Research on Task Scheduling Based on Particle Swarm and Membrane Computing in Cloud Computing

Kun Li*, Liwei Jia, Xiaoming Shi
Computer Teaching and Research Section Department of Public Infrastructure, Henan Medical College, China
**Corresponding Author.*

Abstract

In view of the high scheduling cost and long time in cloud computing task scheduling, this paper proposes a task scheduling algorithm based on particle swarm and membrane computing. First, a scheduling model based on task cost and time is constructed. Second, the particle swarm algorithm is improved as follows: (1) Use chaos algorithm in the population to optimize; (2) Use domain factors to improve inter-individual Interaction; (3) Use a weighting factor to consider the influence of the number of iterations on the solution; (4) Use extreme perturbation to improve particle vitality; (5) Use Levy to improve particle update position. After each iteration, the membrane calculation is used to update the individual. Finally, in the simulation experiment, the algorithm in this paper is compared with the particle swarm algorithm, the improved particle swarm algorithm, and the membrane calculation algorithm. It has better performance in the two indicators of cost and time. The contrast effect.

Keywords: *Cloud computing, task scheduling, chaos, weight factor*

I. Introduction

Task scheduling can arrange reasonable execution of computing tasks submitted by users to the cloud platform, ensuring that users can enjoy high-performance computing, storage and other services at a lower cost. This technology has been widely recognized by the public. In the current technical environment, there are still problems with task scheduling technology. At the same time, the task scheduling problem itself is also a complex mathematical problem, namely NP problem [1,2]. Therefore, it is important to further study task scheduling technology to improve user experience and improve cloud computing service quality. The significance of research. Some scholars conduct research from the perspective of single use of meta-heuristic algorithms. For example, literature [3] proposed the use of an optimized genetic algorithm for cloud computing task scheduling. Simulation experiments show that the optimized genetic algorithm has better results; literature [3-4] a cloud computing scheduling algorithm based on particle swarm optimization algorithm-LBMPSO is proposed. Its core idea is to consider reliability, execution time, transmission time, manufacturing span, round-trip time, transmission cost and load balancing tasks and virtual machines, simulation Experiments show that the algorithm can save manufacturing span, execution time, round-trip time, and transmission cost; Literature [5] proposed a discrete symbiosis search algorithm for cloud computing task scheduling. The simulation results show that when the number of tasks becomes larger, the algorithm converges faster and has better results in large-scale scheduling problems. Literature [6] proposes an improved Henry gas solubility optimization algorithm for cloud computing task scheduling, which uses Henry gas solubility optimization and whale optimization algorithms. Fusion and simulation experiments show that the algorithm is superior to traditional meta-heuristic algorithms in task scheduling; literature [7] proposed a genetic algorithm based on electronic search for cloud computing task scheduling, taking full account of completion time, load balance, and resource utilization Parameters such as rate and multi-cost are used to improve the behavior of task scheduling. Simulation experiments show that the algorithm has good effects in task cost, time, and load balance. Literature [8] proposes to use whale optimization algorithm for cloud computing task scheduling, simulation The experiment shows that the whale algorithm has better results than the traditional meta-heuristic algorithm in terms of load balancing, cost and time; literature [9] proposes to use the Artificial Bee

Colony algorithm for cloud computing task scheduling, and the simulation experiment shows the use of the ABC algorithm. It can also improve the effect of task scheduling; other scholars use multiple meta-heuristics for cloud computing task scheduling. Literature [10] proposes the use of particle swarm and genetic algorithm fusion for cloud computing task scheduling. The simulation experiment shows After the fusion, the scheduling effect of the algorithm has been significantly improved; literature [11] uses the fusion of particle swarm optimization and ant colony algorithm for cloud computing task scheduling, and simulation experiments show that it has better results in terms of completion time and resource utilization.

