{cases} B = \frac{1}{m}\sqrt{\sum_{j=1}^{m}\left|W_{aj} - \bar{W}_{aj}\right|^2} \\ \bar{W}_{aj} = \frac{1}{m}\sum_{j=1}^{m} W_{aj} \end{cases} \tag{7}$$

In Formula (7), B represents the load balancing degree. Regarding the optimization of system execution cost, the study represents the total execution cost per unit time of VMs as C, and the total execution time of VMs as T. The cost generated by a single VM and the total execution cost of the user are shown in Formula (8).

$$\begin{cases} cost(j) = c_j * T_j \\ Cost = \sum_{j=1}^{m} cost(j) \end{cases} \tag{8}$$

For the overall evaluation model of the system, the study weights three optimization objectives and changes the focus of scheduling objectives by changing the weight coefficients. The specific expression is shown in Formula (9).

$$C(s) = \mu_1 * Makespan + \mu_2 * B + \mu_3 * Cost \tag{9}$$

In Formula (10), μ_1, μ_2, and μ_3 respectively represent the weight coefficients of the three optimization objectives. $C(s)$ represents the value of the system objective function, and the smaller the fitness function value, the more excellent the overall performance of the system.

C. CC TS on the Grounds of SA Hybrid Algorithm

Although SA has the capability of jumping out of local optima, as the temperature parameter gradually decreases, the search ability weakens, and it may eventually fall into local optima. In the process of finding the global OS, the convergence speed (CS) is relatively slow. Therefore, the study combines the fast convergence of SA with the efficient optimization of TS to obtain the Integrated Simulated Annealing and Taboo Search (ISATS) algorithm. The TS process adopted by the research is shown in Fig. 4.

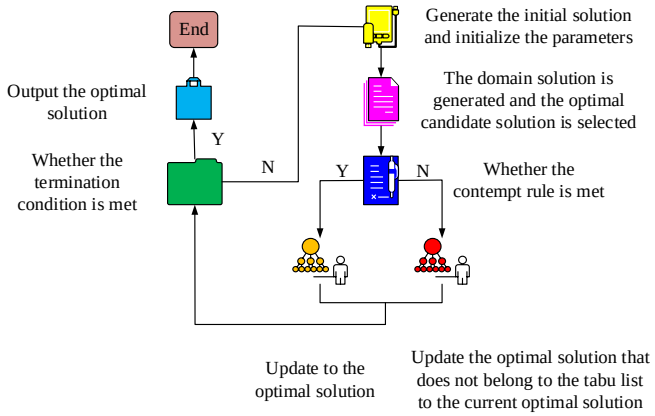


Fig. 4. Taboo search algorithm flowchart.

Fig. 4 shows the process in which the algorithm stores the searched candidate solutions in the taboo table during the search in the neighborhood. The solutions in the taboo table are prohibited from being searched again before being released. When a certain range of solutions are taboo, the contempt rule is used to release some OSs in the taboo table, thereby expanding the search range of the solution space and obtaining the global OS. In the ISATS algorithm, the study uses fitness variance to determine whether the algorithm is trapped in a local OS, and its expression is shown in Formula (10).

$$\sigma^2 = \sum_{i=1}^{M} \left(\frac{aff(s_i) - \overline{aff}(s)}{\max\left\{ \left| aff(s_i) - \overline{aff}(s) \right| \right\}} \right) \tag{10}$$

In Formula (11), σ represents the variance of fitness, $\overline{aff}(s)$ represents the average fitness, and $\max\left\{ \left| aff(s_i) - \overline{aff}(s) \right| \right\}$ represents the maximum difference in fitness among the population. This study sets a reasonable judgment threshold of σ_0, and when $\sigma^2 < \sigma_0$ is met, it indicates that the algorithm has completed initial convergence. After introducing TS into the TS model, the workload factor of the VM is represented by Formula (11).

$$B_j = 1 - \frac{T_j - \overline{T}_j}{T_{j\max} - T_{j\min}} \tag{11}$$

Formula (11) indicates that the longer the task execution time of a VM, the smaller the workload factor of the VM. The value of B_j determines the priority order of scheduling tasks. The study incorporates the workload factor of VMs into the Metropolis criterion, and the average workload factor of VMs is shown in Formula (12).

$$\overline{B} = \frac{1}{m} \sqrt{\sum_{j=1}^{m} \left| B_j - \overline{B}_j \right|^2} \tag{12}$$

In Formula (12), $\overline{B}$ represents the average load factor of the VM. The smaller its value, the smaller the load difference between VMs, indicating a more balanced load. At this point, the probability of accepting new solutions decreases. When there is a significant difference in load between VMs, it will increase the probability of accepting new solutions and make the algorithm jump out of the current solution to find a more excellent solution. The specific steps for studying and constructing the ISATS hybrid algorithm are shown in Fig. 5.

