Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2025 SCPE. Volume 26, Issues 4, pp. 1886–1894, DOI 10.12694/scpe.v26i4.4612

GENETIC ANT COLONY ALGORITHM AND ITS DESIGN AND RESEARCH IN CLOUD COMPUTING PLATFORM RESOURCE SCHEDULING

DONGHUI MEI*, WENWEI SU†, YAN SHI ‡, AND YANXU JIN§

Abstract. In order to solve the problems of slow convergence speed and low efficiency in finding precise solutions in existing cloud computing resource scheduling algorithms, the author proposes a genetic ant colony algorithm and its design and research in cloud computing platform resource scheduling. The author introduces a hybrid algorithm that integrates genetic algorithms with ant colony optimization. This approach begins by encoding parameters and seeks the best combination through evolutionary processes. It effectively merges the ant colony algorithm's feedback mechanism with the genetic algorithm's global search capabilities and rapid convergence. Then, multi-dimensional QoS constraints are proposed according to the needs of different users to perform local and global updates of pheromones. Finally, comparative simulation experiments were conducted on the cloud simulation platform CloudSim with simulated annealing algorithm (SA) and basic ant colony algorithm (ACO). The experimental results show that GAACO has a better time cost than ACO, but the time cost is longer than SA, and as the number of tasks increases, the time gap becomes larger. Compared with ACO, the time is reduced by 50.8%, and compared with SA, the time difference is 4%. Therefore, in terms of time cost, this algorithm is better than ACO. The algorithm proposed by the author effectively shortens the completion time of task scheduling, reduces operating costs, and has superior comprehensive performance.

Key words: Cloud computing, Resource scheduling, Genetic algorithm, Ant colony Pheromone

1. Introduction. Cloud computing, as a new type of business service model, has received widespread attention from both industry and academia since its proposal [1]. As research and applications in cloud computing advance, cloud systems are expanding in scale and growing increasingly complex in their topology. Moreover, the diverse nature of resources presents significant challenges in efficiently scheduling cloud computing tasks, making it a critical area of focus in cloud computing research [2].

In recent years, cloud computing task scheduling and optimization algorithms have developed rapidly. Usually, the first step is to use Map/Reduce to process cloud computing tasks, dividing large-scale tasks into multiple subtasks, and then scheduling each subtask through Map and Reduce stages [3]. Studies indicate that scheduling tasks in cloud computing is classified as an NP problem. To address this, heuristic algorithms are predominantly employed, including optimization algorithms based on genetic algorithms (GA), particle swarm optimization, and ant colony optimization (ACO). These methods aim to minimize task completion times and ensure effective load balancing across node resources to better meet users' practical application requirements. Each of these algorithms has its own distinct strengths and weaknesses. For instance, genetic algorithms excel in global search but often require numerous parameters and can be prone to finding local optima. Ant colony algorithms are strong in local search but may experience slow initial search due to the lack of initial pheromone levels [4]. Particle swarm optimization offers high efficiency early on but can suffer from slow convergence and instability later in the process. The genetic ant colony algorithm integrates the genetic algorithm's global search strengths with the ant colony algorithm's local search capabilities, thereby enhancing both the optimization efficiency and solution quality [5].

In cloud computing environments, resource scheduling problems are highly complex and dynamic, and the introduction of genetic ant colony algorithm provides a new direction for solving this problem. Through this algorithm, cloud computing platforms can allocate computing resources more intelligently, maximize resource

*Information Center of Yunnan Power Grid Co., Ltd., Kunming, Yunnan, 650032, China. (Corresponding author, lrzhoujinyu@163.com)
†Information Center of Yunnan Power Grid Co., Ltd., Kunming, Yunnan, 650032, China.
‡Information Center of Yunnan Power Grid Co., Ltd., Kunming, Yunnan, 650032, China.
§Information Center of Yunnan Power Grid Co., Ltd., Kunming, Yunnan, 650032, China.

utilization, and reduce operating costs and energy consumption while meeting user needs.