In the above research, it is found that although a single meta-heuristic algorithm used for cloud computing task scheduling can achieve certain results, it has its own problems of slow algorithm convergence and low algorithm solution accuracy, and multiple heuristic algorithms are fused together The effect of is significantly better than a single heuristic algorithm, but there is a problem of high algorithm complexity. In this paper, the particle swarm algorithm and the membrane calculation are fused, and the particle swarm algorithm is improved. The particles after each iteration are updated using the membrane calculation. The simulation experiment shows that the fusion algorithm has obvious task cost and time consumption. Improvement.

II. Task Scheduling Model

Most cloud computing task scheduling takes the cost and time of task completion as the main factors of scheduling. This is because the quality of the two effects can effectively reflect the effect of cloud computing task scheduling to a certain extent. This article is based on this starting point. , It is divided into establishing functions for cost and time to quantify user needs, and establishing the final fitness function as an evaluation criterion for evaluating cloud computing task scheduling.

The task completion time is set as the sum of the task transmission time and the task execution time. Setting $RunTime(i, j)$ represents the task completion time of the i -th task on the virtual machine j , so the expression is as follows:

$$RunTime(i, j) = T\_trans(i, j) + T\_comp(i, j) \tag{1}$$

$T\_trans(i, j)$ $Ttrans(i, j)$ is the transmission time of the i -th task on the virtual machine j , and $T\_comp(i, j)$ $Tcomp(i, j)$ represents the execution time of the i -th task on the virtual machine j . Let $Time_{finish}$ denote the set maximum time for the user to complete task i . When the task completion time is greater than the set time, the expression of the task completion time function $T\_time(i, j)$ is shown in formula (2).

$$T\_time(i, j) = \frac{1}{Runtime(i, j)} \tag{2}$$

The task completion cost function is defined as the sum of CPU, bandwidth and memory costs consumed by task i on virtual machine j . The expression is as follows:

$$\mathrm{Cos}t(i, j) = C\_mipss(i, j) + C\_bw(i, j) + C\_ram(i, j) \tag{3}$$

In the formula, $C\_mipss(i, j)$ represents the CPU consumed by task i in virtual machine j , $C\_bw(i, j)$

represents the bandwidth consumed by task i in virtual machine j, and $C\_ram(i,j)$ represents the memory consumed by task i in virtual machine j. Cost_{finish} represents the set maximum cost for the user to complete task i, and the cost of task completion must be less than the set maximum cost, so the cost function $T\_\mathrm{cost}(i,j)$ for task completion is shown in formula (4).

$$T\_\mathrm{cost}(i,j)=\frac{1}{\mathrm{Cost}(i,j)} \tag{4}$$

Therefore, this paper comprehensively considers the time function and cost function, and constructs a cloud computing task scheduling function based on time and cost, as shown in formula (5). This paper needs to use meta-heuristic algorithm to find the optimal value of the scheduling function, so as to obtain the optimal task scheduling effect.

$$F(i,j)=\alpha\times T\_time(i,j)+\beta\times T\_\mathrm{cost}(i,j) \tag{5}$$

In the formula, α is the weight value of the time function, β is the weight value of the cost function, and $\alpha+\beta=1$.

III. Fusion Algorithm Based on Particle Swarm and Membrane Computing

3.1 Particle swarm algorithm

Eberhart and Kennedy proposed Particle Swarm Optimizaion (PSO) in 1995 based on the behavior and characteristics of birds in the biological world during the foraging process, which is a stochastic optimization algorithm for intelligent populations. It can effectively obtain the global optimal solution. The algorithm has the advantages of simple implementation, fast convergence, few parameters, and high efficiency. The D-dimensional vector is used to describe the information of particle i, the position is described as $x_i=(x_{i1},x_{i2},\cdots,x_{iD})^T$, and the velocity is described as $v_i=(v_{i1},v_{i2},\cdots,v_{iD})^T$. Therefore, the update equations for velocity and position are:

$$v_{id}^{k+1}=\omega\times v_{id}^{k}+c_1\times r_1\times(pbest_{id}^{k}-x_{id}^{k})+c_2\times r_2\times(gbest_{id}^{k}-x_{id}^{k}) \tag{6}$$

$$x_{id}^{k+1}=x_{id}^{k}+v_{id}^{k+1} \tag{7}$$

3.2 Improvement of particle swarm algorithm

3.2.1 Population initialization

Aiming at the problem that the PSO algorithm lacks population initialization and the accuracy of the algorithm decreases, this paper uses chaotic mapping to initialize the population of particle swarm optimization, and the chaotic mapping is shown in formula (8).