In Fig. 5, the study first utilizes the improved SA to quickly converge and obtain a current optimal task to VM mapping scheme. Then, this temporary optimal scheme is used as the initial solution of TS. In the subsequent process of adding the solution to the taboo table, the Metropolis criterion considering the load factor is introduced to achieve the goal of global optimization. The final scheme of task mapping to VM is obtained, and the task is executed using this result. Finally, Fig. 6 shows the pseudo-code of the ISATS hybrid algorithm.

In Fig. 6, the ISATS algorithm first initializes the initial solution, and then finds the OS through the SA algorithm. In each iteration, the algorithm generates a new solution and uses the accept-reject criterion to decide whether to accept the new solution. As the iteration progresses, the algorithm updates the weights and optimization targets, and uses Taboo tables to limit the search scope. Finally, the algorithm outputs the processed data set for subsequent use or further analysis. The algorithm may need to be adjusted according to the characteristics of the actual problem.

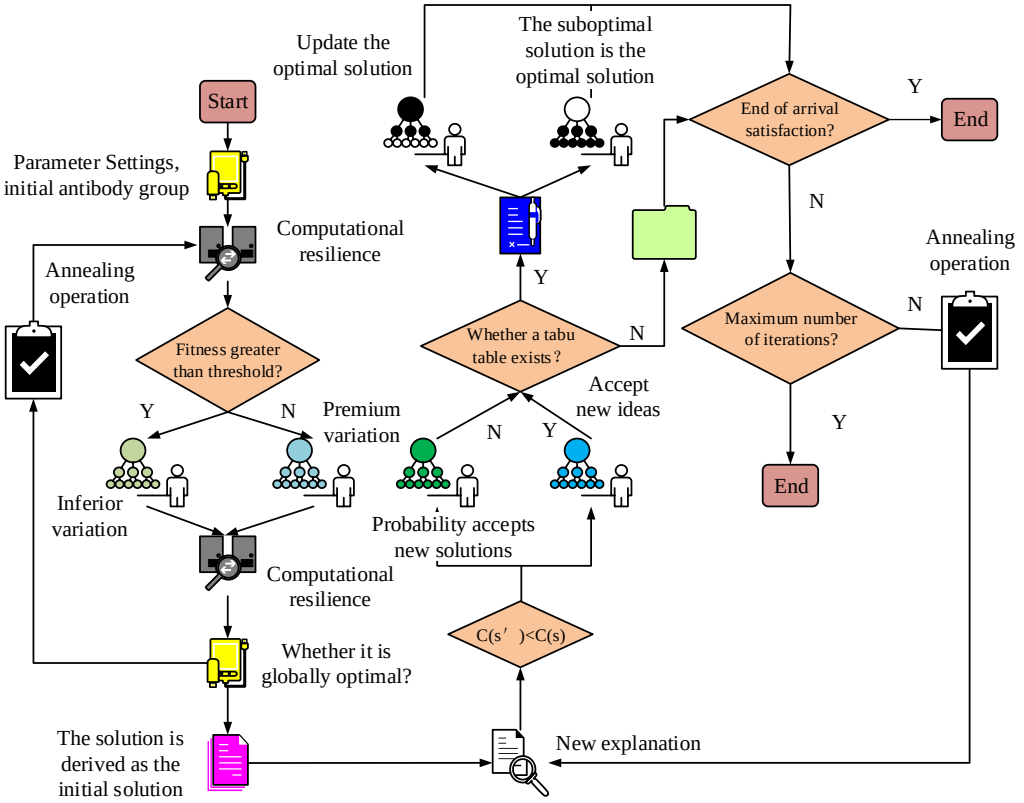


Fig. 5. ISATS hybrid algorithm flowchart.

```
Initialize initial solution (SA)
Set initial temperature T, cooling parameter p, maximum iteration count N
Define weights w1, w2, w3 for optimization objectives
DefineTaboo Table Length L

for i = 1 to N do:
# Update temporary solution using SA
temp_solution = SA algorithm to get optimal solution

while true:
# Generate new solution in the neighborhood
new_solution = Generate_Neighborhood(temp_solution)

# Metropolis criterion to accept or reject the new solution
if Metropolis_Criterion(new_solution, temp_solution) < Threshold:
break
else:
Accept new solutions (temp_solution = new_solution)
Update taboo table (store new solutions in taboo table and record relevant information)

# Update weights and optimization objectives
Update weight coefficients w1, w2, w3 based on fitness function values
Update scheduling results based on optimization objectives

end for
```

Fig. 6. Pseudo code of ISATS hybrid algorithm.

IV. Performance Analysis of CC TS Model on the Grounds of ISATS Hybrid Algorithm

For the ISATS hybrid algorithm proposed in the study, preliminary performance analysis of the algorithm was conducted through testing functions. The algorithm's optimal value, worst value, average value, and Standard Deviation (SD) were evaluated, and the superiority of the model was verified by comparing the algorithms. Then, the study verified the application effect of the ISATS hybrid algorithm in CC scheduling through simulation experiments. The experiment evaluated and analyzed the task CT and load balancing degree.

