International Journal of Multimedia and Ubiquitous Engineering
Vol.12, No.2 (2017), pp.83-94
http://dx.doi.org/10.14257/ijmue.2017.12.2.06

Differential Evolution Cloud Computing Scheduling Strategy Based on Dynamic Adjustment

Li Yi-ran[1] and Zhang Chun-na[2]

[1]*College of Applied Technology, University of Science and Technology Liaoning, Anshan Liaoning 114011, China*
[2]*School of Software, University of Science and Technology Liaoning, Anshan Liaoning 114051, China*
E-mail: lyr7879@163.com

Abstract

This paper presents an improved differential evolution algorithm(CDEI) to solve the problem of resource scheduling and load balancing in clofud computing environment. Firstly, based on the basic differential evolution algorithm, the chaotic strategy is introduced and the individual replacement is implemented at the right time to enhance the diversity of the population; furthermore, the scaling factor and crossover probability are dynamically adjusted in the iteration of the algorithm, and the strategy of early high to low is adopted to overcome the prematurity of the algorithm; finally, the array representation is used in the coding of cloud computing scheduling. The experimental results show that the improved algorithm has the advantages of fast convergence, high precision and low consumption.

Keywords*: cloud computing; chaos strategy; differential evolution; parameter control; convergence*

1. Introduction

Cloud computing is a distributed computing model, which uses virtualization technology to bring together a large number of decentralized resources, users can apply for and obtain them in the appropriate platform [1-3]. The core work of the cloud computing platform is the resource management, and for the user's request to implement reasonable deployment, in which, due to the difference of system nodes and the dynamic demand, it is easy to cause the unbalanced load of the cloud platform, thereby reduce the resource utilization [4]. Therefore, resource scheduling strategy is the core of resource management in cloud computing environment.

Cloud computing resource scheduling is based on the scheduling strategy to implement the dynamic allocation of resources in the limited cluster, that is, according to the user request and take into account the resource load dynamic deployment [5-6]. Obviously, due to its dynamic and complex, it can be attributed to a NP problem, if the scheduling strategy is not properly chosen, the resource load will be unbearable. Therefore, it is necessary to use dynamic intelligent optimization algorithm to solve. At present, the related algorithms include: genetic algorithm, ant colony algorithm, particle swarm optimization algorithm, differential evolution algorithm and so on. For the above algorithms, they have their own advantages. The paper based on the characteristics of cloud computing, to be used differential evolution algorithm. The algorithm is a global stochastic search optimization algorithm, the simple mutation operation and the one-to-one competitive survival strategy based on the difference reduce the complexity of the genetic operation, its operation is simple and the parameters are few, so it is widely used in multi-objective optimization problem [7-8].

ISSN: 1975-0080 IJMUE

In order to provide efficient support for the cloud platform, this paper proposes a improved differential evolution algorithm with chaos. The algorithm uses the array manner to re-encode the task-resource in the scheduling strategy. In addition, the chaotic strategy is introduced to implement the replacement operation for the oscillating individuals, to disturb the population, and to adjust the scaling factor and crossover probability from time to time, to be enhanced the flexibility of control parameters in the algorithm. In this way, local perturbation, global search and the performance of the algorithm have been greatly improved, while the consumption of the platform is reduced correspondingly.

2. Cloud Computing Resource Scheduling Theory

Processing methods of cloud computing is parallel and distributed, where the task and resources are not simple relationship of one-to-one [9]. First of all need to complete the task - resource mapping, and then complete the resource - node mapping, that is, the task scheduling problem is described as: n tasks are reasonably allocated to m resources by the scheduling algorithm, and to deal with, to ensure resource load balancing.

Currently, the cloud computing environment mainly uses the Map/Reduce model by Google. In this paper, the sub-tasks are independent of each other. The resource model is defined as follows: using a five-tuple to describe, $F = \{T, R, E, f, p\}$, in the formula, f is the objective function, p is the scheduling algorithm, T is the total task. The decomposed subtasks can be expressed as $T = \{t_1, t_2, \cdots t_n\}$, the resource R can be expressed as $R = \{r_1, r_2, \cdots r_m\}$, the physical device E can be expressed as $E = \{e_1, e_2, \cdots e_m\}$, here, the mapping relationship $T \to R$ is controlled by cloud computing center, the mapping relationship $R \to E$ need to be assigned to the physical device by the scheduler. Thus, the scheduling process of a task evolves into the mapping process $T \to R \to E$. Based on this, the resource scheduling problem in the cloud computing environment is transformed into the solving process of TR matrix. tr as all feasible solution, the set of TR matrix can be expressed as:

$$TR = \begin{bmatrix} tr_{1,1} & tr_{1,2} & \cdots & tr_{1,n} \\ tr_{2,1} & & & \\ \vdots & & \ddots & \\ tr_{m,1} & & & tr_{m,n} \end{bmatrix} \tag{1}$$

In the formula, $tr_{i,j}$ is the load relationship between the load t_i and the virtual resource r_k, meet $\{tr \mid tr \in [0,1], \sum_{i=1}^{m} tr_{i,j} = 1\}$.