$$x_{n+1}=\begin{cases}x_n/\delta & 0<x_n\le\delta\\ (1-x_n)/\delta & \delta<x_n\le 1\end{cases} \tag{8}$$

In the formula, x_{n+1} and x_n represent the $n+1$ and n respectively, and δ is the chaos parameter.

3.2.2 Domain factor

In order to ensure that the population has better global optimization capabilities, this article introduces the concept of domain factors, which mainly have the influence of attracting other surrounding examples in the iterative process, and the speed of obtaining examples can be better through consideration of these influencing factors. , So as to ensure that the particles have a better solution. The domain factor is set as shown in formula (9).

$$nbest_k = \max | f(k) - \frac{\sum_{i=1}^{n} f(i)}{N} | \tag{9}$$

In the formula, $f(i)$ and $f(k)$ represent the fitness function values of the i and k individuals, and N represents the number of individuals.

3.2.3 Weight factor

In the basic particle swarm algorithm, the weight factor in the particle velocity formula is generally set to a certain fixed value. Obviously, such a setting is unreasonable and does not conform to the state change of the particles in the iterative process. In this paper, the setting of the weight factor and the number of iterations are comprehensively considered, and the range of the weight factor change is set, so that the value of the weight factor changes dynamically as the number of iterations continues to increase, and the setting is shown in formula (10);

$$\omega = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \times \frac{t}{t_{\max}} \tag{10}$$

3.2.4 Non-linear extreme value disturbance

Particles tend to fall into the local optimum in the iterative process, which causes a large number of other particles to gather around the particle, which reduces the individual optimization ability of the entire algorithm. This paper introduces the concept of extreme perturbation factor to effectively ensure that the particle has "vitality", To prevent the algorithm from stagnating after obtaining the optimal solution locally, to promote the particles to search for more unknown intervals, and to improve the search ability. The perturbation operator is set as shown in formula (11).

$$dis = dis_{\max} - \frac{(dis_{\max} - dis_{\min})}{\sum_{iter=1}^{\max} t} (t_{\max} - t) \tag{11}$$

In the formula, $\omega_{\max}$ is the maximum value of the weight; $\omega_{\min}$ is the minimum value of the weight; $t_{\max}$ is the maximum number of iterations; t is the current number of iterations. dis is the perturbation operator, $dis_{\max}$ and $dis_{\min}$ respectively represent the range of the perturbation operator, so the particle velocity update formula is:

$$v_{id}^{k+1} = \omega \times v_{id}^{k} + c_1 \times r_1 \times dis \times (pbest_{id}^{k} - x_{id}^{k}) + c_2 \times r_2 \times (gbest_{id}^{k} - x_{id}^{k}) + c_3 \times r_3 \times (nbest_{id}^{k} - x_{id}^{k}) \tag{12}$$

In the formula, ω is the weighting factor, c_1, c_2 and c_3 are the learning factors, $nbest$ is the domain particle, and dis is the perturbation operator.