A. Test Functions and Parameter Settings

This study was simulated using the Cloudsim cloud simulation platform. To evaluate the algorithm performance, three algorithms were applied to high-dimensional (HD) unimodal and HD multimodal test functions, and 20 experiments were conducted. By comparing the optimal, average, worst fitness values, and SD obtained, it observed the convergence accuracy and stability. The comparative algorithms include Whale Optimization Algorithm (WOA), Grey Wolf Optimizer Algorithm (GWO), and ISATS algorithm. The specific selection of testing functions is showcased in Table I.

The study selected two sets of HD unimodal functions and two sets of HD multimodal functions for testing, with dimensions of 30 for all four functions. The experiment was for testing the optimal performance by setting the parameters of the CC TS model and algorithm. The specific parameter settings are showcased in Table II.

In the ISATS algorithm, the initial temperature was set to 100 ℃, the cooling parameter was 0.9, the termination temperature was 1 ℃, and the population size was 150. The weight coefficients for CT, load balancing, and execution cost were 0.4, 0.3, and 0.3. In the experimental environment, the CPU model was selected as Inter i5 12400F, and the GPU was selected as GeForce RTX ™ 2080 Ti, with a memory size of 2 * 8GB.

B. Performance Analysis on the Grounds of Test Functions

The study analyzed the performance of the ISATS algorithm through four testing functions, and the results of the HD unimodal testing function are showcased in Fig. 7. Fig. 7(a) showcases the F1 test function results, where the optimal value of ISATS was 7.15E-247, which is significantly better than the 3.99E-28 of the WOA algorithm and the 1.10E-28 of the GWO algorithm. In the average and SD results, the average of ISATS was 1.27E-229, with a SD of 0. The results indicated that the ISATS algorithm exhibits extremely high stability and superior optimization ability in multiple runs. Fig. 7(b) showcases the results of the F2 test function, where the optimal value of ISATS was 1.28E-144, which is also significantly better than WOA and GWO algorithms. This further proved that the ISATS algorithm has good optimization ability in the F2 test function.