According to the relationship between t_i and r_i, the consuming time of t_i execute on e_i is $TE(t_i, e_i)$, and the corresponding matrix can be expressed as:

$$TE = \begin{bmatrix} te_{1,1} & te_{1,2} & \cdots & te_{1,n} \\ te_{2,1} & & & \\ \vdots & & \ddots & \\ te_{m,1} & & & te_{m,n} \end{bmatrix} \tag{2}$$

In the formula, $te_{i,j}$ is the time for the i-th load is executed on the j-th physical device. The execution load of a physical device e_j is t_i, the earliest start time of remove the waiting time is w_j, then the total execution time of the load t_i executed on e_j can be defined as:

$$S(t_i) = w_j + TE(t_i, e_i) \tag{3}$$

For all the loads $\{t_1, t_2, \cdots, t_n\}$, the total execution time can be defined as:

$$S'(t) = \sum_{i=1}^{n} S(t_i) \tag{4}$$

This paper focuses on the time, so the load consumption time as the objective function should meet to achieve a minimum, defined as follows:

$$F(t) = \min \sum_{i=1}^{n} S(t_i) \tag{5}$$

3. Differential Evolution Algorithm with Chaos

3.1. The Basic Principle of Differential Evolution Algorithm

The differential evolution algorithm is a kind of optimization algorithm with real vector coding in continuous space and can complete parallel and random search, which has the characteristics of simple and less controlled parameter [10-14]. The basic idea of DE is similar to that of genetic algorithm, that is, using mutation operation to generate new individuals, and then implementing cross and selection operations, through constant iterative evolution to search the global optimal solution. DE implements the mutation operation by the difference strategy, which is different from the genetic algorithm, and enhances the searching ability of the algorithm by using the population characteristic [15]. The solution of nonlinear minimization problem is a classic example in the optimization problem, the basic mathematical model is as follows:

$$\begin{cases} \min f(x_1, x_2, \cdots x_Q) \\ x_j^L \le x_j \le x_j^U, j = 1, 2, \cdots, Q \end{cases} \tag{6}$$

In the formula, Q is the dimension of solution space, U、L are the upper and lower limits of the component x_j.

The basic steps of differential evolution algorithm are as follows:

(1) Population initialization

Setting the population size is N_p, then the k-th generation individual i can be expressed as $\{x_{j,i} \mid x_{j,i}^L \le x_{j,i}^L x_{j,i} \le x_{j,i}^U, i = 1, 2, \cdots, N_p;\ j = 1, 2, \cdots, Q\}$, Each generation in the algorithm is composed of N_p vector with dimension Q, the primary population $x_{j,i}(0)$ can be expressed as:

$$x_{j,i}(0) = x_{j,i}^L + rand(0,1) \cdot (x_{j,i}^U - x_{j,i}^L) \tag{7}$$

In the formula, $rand(0,1)$ is a random number, $\in (0,1)$, uniform distribution.

(2) Mutation operation

For each target individual $x_{j,i}$ in the population, DE generally adopts the difference strategy to carry out mutation operation, that is, randomly select two individuals in the population to vector fusion with the individual to be mutated, and then generate new variant individual $V_{j,i}(k+1)$.

$$V_{j,i}(k+1) = x_{r1}(k) + F \cdot (x_{r2}(k) - x_{r3}(k)) \tag{8}$$

In the formula, $V_{j,i}(k+1)$ is new variant individual; F is the scaling factor, $\in (0,1)$; r_1、r_2、$r_3 \in [1, N_p]$ is used to control the vector difference of randomly selected individuals and is also different from the individual to be mutated, at the same time, $r_1 \ne r_2 \ne r_3 \ne j$.

In the iteration of the algorithm, both the randomly selected individuals and the newly generated individuals must be effective and meet the established boundary conditions, that is, $\{V_{j,i}(k+1) \mid x_{j,i}^{L} \le V_{j,i}(k+1) \le x_{j,i}^{U}\}$.

(3) Cross operation

The individual to be measured $V_{j,i}^{'}(k+1)$ is obtained by the cross transform of the individual, is corresponding to $V_{j,i}^{'}(k+1)=\{V_{1,i}^{'}(k+1),V_{2,i}^{'}(k+1),\cdots,V_{Q,i}^{'}(k+1)\}$, when the intermediate is crossed transform, it is possible the tested individual contains at least one vector is generated from $x_{j,i}$ by the random strategy, the cross should follow the following formula:

$$V_{j,i}^{'}(k+1)=\begin{cases}V_{j,i}(k+1) & rand(j) \le CR \text{ or } j \in [1,Q] \\ x_{j,i}(k)\end{cases} \tag{9}$$

In the formula, j is a limited random number, to ensure that the individual to be measured will not be completely separated from the target $x_{j,i}$; $CR \in [0,1]$, is called the crossover probability, used to adjust the difference of the new generation individual and the original individual, as a random number.