3.2.5 Optimization of individual particle position

EDWARDS AM [12] studied the activity characteristics of specific animals and came to a conclusion in line with Levy's flight characteristics, that is, this flight feature can meet the local search in a small range, and it can also meet the global search in a large range, which is effective balancing the relationship between local and global. The distribution density function of Levy's flight step change can be approximately expressed as follows:

$$Levy(s) \sim |s|^{-1-\beta}, 0 < \beta \le 2 \tag{13}$$

$$s = \mu / |v|^{1/\beta} \tag{14}$$

$$\mu \sim N(0, \sigma_\mu^2), v \sim N(0, \sigma_v^2) \tag{15}$$

In the formula, $\sigma_\mu = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \sigma_v = 1$

The particle swarm algorithm and other population intelligence algorithms also have similar Levy flight characteristics, which are easy to fall into the local optimum. Therefore, the Levy flight mechanism is introduced in the foraging individual update behavior, and the formula (16) is as follows:

$$x_{id}^{k+1} = x_{id}^{k} \times a(t) \times sign(rand) \oplus s + v_{id}^{k+1} \tag{16}$$

In the formula, $rand$ is a random number between [-1, 1]. $sign(rand)$ is the Levy flight direction as shown in formula (20), and $a(t)$ is a scale factor set to 1, as shown in formula (17).

$$sign(rand) = \begin{cases} 1 & rand \ge 0 \\ -1 & rand < 0 \end{cases}; -1 \le rand \le 1 \tag{17}$$

3.3 Individual screening based on membrane calculation

Membrane Computing (MC) is a computational model abstracted from the mechanism by which cells process chemical substances. The model has only one main membrane and multiple auxiliary membranes. The main membrane functions to collect the best particles and the local optimization of particle swarm algorithm is performed, and the auxiliary membrane is optimized globally. The specific process is to first decompose the tasks in cloud computing through coding rules, and randomly assign individual particles to different membranes, and then use evolution rules to send the global optimization results obtained in the auxiliary membrane to the main membrane. Continue to iterate until the maximum number of iterations is reached, and finally obtain the optimal particles of the entire membrane system.

3.3.1 Coding rules

In this paper, the structure of membrane calculation is used to randomly allocate particles to the membrane

structure to ensure that the main membrane and auxiliary membrane contain at least one particle (except for the surface membrane), where $q_i(i=1,2,3\cdots n)$ represents the individual particle in the auxiliary membrane, and the dimension of the search space is set For d , the position of the particle $x_i^t=(x_{i1}^t,x_{i2}^t,\cdots x_{id}^t)$ will be regarded as the object processed by the membrane system, and the speed of the particle represents the current state of the individual, so the solution set corresponding to all particles in the population is the object set of the membrane system, in the search space of the entire population Within, the multiple set is denoted as $w_i=(x_{1i},x_{2i},\cdots x_{ni})$, where i represents the number of particles in the membrane, and $x_{1i}^t,x_{2i}^t,\cdots x_{di}^t$ represents the solution corresponding to each particle in the i -th generation in the algorithm iteration in the t -th membrane.

3.3.2 Evolution rules

In order to better improve the performance of the algorithm, find a better optimal solution. In the auxiliary film, initialize the population of the particle swarm algorithm and introduce domain particles to enhance the diversity of the population to obtain a better global optimal solution; at the same time, the particle swarm algorithm is used in the main film to adopt weight factors and nonlinear extreme value disturbances for the particle swarm algorithm Improve the local search performance and convergence speed of the algorithm to obtain the local optimal solution. The main membrane and the auxiliary membrane communicate with each other through information factors, and compare the optimal solutions generated in the two membranes in time, and finally produce the optimal solution of the overall algorithm. The high-quality particles in the auxiliary film are used as information to be transmitted to the main film. The evolutionary rule is that after the particles update their speed and position according to formulas (12) and (16), according to the sorting result of the fitness value from high to low, the fitness the particles with high value are sent to the main membrane for continuous iteration, and the optimal particle individuals of the main membrane are constantly updated. Suppose the dimension is A, the auxiliary membrane is B, and the operation evolution rule from the main membrane to the auxiliary membrane adopts the structure of formula (18) as follows:

$$\begin{aligned} &X_{1L}^{'}\ldots X_{nL}^{'}\ldots X_{dL}^{'}\rightarrow \\ &X_{1L}^{'}\ldots X_{nL}^{'}\ldots X_{dL}^{'}(X_{1L}^{'}\ldots X_{nL}^{'}\ldots X_{dL}^{'}) \end{aligned} \tag{18}$$