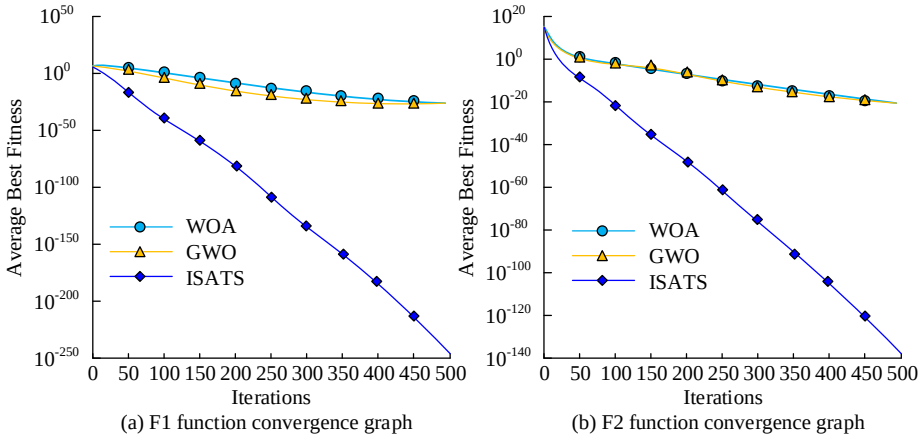
The outcomes of the HD multimodal test function are showcased in Fig. 8, where Fig. 8(a) and (b) represent the F3 and F4 test function results, respectively. The outcomes showcased that the ISATS algorithm has significantly improved CS and accuracy compared to the original algorithm in testing functions F3 and F4. Moreover, in F4, it even converged perfectly to the global OS, and by observing the SD, the ISATS algorithm had a slightly higher SD in the F3 function than the WOA algorithm. In the F4 function, the SD was significantly lower than that of the WOA and GWO algorithms, and it exhibited extremely fast CS. The CS, accuracy, and stability of the ISATS algorithm showed good performance through testing function analysis.

TABLE I. Test Function Expression and Related Parameters

Function	Expression	Dimension	Value range
HD unimodal function	$F_1(x)=\sum_{i=1}^{n} x_i^{\ 2}$	30	[-100,100]
	$F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[-10,10]
HD multimodal function	$F_3(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	[-500,500]
	$F_4(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	30	[-5.12,5.12]

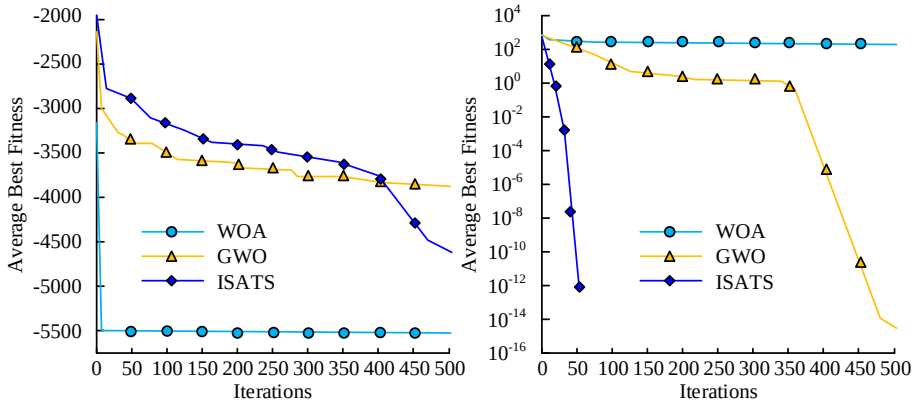
TABLE II. Task, VM, and Host Related Parameters

Argument	Value	Argument	Value	Argument	Value
Number of tasks	200	Task length	Rand (1000,10000)	Input file size	300MB
Output file size	300MB	Number of virtual machines	10	Virtual machine memory	512MB
VM broadband	500MB	Processing speed	1000	Processor core	1
Host memory	2GB	Host storage capacity	1000000MB	Host broadband	10000M



(a) F1 function convergence graph (b) F2 function convergence graph

Fig. 7. High dimensional unimodal test function results.

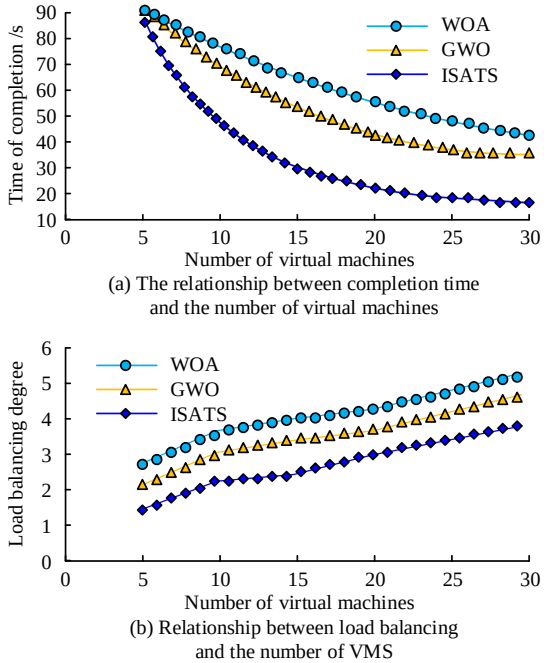


(a) F3 function convergence graph (b) F4 function convergence graph

Fig. 8. High dimensional multimodal test function.