The author's goal is to design and optimize a cloud computing platform resource scheduling model based on genetic ant colony algorithm, aiming to improve the efficiency and fairness of resource scheduling [6].

2. Literature Review. With the advancement of technology, high-performance computing has been increasingly applied in fields such as climate simulation, fluid mechanics, molecular dynamics, and bioinformatics [7]. The speed of data processing and the response time to user demands cannot be effectively improved for high-performance computing under high-performance concurrency, multiple computing system models, data and cloud storage.

How to allocate resources, energy-saving scheduling and load balancing for high-performance computing system platforms is the core of enhancing performance. In order to improve the utilization rate of high-performance computing systems and reduce system load imbalance, engineering scholars have conducted many scheduling algorithm studies [8]. Abbasi, S. et al. developed an algorithm that applies genetic algorithms to manage faults and costs in resource allocation for services. The core idea is to leverage genetic algorithms to choose the most suitable resources for different services. The algorithm focuses on minimizing both processing and energy costs, with these costs serving as the objective function to drive the optimization process [9]. Malathi, K. et al. introduced a genetic algorithm-based system for scheduling independent tasks, which optimizes both time and resource usage while considering the safety requirements for task allocation.

Currently, a range of metaheuristic algorithms, including genetic algorithms, are employed to address task scheduling challenges [10]. Chu, L. et al. developed a collaborative scheduling model for managing multiple equipment resources at automated container terminals, aiming to reduce completion time and enhance loading and unloading efficiency. Their comparison of particle swarm optimization and genetic algorithms demonstrated that their proposed algorithm significantly boosts both global and local search capabilities in finding optimal solutions. Furthermore, their findings show that the collaborative scheduling approach, which takes into account mixed processes, effectively enhances the efficiency of automated container terminal operations. This research offers valuable insights for optimizing loading and unloading processes and improving coordinated scheduling at automated docks [11].

However, genetic algorithms have better search space solution capabilities and require more local parameters, making it easy to obtain locally excellent solutions; Ant colony algorithm has better ability to search for exact solutions, but due to insufficient initial pheromones, the initial search for solutions is slower. In response to the above shortcomings, the author proposes a genetic ant colony algorithm and its design and research in cloud computing platform resource scheduling. The genetic ant colony algorithm, which combines user multi-dimensional QoS (quality of service) constraints, integrates user needs into the algorithm in mathematical form, and uses genetic algorithm to encode heuristic factors, expected heuristic factors, and pheromone volatility coefficients. In the process of evolution, the optimal combination is found, which improves the convergence speed and global search ability of the ant colony algorithm, and optimizes the virtual machine resource load and user comprehensive cost.

3. Method.

3.1. Problem description of cloud computing resource scheduling. Virtualization has fundamentally transformed cloud computing compared to traditional distributed resource scheduling models. As illustrated in Figure 3.1, cloud computing resource scheduling involves breaking down each task into multiple independent subtasks [12].

Upon receiving a request, the system allocates a specific amount of virtual resources, with each subtask being assigned to a corresponding virtual resource node.

In cloud computing, resource scheduling is defined as follows: Tasks are segmented into independent subtasks and assigned to m virtual resource nodes for executio, where $m < n$.

Let $T = \{t_1, t_2, \cdots, t_n\}$ represent the set of subtasks, where $t_j (0 < j \leqslant n)$ represents the jth subtask.

Let $VM = \{vm_1, vm_2, \cdots, vm_m\}$, represent the set of virtual resource nodes, where $vm_i (0 < i \leqslant m)$ represents the i-th virtual resource node, and each t_j can only execute on one vm_i.