3.4 Algorithm steps

Step 1: Initialize the requirements of task scheduling under cloud computing, assign each task under cloud computing to a particle in the particle swarm algorithm, define the relevant parameters required by the particle swarm algorithm and membrane computing, and define the evolution rule of membrane computing, Divide the particles into the membrane structure, ensure that the auxiliary membrane contains one particle, and use formula (5) as the fitness function of the particle swarm algorithm;

Step 2: Optimize the particle swarm algorithm according to formula (8-11), use formula (12) and formula (16) to update the speed and position of particles;

Step 3: According to the rules in the main membrane and auxiliary membrane, initially set the local optimal solution and the global optimal solution of the particles;

Step 4: Sort according to the fitness value of the particles in each auxiliary film, and send the particles with high fitness value in the respective film to the main film;

Step 5: The main membrane reorders the high-quality particles with high fitness values sent from each auxiliary membrane in each iteration, discards the inferior particles to the surface membrane, and updates the particle fitness

value in the main membrane;

Step 6: Determine whether the algorithm has reached the maximum number of iterations, if it reaches the maximum number of iterations, stop the iteration, go to step 7, otherwise go to step 2;

Step 7: Output the optimal particle, that is, the optimal particle corresponds to the optimal cloud task.

IV. Experimental Simulation

In order to further illustrate the efficiency of this algorithm in task scheduling in cloud computing, the algorithm of this paper is expressed as IPSOMC, compared with the PSO algorithm, the IPSO algorithm of literature [4], and the MC algorithm of literature [13] perform cost summation under cloud computing task scheduling. Time comparison. The hardware platform is CPU Core i5, memory is 8GDDR3, hard disk capacity is 1000G, operating system is Window10, and simulation platform is Matlab2012a. The tasks in cloud computing are divided into small tasks and large tasks for comparison. The comparison indicators are mainly time and cost for comparison. Set the inertia weight of PSO to 0.5, c_1, c_2 to 0.5, ω to 0.5, c_1, c_2 in the IPSO algorithm to 1, ω to 1, the auxiliary film in the MC algorithm to be 3, the δ in IPSOMC to be 0.4, $dis_{\max}$ and $dis_{\min}$ are 1 and 0.1 respectively, $\omega_{\max}$ and $\omega_{\min}$ are 1 and 0.1 respectively, c_1, c_2, and c_3 are all set to 0.5, and there are 3 auxiliary membranes. Set the number of small tasks from 100 to 1000, incrementing by 100 tasks each time, and set the number of large tasks from 3000 to 10,000, incrementing by 1000 tasks each time.

4.1 Comparison of four algorithms under small tasks

Figure 1-2 shows the comparison of the time and cost of the four algorithms under small tasks. From the curve in Figure 1, it is found that the IPSOMC algorithm is obviously better than the MC algorithm, IPSO algorithm and PSO algorithm, and compared to the MC algorithm, the average time to complete the IPSO algorithm and the PSO algorithm is reduced by 19.2%, 17.8%, and 18.9% respectively; from Figure 2 we find IPSOMC algorithm is obviously better than MC algorithm, IPSO algorithm and PSO algorithm, and compared with MC algorithm, IPSO algorithm and PSO algorithm save nearly 16.8%, 13.1%, 14.7% on average consumption cost respectively, which shows that IPSOMC algorithm effectively reduces consumption cost. From the comparison of the small tasks, the IPSOMC algorithm can effectively save the completion time and reduce the cost of the task. In the synthesis of Figure 1-2, it is found that there is not much difference between the four algorithms, and the algorithm in this paper has a certain advantage.