C. Simulation Analysis on the Grounds of ISATS Algorithm

This study kept the number of CC tasks scheduled at 200 and observed the impact of different numbers of VMs on algorithm scheduling performance by increasing the number of VMs. The outcomes are showcased in Fig. 9. Fig. 9(a) showcases the influence of the number of VMs on the CT. The outcomes showed that the CT of each algorithm decreases with the increase of the number of VMs, with the shortest CT of the ISATS algorithm being 8.6 seconds. Fig. 9(b) showcases the influence of the number of VMs on load balancing. The outcomes showcased that the load balancing degree of each algorithm grows with the growth of the number of VMs, with the highest load balancing degree of GWO algorithm reaching 5.1. The outcomes showcased that the ISATS algorithm possesses high efficiency and low load balancing, and the selection of the number of VMs needs determining by actual needs. If model efficiency is taken as the primary consideration, the number of VMs can be increased. If load balancing is taken as the primary consideration, the number of VMs can be reduced.

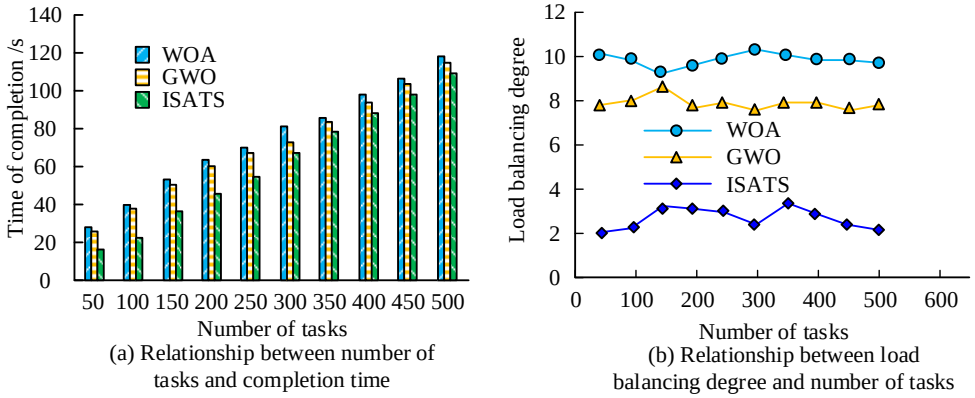


(a) The relationship between completion time and the number of virtual machines

(b) Relationship between load balancing and the number of VMS

Fig. 9. The impact of the number of VMs on the scheduling performance of the model.

When keeping the number of VMs at 20, this study observed the impact of task quantity on algorithm scheduling performance, and the outcomes are showcased in Fig. 10. Fig. 10(a) showcases the impact of work quantity on CT. Fig. 10(b) showcases the impact of workload on load balancing. In Fig. 10(a), compared to the GWO algorithm, as the tasks increased from 50 to 500, the ISATS algorithm reduced the CT by up to 28.8%. In Fig. 10(b), the system load of the ISATS algorithm was significantly improved. This may be because the ISATS algorithm utilizes the improved global search ability of SA in the early stage to provide TS with a good initial solution, while TS itself has strong optimization ability, which improves the final performance of the algorithm.



(a) Relationship between number of tasks and completion time

(b) Relationship between load balancing degree and number of tasks

Fig. 10. The impact of workload on scheduling performance.

To further analyze the scheduling performance of the model, the influence of observing the number of tasks on the algorithm's fitness value was studied, and the outcomes are showcased in Fig. 11. In Fig. 11, the overall fitness of the ISATS algorithm exceeded that of the WOA and GWO algorithms, and as the tasks grew, the absolute difference in fitness also gradually increased. The overall fitness value of the ISATS algorithm proposed in the study was about 5.6% better than that of the WOA algorithm. The reason is that the fitness value is directly relevant to the weight coefficient of the optimization objective, which makes the ISATS algorithm have a lower fitness value. On the grounds of the above results, the study proposed that the ISATS algorithm has good performance in both CT and load balancing, which can further optimize the CC TS model.