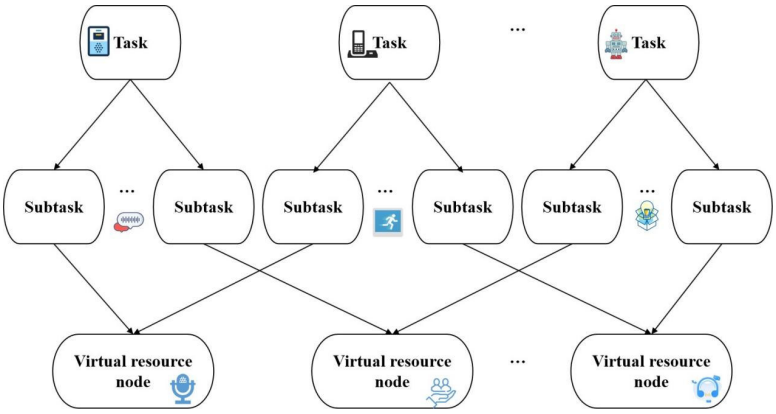

Fig. 3.1: Resource Scheduling in Cloud Computing

The correspondence between T and VM can be represented by the allocation matrix X as

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{3.1}$$

Here x_{ij} represents the correspondence between t_j and $vm_i, x_{ij} \in \{0,1\}, \sum_{i=1}^{m} x_{ij} = 1, i \in \{1,2,\cdots,m\}, j \in \{1,2,\cdots,n\}$, indicating that if t_j is executed on $vm_i, x_{ij} = 1$, otherwise, x_{ij}. The expected time for t_j to complete on vm_i is represented by ET_{ij}, which corresponds to the allocation relationship matrix X [13,14]. The ET matrix is

$$ET = \begin{bmatrix} ET_{11} & ET_{12} & \cdots & ET_{1n} \\ ET_{21} & ET_{22} & \cdots & ET_{2n} \\ \vdots & \vdots & & \vdots \\ ET_{m1} & ET_{m2} & \cdots & ET_{mn} \end{bmatrix} \tag{3.2}$$

If the starting time of vm_i is c_i, then the expected completion time of vm_i's processing task is $CT_i = c_i + \sum_{j=1}^{n} ET_{ij} \times x_{ij}$, where $i \in \{1,2,\cdots,m\}$ and $j \in \{1,2,\cdots,n\}$. Define $CT_{max} = max\{CT_i\}, CT_{min} = min\{CT_i\}$. Therefore, the expected time to complete the total task is CT_{max}, and the fitness function of the total task completion time is

$$f_{CT} = \frac{CT_{max} - CT_{min}}{\sum_{j=1}^{m}(CT_i - CT_{min})} \tag{3.3}$$

Similarly, the total cost completed by t_i on vm_j is represented by EC_{ij}. Let A_{ij} represent the resources occupied by t_i on vm_j, then EC_{ij} is positively correlated with the time consumed ET_{ij} and the resources occupied A_{ij}. Let its correlation coefficient be ξ, then

$$EC_{ij} = \xi \times ET_{ij} \times A_{ij} \tag{3.4}$$

Corresponding to the allocation relationship matrix X, the EC matrix is

$$EC = \begin{bmatrix} EC_{11} & EC_{12} & \cdots & EC_{1n} \\ EC_{21} & EC_{22} & \cdots & EC_{2n} \\ \vdots & \vdots & & \vdots \\ EC_{m1} & EC_{m2} & \cdots & EC_{mn} \end{bmatrix} \tag{3.5}$$

Let EC_i be the cost of processing task vm_i, where $i \in \{1, 2, \cdots, m\}$ is $EC_i = \sum_{j=1}^{n} EC_{ij} \times x_{ij}$. So, the fitness function of the total cost of task expenses is

$$f_{EC} = \frac{\sum_{i=1}^{m} EC_i}{\sum_{i=1}^{m} \sum_{j=1}^{n} EC_{ij}} \tag{3.6}$$

The resource scheduling adaptation function of the optimization algorithm is

$$F_{fitness} = a \times f_{cr} + b \times f_{EC} \tag{3.7}$$

In the formula: $a + b = 1, 0 \leqslant a, b \leqslant 1$. The goal of the algorithm is to find a suitable matrix X that minimizes the value of F_{fitues}.