(4) Select operation

The fitness values of the tested individuals and the target individuals were compared, and those who retained the smaller ones became the new individuals of the next generation. Greedy algorithm is used to determine the new individual:

$$x_{j,i}(k+1)=\begin{cases}V_{j,i}^{'}(k+1) & f(V_{j,i}^{'}(k+1)) \le f(x_{j,i}(k)) \\ x_{j,i}(k) & f(V_{j,i}^{'}(k+1)) > f(x_{j,i}(k))\end{cases} \tag{10}$$

3.2. Chaos Strategy

Chaos is a nonlinear phenomenon, which exists widely in nature. It has the characteristics of randomness, ergodicity and regularity, and can search all the states in the limited region according to its own law [16]. In the evolution of algorithms, with the extreme value emerging, the diversity of the population decreased, the individual is easily fall into the local optimal constraint. At this time, chaotic sequences can be used to disturb and to replace these stagnant individuals with a certain probability. In this way, the active individuals continue to complete the evolution, while the individual into the oscillation can be replaced, which enhances the diversity of the algorithm. In order to optimize the search by using the property of chaos, the search procedure is as follows:

(1) Define an initial region, set N dimensional initial vector $R_0=(R_{01},R_{02},\cdots,R_{0N})$, the values in R_0 are adjacent to each other, and the difference is very small.

(2)The initial vector R_0 is calculated by using the logistics equation, and chaotic sequence $c_1,c_2,\cdots,c_n$ is generated. Here, after several iterations, the system will be completely in a chaotic state. The vector layer can be expressed as:

$$c_{i+1}=c_i(1-c_{i-1})\lambda \tag{11}$$

In the formula, λ is an iterative control parameter.

(3) Set the initial vector $c_{j,i}(0)$, then produce:

$$c_{j,i+1}(0)=c_{j,i}(0)\cdot(1-c_{j,i}(0))\cdot\lambda_{j,i}(0) \tag{12}$$

In the formula, $i \in [1,N_p]$.

(4) setting the individual $x_i(0)$ in the initial population, after the chaotic sequence mapping, the j-th component $x_{j,i}(0)$ can be expressed as:

$$x_{j,i}(0) = r \cdot rand(j) \cdot c_{j,i}(0) + x_{j,i} \tag{13}$$

In the formula, r is the activity radius of individual $x_{j,i}$, $rand(j) \in [-1,1]$.

Population initialization, the probability that the i-th individual is replaced is ρ, the formula is:

$$\rho = \frac{i}{N_p} \tag{14}$$

Here, set a threshold δ to measure the diversity of the population. When the diversity of the population reaches it, the replacement operation may be implemented. Diversity control parameters are defined as follows:

$$\delta = \frac{1}{N_p} \sum_{j=1}^{Q} \frac{\sum_{i=1}^{N_p} | x_{j,i}(k) - \frac{1}{N_p} \sum_{i=1}^{N_p} x_{j,i}(k) |}{x_j^U - x_j^L} \tag{15}$$

In the formula, $\delta \in [0,1]$.

3.3. Parameter Optimization of DE Algorithm

The DE algorithm includes three important parameters: population size N_p, scaling factor F and crossover probability CR. According to experience, the population size should be moderate, too much time will be spent in the calculation when it is too large, and the diversity of the population will be reduced and the convergence can't be guaranteed when it is too small, it usually is 5-10 times of the dimension, that is $N_p \in [5Q, 10Q]$; the scaling factor is used to control the scaling size of the difference vector, $F \in [0.5,1]$; the crossover probability is used to adjust the diversity of the population. It should be noted that the crossover probability is too large to make the algorithm into a random algorithm, the allowed range is $CR \in [0,1]$.

The reasonable setting of F and CR can effectively adjust the convergence of the algorithm. Because the DE algorithm has the characteristics of easy precocity, it is necessary to strengthen the global search and improve the diversity of the population in the early stage of the algorithm, while local control should be strengthened to improve the accuracy of the algorithm in the latter. In this way, the value of F and CR reflects slightly higher pre-set, and the latter need to set down, the following are the improved parameters:

$$\begin{cases} F(k+1) = F(k) - \dfrac{F(0) - F_{\max}}{k_{\max}} \\ CR(k+1) = CR(k) - \dfrac{CR(0) - CR_{\min}}{k_{\max}} \end{cases} \tag{16}$$

In the formula, $F(0)$ and $CR(0)$ are scaling factor and crossover probability of the first generation; $F_{\max}$ and $CR_{\min}$ are the maximum value the scaling factor and the minimum value of the crossover probability in the iteration, $k_{\max}$ is the maximum number of iteration.