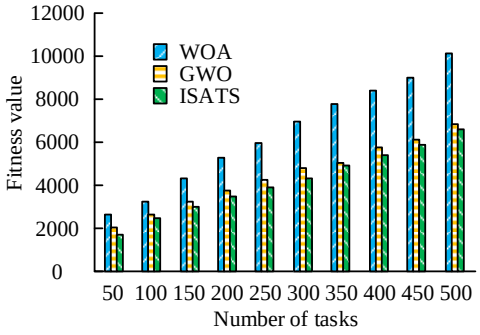


Fig. 11. The impact of workload on algorithm fitness values.

V. Discussion

In the study of CC TS, ISATS hybrid algorithm showed remarkable performance advantage. The experimental data indicated that the ISATS algorithm shows excellent CS and accuracy, which is superior to WOA and GWO algorithm in both unimodal and multimodal test functions. In addition, simulation experiments with different number of VMS and tasks showed that the ISATS algorithm can effectively reduce task CT and improve the load balancing degree of the system. In the HD multimodal test function, the ISATS algorithm not only rapidly converged to the global optimal solution, but also had significantly higher stability and optimization ability than the comparison algorithm, which verified the applicability and efficiency of the hybrid algorithm in dealing with complex optimization problems. The main reason is that the ISATS algorithm, by integrating SA and TS techniques, effectively avoided the common problem of falling into the local optimal, while enhancing the global search capability. The significant contribution of this study is to provide an efficient algorithmic framework for resource scheduling problems in large-scale and complex CC environments. The design of ISATS algorithm takes into account the computational efficiency and optimization quality, ADAPTS to the changing task demand and resource allocation state, and significantly improves the flexibility and response speed of TS. In addition to CC, the structure and performance characteristics of ISATS algorithms are also applicable to other areas requiring resource scheduling and optimization, such as big data processing, industrial automation, and intelligent transportation systems. By adjusting the parameters and optimizing the target, the ISATS algorithm can be widely used in a variety of compute-intensive and data-intensive application scenarios, and has wide application potential and practical value.

VI. Conclusion

A TS model for CC was studied to address the limitations in handling multi-objective tasks. By combining SA with TS and utilizing SA's global search ability and TS's efficient optimization ability, a CC TS model on the grounds of the ISATS algorithm was constructed. The model proposed in the study had an optimal value of 7.15E-247 in HD unimodal testing functions. In the average and SD results, the average value of ISATS was 1.27E-229, with a SD of 0. In HD and multimodal testing, the ISATS algorithm converged steadily to the global OS, demonstrating better search ability and stability.

In the simulation experiment, as the number of VMs increased, the CT of the ISATS algorithm was 8.6 seconds. Compared with the GWO algorithm, the CT was shortened by up to 28.8% as the tasks increased. The research results indicated that the ISATS algorithm showed significant advantages in CC TS, especially in dealing with large-scale TS problems, effectively improving efficiency and load balancing. Although the ISATS algorithm performed well in simulation experiments, there are still shortcomings. For example, the model still needs to be applied and analyzed in actual CC environments. It further optimizes algorithm parameters to adapt to more diverse application scenarios. In future research, the ISATS algorithm can be utilized to different CC scenarios, like cloud storage, big data processing, etc., to evaluate its performance in various applications.

FUNDING

The research is supported by Department of Education of Guangxi Zhuang Autonomous Region, Special Project for the Pilot Construction of the 1+X Certificate System in Guangxi Education Science during the 14th Five Year Plan for 2022, 2022ZJY2209 (Key Project): Research and practice on the talent training mode of vocational undergraduate big data major post course certificate under the background of 1+X certificate.

REFERENCES

[1] Ghannadi, Parsa, Seyed Sina Kourehli, and Seyedali Mirjalili. "A review of the application of the simulated annealing algorithm in structural health monitoring (1995-2021)." Frattura ed Integrità Strutturale , 2023, 17(64): 51-76.

[2] Abba Haruna A, Muhammad L J, Abubakar M. Novel Thermal-Aware Green Scheduling in Grid Environment. Artificial Intelligence and Applications, 2022, 1(4):244-251.