3.2. Genetic Ant Colony Algorithm for Solving Cloud Computing Task Scheduling. The foundational model of the ant colony algorithm was developed by Italian researcher DIRIGO, drawing inspiration from the natural foraging behavior of ants. This behavior demonstrates self-organization and is effectively a solution to the shortest path problem, making it applicable to classic NP problems like the Traveling Salesman Problem (TSP) [15]. On the other hand, the genetic algorithm (GA) emulates Darwinian natural selection and genetic principles to search for optimal solutions through simulated evolution. By integrating genetic algorithms with ant colony algorithms, it is possible to enhance the global search capabilities and convergence speed of the ant colony approach.

The general process of using genetic ant colony algorithm to solve cloud computing task scheduling problems is as follows. Let $\alpha_q, \beta_q, \gamma_q$ be the values of the heuristic factor, expected heuristic factor, and pheromone volatilization coefficient corresponding to the q-th generation of the genetic population after encoding, crossover, mutation, and decoding, respectively. The heuristic factor signifies the influence of accumulated data as ants move, while the expected heuristic factor indicates the weight assigned to this heuristic information during the selection of virtual machines. The heuristic factor, expected heuristic factor, and pheromone volatilization coefficient are encoded in binary, with each parameter occupying 20, 20, and 40 binary bits, respectively; When selecting chromosomes, use roulette wheel to describe n virtual machines in the data center using $G(V, E)$, where V is the set of virtual machines and E is the set of virtual machine task execution time. Assuming there is an ant in the system, the ant's selection of the next task execution virtual machine is determined based on the amount of information on each virtual machine. Use taboo table to represent the virtual machines that the ant has already selected, and $allowed_k = \{V - tabu_k\}$ to represent the virtual machines that ant k can choose next, when the number of tasks exceeds the number of virtual machines, each virtual machine may be selected multiple times. The heuristic function $\eta_{is}(t) = 1/D_{ij}$ represents the expected transfer of ant k from virtual machine i to virtual machine s at time t, and $\tau_{is}(t)$ represents the residual information from virtual machine i to virtual machine s at time t. Initially, the information is the same, that is $\tau_{is}(0) = const$[16]. The probability $p_{ij}^k(t)$ of ant k transferring from virtual machine i to virtual machine j is:

$$p_{ij}^k(t) = \begin{cases} \frac{|\tau_{ij}(t)|^{\alpha_q} |\eta_{ij}(t)|^{\beta_q}}{\sum_{j \in allowed_k} |\tau_{ij}(t)|^{\alpha_q} |\eta_{ij}(t)|^{\beta_q}} & j \in allowed_k \\ 0 \quad j \neq allowed_k \end{cases} \tag{3.8}$$

When ants are in motion, in order to avoid excessive accumulation of pheromones, they need to update their pheromones after completing each scheduling, which can be done according to the following rules:

$$\tau_{ij}(t + n) = (1 - \gamma_q)\tau_{ij}(n) + \Delta\tau_{ij}(n) \tag{3.9}$$

$$\Delta\tau_{ij}(n) = Q/multiQoS_p \tag{3.10}$$

In the formula: Q is the intensity of pheromones; $multiQoS_p$ refers to multidimensional QoS constraints [17].

Considering the specifics of cloud computing task scheduling, the enhancements of the genetic ant colony algorithm over the standard ant colony algorithm are primarily evident in the following areas:

1. The heuristic factors, expected heuristic factors, and pheromone volatility coefficients have undergone cross variation in the genetic population, increasing the likelihood of understanding and improving global search capabilities.
2. In terms of pheromone updates, global and local updates have been carried out, while also incorporating the requirements for cost, time, system load, and service quality in cloud computing task scheduling, proposing multidimensional QoS constraints.