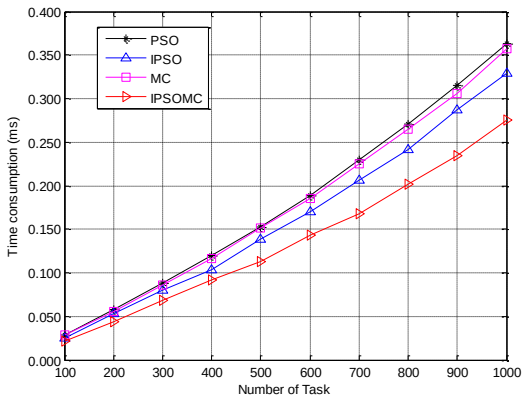

Fig 1: Comparison of completion time under the four algorithms for small tasks

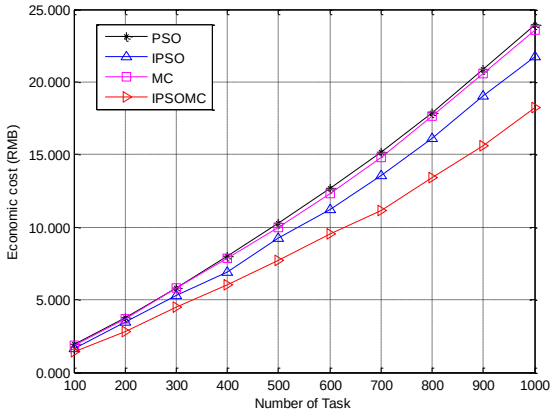

Fig 2: Comparison of consumption costs under the four algorithms for small tasks

4.2 Comparison of four algorithms under large tasks

Figure 3-4 shows the comparison of the cost and time of the four algorithms under the big task. It is found from Figure 3 that the IPSOMC algorithm has obvious advantages. With the gradual increase in the number of tasks, although the curves of the four algorithms all show an upward trend, the curve of the IPSOMC algorithm not only fluctuates the least and rises the slowest, and is compared with the MC algorithm. , The average completion time of the IPSO algorithm and the PSO algorithm is reduced by 33.18%, 22.72%, and 34.1% respectively; from Figure 4, it is found that the curve corresponding to the IPSOMC algorithm is relatively smoother than the corresponding curves of the other three algorithms, and compared to the MC algorithm, the IPSO algorithm The average execution cost of the PSO and PSO algorithms are saved by nearly 31.3%, 20%, and 32%, respectively, which shows that there is indeed an advantage in reducing costs. From the comparison of large tasks, the IPSOMC algorithm can effectively save the completion time and reduce the cost of task consumption, which is very suitable for scheduling with a large number of tasks.

Through the analysis of the above task scheduling, it is obtained that the IPSOMC algorithm has significantly improved local search and global search capabilities compared with the PSO algorithm. The performance improvement effect of the algorithm is very significant, which saves task completion time and reduces consumption costs. The effect of the improvement through the optimization of the IPSOMC algorithm is very obvious, which further shows that the IPSOMC algorithm can effectively adapt to cloud computing task scheduling.