[3] Haznedar B, Arslan M T, Kalinli A. Optimizing ANFIS using simulated annealing algorithm for classification of microarray gene expression cancer data. Medical & Biological Engineering & Computing, 2021, 59(4): 497-509.

[4] Barkhordari M S, Tehranizadeh M. Response estimation of reinforced concrete shear walls using artificial neural network and simulated annealing algorithm//Structures. Elsevier, 2021, 34(1): 1155-1168.

[5] Yildiz B, Mehta P, Sait S, Panagant N, Kumar S, Yildiz A. A new hybrid artificial hummingbird-simulated annealing algorithm to solve constrained mechanical engineering problems. Materials Testing, 2022, 64(7): 1043-1050.

[6] Abualigah, Laith, and Ali Diabat. A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Cluster Computing, 2021, 24(1): 205-223.

[7] Pirozmand P, Hosseinabadi A, Mirkamali S, Li Y. An improved particle swarm optimization algorithm for task scheduling in cloud computing. Journal of Ambient Intelligence and Humanized Computing, 2023, 14(4): 4313-4327.

[8] Zolfi K, Jouzdani J. A mathematical model and a simulated annealing algorithm for unequal multi-floor dynamic facility layout problem based on flexible Bay structure with elevator consideration. Journal of Facilities Management, 2023, 21(3):352-386.

[9] Moradi N, Kayvanfar V, Rafiee M. An efficient population-based simulated annealing algorithm for 0–1 knapsack problem. Engineering with Computers, 2022, 38(3): 2771-2790.

[10] Abdel-Basset M, Ding W, El-Shahat D. A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection. Artificial Intelligence Review, 2021, 54(1): 593-637.

[11] Tanha M, Hosseini Shirvani M, Rahmani A M. A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments. Neural Computing and Applications, 2021, 33(1): 16951-16984.

[12] Fontes D, Homayouni S M, Gonçalves J F. A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources. European Journal of Operational Research, 2023, 306(3): 1140-1157.

[13] Fu X, Sun Y, Wang H, Li H. Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm. Cluster Computing, 2023, 26(5): 2479-2488.

[14] Hamed A Y, Alkinani M H. Task scheduling optimization in cloud computing based on genetic algorithms. Computers, Materials & Continua, 2021, 69(3): 3289-3301.

[15] Pirozmand P, Hosseinabadi A A R, Farrokhzad M, Sadeghilalimi M, Mirkamali S, Slowik A. Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing. Neural computing and applications, 2021, 33(1): 13075-13088.

[16] Bezdan T, Zivkovic M, Bacanin N, Strumberger I, Tuba E, Tuba M. Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm. Journal of Intelligent & Fuzzy Systems, 2022, 42(1): 411-423.

[17] Khan M, Santhosh R. Task scheduling in cloud computing using hybrid optimization algorithm." Soft computing, 2022, 26(23): 13069-13079.

[18] Abualigah L, Diabat A. A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Cluster Computing, 2021, 24(2): 205-223.

[19] Ghafari R, Kabutarkhani F H, Mansouri N. Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review. Cluster Computing, 2022, 25(2): 1035-1093.

[20] Praveenchandar J, Tamilarasi A. Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing. Journal of Ambient Intelligence and Humanized Computing, 2021, 12(1): 4147-4159.

[21] NoorianTalouki R, Shirvani M H, Motameni H. A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms. Journal of King Saud University-Computer and Information Sciences, 2022, 34(8): 4902-4913.

[22] Weiqing G E, Yanru C. Task-scheduling algorithm based on improved genetic algorithm in cloud computing environment. Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering), 2021, 14(1): 13-19.

[23] Bagheri A, Bagheri M, Lorestani A. Optimal reconfiguration and DG integration in distribution networks considering switching actions costs using Taboo search algorithm. Journal of Ambient Intelligence and Humanized Computing, 2021, 12(1): 7837-7856.

[24] Ahmed Z H, Yousefikhoshbakht M. An improved Taboo search algorithm for solving heterogeneous fixed fleet open vehicle routing problem with time windows. Alexandria Engineering Journal, 2023, 64(2): 349-363.

[25] Umam M S, Mustafid M, Suryono S. A hybrid genetic algorithm and Taboo search for minimizing makespan in flow shop scheduling problem. Journal of King Saud University-Computer and Information Sciences, 2022, 34(9): 7459-7467.