3.3. Establishment of Multidimensional QoS Constraint Function. The ordinary ant colony algorithm does not impose multidimensional QoS constraints when solving problems, but in the actual cloud computing scheduling process, there are corresponding requirements. Therefore, the author added multidimensional QoS constraints in three aspects: cost, time, and reliability, and established corresponding constraint functions. Let K_{best} be the set of tasks that allocate all current tasks to the best performing virtual machine, K_{woos} be the set of tasks that allocate all current tasks to the worst performing virtual machine, p be the number of the current ant, K be the task allocation scheme for the p ant with the number, K_j be the resource number assigned to the K_j-th task, and k be the number of current tasks to be allocated. Based on this, the author proposes the following hypotheses: Assumption 1 uses T_{min} to represent the time cost of allocating all current tasks to the best performing resources, T_{max} to represent the time cost of allocating all current tasks to the worst performing resources, T_c to represent the time cost of using genetic ant colony algorithm for task scheduling, and T_{res} to represent time constraints, which are mathematically defined as:

$$T_{min_p} = \frac{\sum_{i=1}^{k} J_i(length)}{k \cdot Vm_{best(mips)}} + \frac{\sum_{i=1}^{k} J_i(inputfileSize)}{k \cdot Vm_{best(bw)}} + \frac{\sum_{i=1}^{k} J_i(outfileSize)}{k \cdot Vm_{best(bw)}} \tag{3.11}$$

$$T_{max_p} = \frac{\sum_{i=1}^{k} J_i(length)}{k \cdot Vm_{worst(mips)}} + \frac{\sum_{i=1}^{k} J_i(inputfileSize)}{k \cdot Vm_{worst(bw)}} + \frac{\sum_{i=1}^{k} J_i(outfileSize)}{k \cdot Vm_{worst(bw)}} \tag{3.12}$$

$$T_{c_p} = \sum_{i=1}^{k} \frac{J_i(length)}{k \cdot Vm_{K_j(mips)}} + \sum_{i=1}^{k} \frac{J_i(inputfileSize)}{k \cdot Vm_{K_j(bw)}} + \sum_{i=1}^{k} \frac{J_i(outfileSize)}{k \cdot Vm_{K_j(bw)}} \tag{3.13}$$

$$T_{res_{K_j}} = \frac{T_{c_p} - T_{min_p}}{T_{max_p} - T_{min_p}} \tag{3.14}$$

Assuming that C_{min} represents the cost cost of allocating all current tasks to the best performing resources, C_{max} represents the cost of allocating all current tasks to the worst performing resources, C_c represents the cost of using genetic ant colony algorithm for task scheduling, and C_{res} represents the cost constraint, their mathematical definitions are:

$$C_{min_p} = \frac{\sum_{i=1}^{k} J_i(length)}{k \cdot Vm_{best(mips)}} \cdot perofmips + \frac{\sum_{i=1}^{k} J_i(inputfileSize)}{k \cdot Vm_{best(bw)}} \cdot perofbw + \frac{\sum_{i=1}^{k} J_i(outfileSize)}{k \cdot Vm_{best(bw)}} \cdot perofbw \tag{3.15}$$

$$C_{max_p} = \frac{\sum_{i=1}^{k} J_i(length)}{k \cdot Vm_{worst(mips)}} \cdot perofmips + \frac{\sum_{i=1}^{k} J_i(inputfileSize)}{k \cdot Vm_{worst(bw)}} \cdot perofbw + \frac{\sum_{i=1}^{k} J_i(outfileSize)}{k \cdot Vm_{worst(bw)}} \cdot perofbw \tag{3.16}$$

$$C_{c_p} = \frac{\sum_{i=1}^{k} J_i(length)}{k \cdot Vm_{K_j(mips)}} \cdot perofmips + \frac{\sum_{i=1}^{k} J_i(inputfileSize)}{k \cdot Vm_{K_j(bw)}} \cdot perofbw + \frac{\sum_{i=1}^{k} J_i(outfileSize)}{k \cdot Vm_{K_j(bw)}} \cdot perofbw \tag{3.17}$$

$$C_{res_{K_j}} = \frac{C_{c_p} - C_{min_p}}{C_{max_p} - C_{min_p}} \tag{3.18}$$

Table 3.1: Parameters of Genetic Ant Colony Algorithm

Parameter symbols	Parameter meaning	Parameter values
evolutionNum	evolutional generation	100
population	Genetic population size	10
m	Number of ants	30
P_c	crossover probability	0.34
P_m	mutation probability	0.09
$\aleph_{max}$	Maximum Inspiration Factor	1.00
β_{max}	Expected maximum inspiration factor	2.00
γ_{max}	Maximum evaporation coefficient of pheromones	0.10
Q	Pheromone intensity	50.00

In the formula, perofmips represents the cost of executing instructions per unit time; Perofbw is the cost per unit time bandwidth [18].