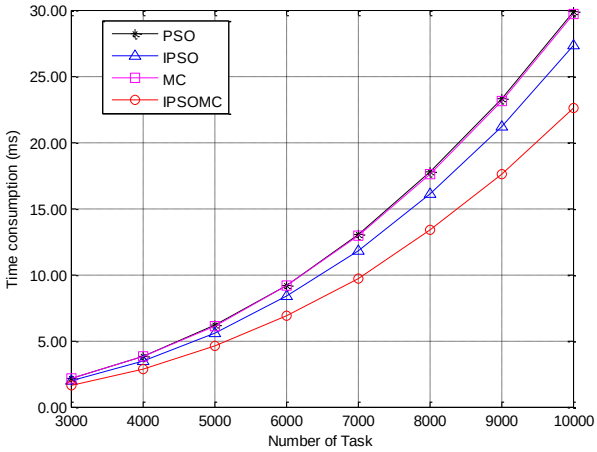

Fig 3: Comparison of completion time under four algorithms for large tasks

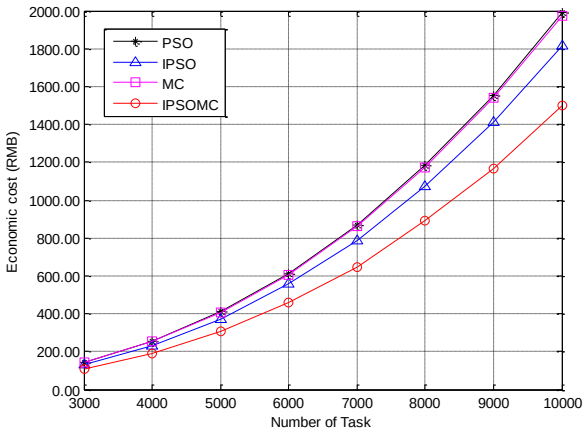

Fig 4: Comparison of consumption costs under the four algorithms for large tasks

V. Conclusion

In view of the high cost and long time spent on cloud computing task scheduling, this paper proposes a cloud computing task scheduling scheme based on the fusion of particle swarm algorithm and membrane computing. In the particle swarm algorithm, the population is initialized, field factors, weight factors and nonlinear extreme value disturbances, Levy optimization and other measures are used to improve the global and local search capabilities. The evolutionary rules of membrane computing are used to obtain the particle swarm algorithm in each iteration. Optimal solution. The simulation experiment results show that the algorithm in this paper has a good effect in terms of completion time and consumption cost as the main scheduling indicators. The next step needs to consider the task scheduling effect of virtual machine load in cloud computing.

References

[1] Singh R M, Paul S, Kumar A. Task scheduling in cloud computing. International Journal of Computer Science and Information Technologies, 2014, 5(6): 7940-7944.

[2] Ibrahim I M. Task scheduling algorithms in cloud computing: A review. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 2021, 12(4): 1041-1053.

[3] Jang S H, Kim T Y, Kim J K, et al. The study of genetic algorithm-based task scheduling for cloud computing. International Journal of Control and Automation, 2012, 5(4): 157-162.

[4] Awad A I, El-Hefnawy N A, Abdel-kader H M. Enhanced particle swarm optimization for task scheduling in cloud computing environments. Procedia Computer Science, 2015, 65: 920-929.

[5] Abdullahi M, Ngadi M A. Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Generation Computer Systems, 2016, 56: 640-650.

[6] Abd Elaziz M, Attiya I. An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing. Artificial Intelligence Review, 2021, 54(5): 3599-3637.

[7] Velliangiri S, Karthikeyan P, Xavier V M A, et al. Hybrid electro search with genetic algorithm for task scheduling in cloud computing. Ain Shams Engineering Journal, 2021, 12(1): 631-639.

[8] Chen X, Cheng L, Liu C, et al. A woa-based optimization approach for task scheduling in cloud computing systems. IEEE Systems journal, 2020, 14(3): 3117-3128.

[9] Jena R K. Task scheduling in cloud environment: A multi-objective ABC framework. Journal of Information and Optimization Sciences, 2017, 38(1): 1-19.

[10] Agarwal M, Srivastava G M S. Genetic algorithm-enabled particle swarm optimization (PSOGA)-based task scheduling in cloud computing environment. International Journal of Information Technology & Decision Making, 2018, 17(04): 1237-1267.

[11] Ju J H, Bao W Z, Wang Z Y, et al. Research for the task scheduling algorithm optimization based on

hybrid PSO and ACO for cloud computing. International Journal of Grid and Distributed Computing, 2014, 7(5): 87-96.

[12] Edwards A M, Phillips R A, Watkins N W, et a1. Revisiting Levy flight search patterns of wandering albatrosses, bumblebees and deer. Nature, 2007, 449: 1044-1048.

[13] Gh. Paun. Computing with membranes. Journal of Computer and System Science, 2000, 61(1):108-143.