Assuming 3Res is the quantified reliability value when using genetic ant colony algorithm for task scheduling; ResRes is a reliability constraint, and its mathematical definitions are:

$$Res_p = \frac{1}{500} \sum_{i=1}^{k} \frac{J_i(length)}{Vm_{K_i}(mips)} \tag{3.19}$$

$$resRes_{K_j} = Res_p / antSize \tag{3.20}$$

Among them, antSize is the size of the ant population.

Assuming $4\lambda_1, \lambda_2, \lambda_3$ are weight coefficients ($\lambda_1 + \lambda_2 + \lambda_3 = 1$) for cost, time, and reliability, which can be adjusted according to user requirements, the author selects three coefficients of 0.3, 0.4, and 0.4, respectively. The final multidimensional QoS constraint function is defined as:

$$multiQoS_p = \lambda_1 \cdot Tres_p + \lambda_2 \cdot Cres_p + \lambda_3 \cdot resRes_p \tag{3.21}$$

3.4. Simulation experiment. In order to test the performance of the algorithm proposed in this article, the CloudSim 3.0.2 cloud simulation platform from a certain university's grid laboratory was used in the experiment, and simulation comparisons and result analysis were conducted with the basic ant colony algorithm (ACO) and simulated annealing algorithm (SA). In the cloud simulation platform, first create a new MyAll ocateTest class for initial configuration of the cloud environment, including the creation of the data center, initialization of the scale parameters of cloud computing tasks, task size and input/output data file size, creation of virtual machine resources. Each virtual machine resource encompasses attributes such as CPU count, memory size, bandwidth, and instruction processing speed. CloudSim objects are instantiated to introduce cloud computing tasks, and three algorithms—GAACO, ACO, and SA—are implemented within the DatacenterBroker class [19]. The parameters pertinent to the genetic ant colony algorithm are detailed in Table 3.1.

Compare the performance of genetic ant colony algorithm from four aspects, namely average time cost, average cost, algorithm service quality, and system resource load rate. At the beginning, the task size is 10, the cloud computing resources are 10, the virtual machine storage size is 10GB, the memory size is 256MB, the number of CPUs is 1, the bandwidth is 1000MB, the unit time bandwidth cost is 0.01 yuan/s, and the unit time instruction cost is 0.01 yuan/s. The task size is tested in increments of 10. The quality of algorithm services is reflected through multiQoS. Define the resource load rate as:

$$sysuse = \frac{1}{n} \sqrt{\sum_{i=1}^{n} (use_i - use_{Avg})^2} \tag{3.22}$$

In the formula: use_i is the load of the i-th resource; use_{Avg} is the average load of the system; n is the number of resources [20].

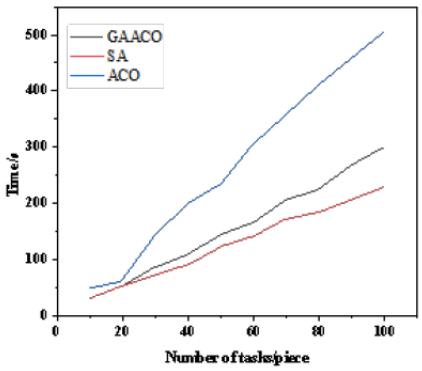

Fig. 4.1: Average time cost of each algorithm

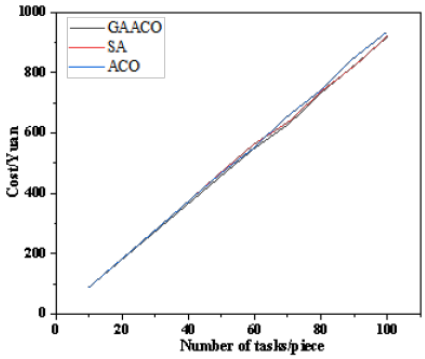

Fig. 4.2: Cost of each algorithm

4. Results and Discussion. With an increase of 10 tasks, the average time cost for each algorithm is computed, as illustrated in Figure 4.1. The results reveal that GAACO performs better in terms of time cost compared to ACO, although it still takes more time than SA. Additionally, as the task count grows, the time difference between GAACO and SA becomes more pronounced. Compared with ACO, the time is reduced by 50.8%, and compared with SA, the difference is 4%. Therefore, in terms of time cost, this algorithm is better than ACO, but not significantly different from SA.

The cost of each algorithm under different task quantities is shown in Figure 4.2. As the number of tasks increases by a multiple of 10, it can be seen that the differences between the algorithms are not significant, with an average cost difference of only about 1%.

The experimental results of the service quality of each algorithm are shown in Figure 4.3. It can be seen that the service quality of this algorithm and SA slowly increases with the increase of task quantity, while ACO shows a sharp upward trend in a straight line. Service quality is a comprehensive indicator of cost, time, and reliability. It can be seen that GAACO's overall performance is better than ACO and SA, with reductions of 14.3% and 76.7%, respectively.

The experimental results of the system load of each algorithm are shown in Figure 4.4. It can be seen that the system load of ACO has been maintained at a high level, while the system load of GAACO is higher than SA but significantly better than ACO. The average system load of GAACO is reduced by 50.1% compared to ACO. Combining Figures 4.1-4.3, it can be seen that SA is more inclined towards an average scheduling approach, where the number of tasks allocated to each virtual machine is basically the same. Therefore, the

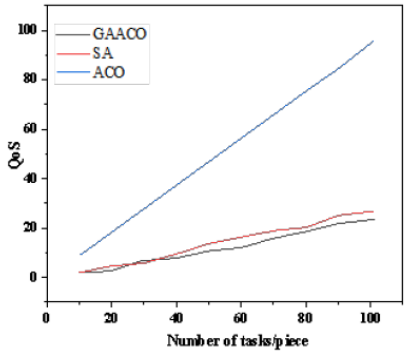

Fig. 4.3: Service Quality of Various Algorithms

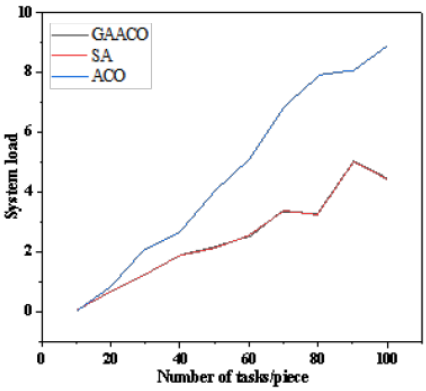

Fig. 4.4: Load of each algorithm system

load obtained using equation (3.22) is 0. However, cloud computing task scheduling often requires cost, time, and reliability to be balanced based on user requirements, which is reflected in QoS. In summary, the algorithm proposed by the author can better meet the needs of customers in the actual scheduling process.

5. Conclusion. The author introduces the genetic ant colony algorithm and explores its application in resource scheduling on cloud computing platforms. After a thorough investigation into cloud computing task scheduling, the proposed algorithm is evaluated against the basic ant colony algorithm and simulated annealing algorithm across four dimensions. The findings indicate that the new algorithm outperforms the other two, offering superior balance in terms of time cost, cost efficiency, reliability, and system load. This makes it a highly effective method for meeting the multi-dimensional QoS requirements of users.

REFERENCES

[1] Sodinapalli, N. P. , Kulkarni, S. , & Venkatareddy, S. P. . (2022). An efficient resource utilization technique for scheduling scientific workload in cloud computing environment. IAES International Journal of Artificial Intelligence, 11(1), 367-378.
[2] Hu, B. , Cao, Z. , & Zhou, M. . (2022). Scheduling real-time parallel applications in cloud to minimize energy consumption. IEEE transactions on cloud computing, 10(1), 662-674.
[3] Wang, Y. , Junqing, Y. U. , & Zhibin, Y. U. . (2023). Resource scheduling techniques in cloud from a view of coordination: a holistic survey. Frontiers of Information Technology & Electronic Engineering, 24(1), 1-40.
[4] Belgacem, A. , & Beghdad-Bey, K. . (2022). Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost. Cluster Computing, 25(1), 579-595.

[5] Sangani, S. , & Patil, R. . (2023). Reliable and efficient webserver management for task scheduling in edge-cloud platform. international journal of electrical and computer engineering, 13(5), 5922-5931.
[6] Huang, Q. , Sun, L. , & Chen, P. C. H. . (2022). Automatic scaling mechanism of intermodal edi system under green cloud computing. Journal of advanced transportation, 2022(Pt.7), 1-16.
[7] Bi, W. , Yu, F. , & Higgs, C. R. . (2022). Resource load prediction of internet of vehicles mobile cloud computing. Computers, materials & continua, 73(1 Pt.1), 165-180.
[8] Jabir, K. V. T. , Suseelan, D. P. , & Keerikkattil, P. M. . (2022). Multi-objective scheduling policy for workflow applications in cloud using hybrid particle search and rescue algorithm. Service Oriented Computing and Applications, 16(1), 45-65.
[9] Abbasi, S. , Rahmani, A. , Balador, A. ,& Sahafi, A. . (2023). A fault-tolerant adaptive genetic algorithm for service scheduling in internet of vehicles. Applied Soft Computing, 32(1), 1-31.
[10] Malathi, K. , Anandan, R. , & Vijay, J. . (2023). Cloud environment task scheduling optimization of modified genetic algorithm. J. Internet Serv. Inf. Secur., 13, 34-43.
[11] Chu, L. , Liang, D. , Zhou, Y. , & Zhang, J. . (2024). Optimal model and algorithm design for the multi-equipment resource collaborative scheduling of automated terminals considering the mixing process. Journal of Marine Science and Application, 23(2), 479-490.
[12] Wang Jia-Yi, R. L. Q. F. . (2022). Improved genetic algorithm to solve flexible job-shop scheduling problem. Manufacturing Automation, 44(12), 91-94.
[13] Barredo, P. , & Puente, J. . (2023). Precise makespan optimization via hybrid genetic algorithm for scientific workflow scheduling problem. Natural computing, 22(4), 615-630.
[14] Sun, H. . (2023). Optimizing manufacturing scheduling with genetic algorithm and lstm neural networks. International Journal of Simulation Modelling, 22(3), 508-519.
[15] Pekel, E. . (2022). A simple solution to technician routing and scheduling problem using improved genetic algorithm. Soft Computing, 26(14), 6739-6748.
[16] Yang, H. , Xia, T. , & Xia, Z. Z. D. . (2023). Multi-interference and multi-model dynamic scheduling of the small satellite based on dual population genetic algorithm. Journal of Internet Technology, 24(6), 1199-1209.
[17] Xia, J. , Yan, Y. , & Ji, L. . (2022). Retracted article: research on control strategy and policy optimal scheduling based on an improved genetic algorithm. Neural Computing and Applications, 34(12), 9485-9497.
[18] Abohamama, A. S. , El-Ghamry, A. , & Hamouda, E. . (2022). Real-time task scheduling algorithm for iot-based applications in the cloud–fog environment. Journal of Network and Systems Management, 30(4), 1-35.
[19] Zhou, M. T. , Ren, T. F. , & Feng, D. X. Y. . (2023). Task scheduling and resource balancing of fog computing in smart factory. Mobile networks & applications, 28(1), 19-30.
[20] Shirvani, M. H. , & Talouki, R. N. . (2022). Bi-objective scheduling algorithm for scientific workflows on cloud computing platform with makespan and monetary cost minimization approach. complex & intelligent systems, 8(2), 1085-1114.

Edited by: Hailong Li
Special issue on: Deep Learning in Healthcare
Received: Aug 16, 2024
Accepted: Sep 24, 2024