4. Cloud Environment Scheduling Strategy

4.1. Coding Rule

The task of cloud computing resource scheduling is to assign n tasks to m resources, the goal of the algorithm is to minimize the total task-time. The traditional scheduling algorithm uses a special character segmentation approach to complete, such as the mapping $T \to R$, coding is (0,3, -1,2, -1,6,5), task 0,3 is mapped to resource 0; task 2 is mapped to resource 1; task 6,5 is mapped to resource 2, in this way, the each other corresponding distribution mode more clearly, the deployment of resources is also relatively easy. But for DE, it will produce some problems, due to the mutation operator in the algorithm will inevitably repeat the individual, if this coding is used, it will obtain the unnecessary repeated solution, which results in the lack of the optimal solution. Based on this, this paper designs an array representation, which is to set up a two-dimensional array $A[m][n]$, the second subscript identifies the resource number, the array value corresponds to the task number. Such as $A[0][0]=1$, $A[1][0]=3$, respectively identifies task 0,3 is mapped to resource 0.

4.2. Algorithm Step

Step 1: Determining population size N_p, set the maximum number of iteration k, scaling factor F and crossover probability CR;

Step 2: Population initialization, the use of chaotic sequences to implement individual mapping;

Step 3: To perform a mutation operation produce a new $V_{j,i}(k+1)$;

Step 4: To perform a cross operation produce a individual to be tested $V'_{j,i}(k+1)$;

Step 5: The selection operation is performed to obtain a new individual;

Step 6: The diversity of the population in each generation is calculated by using the chaotic strategy, to determine the threshold value and whether to implement individual replacement;

Step 7: Determines whether the current iteration is out of bounds, and if not, return to step 3, otherwise the algorithm terminates.

5. Experimental Analysis

For the improved algorithm CDEI, this paper uses the CloudSim platform to simulate. The population size is 50 and the maximum number of iteration is 800 in the comparative analysis. $F(0)=0.75$, $CR(0)=0.8$. The performance of the algorithm and the performance of the platform optimization are investigated in the experiment. The former mainly analyzes the ratio of the convergence of the algorithm to the number of iteration; the latter mainly investigates the ratio of task-to-time in the platform. First, the iterative performance in the optimized algorithm is analyzed, and selecting the traditional differential evolution algorithm (DE) and the differential evolution algorithm with chaos (CDE) are compared with CDEI. Four benchmark functions are used to complete the test, taking into account the randomness of the algorithm, the three algorithms are run 20 times to get the mean value, which is the number of iteration compared to DE and CDEI, the convergence is the comparison of DE, CDE and CDEI. The following is a description of the benchmark function and an iteration time ratio legend:

(1) The function formula of $Schwefel$ is as follows:

$$Schwefel(x) = \sum_{i=1}^{D} (x_i \sin\sqrt{|x_i|})$$

(2) The function formula of $Rosenbrock$ is as follows:

$$Rosenbrock(x) = \sum_{i=1}^{D} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$$

(3) The function formula of *Noncontinuous Rastrigin* is as follows:

$$Noncontinuous\ Rastrigin = \sum_{i=1}^{D} [x_i^2 - 10\cos 2\pi x_i + 10]$$

$$x_i = \begin{cases} k_i & |k_i| < 0.5 \\ round(2k_i)/2 & |k_i| \geq 0.5 \end{cases}$$

(4) The function formula of *Griewank* is as follows:

$$Griewank = \frac{1}{4000}\sum_{i=1}^{D} (x_i)^2 - \prod_{i=1}^{D} \cos(x_i / \sqrt{i}) + 1$$

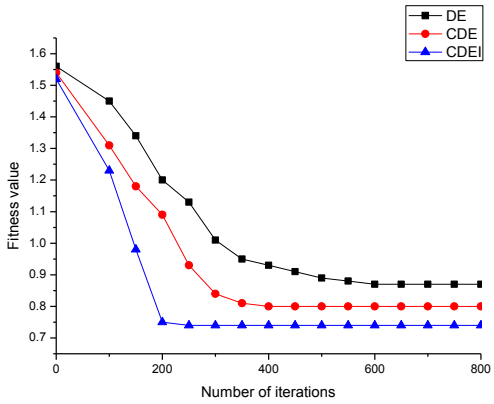
Table 1. Parameter of Benchmark Function

benchmark function name	dimension	Range	optimal value
Schwefel	30	$[-10,10]^D$	−12569.5
Rosenbrock	30	$[-10,10]^D$	0
Noncontinuous Rastrigin	30	$[-5.12,5.12]^D$	0
Griewank	30	$[-600,600]^D$	0

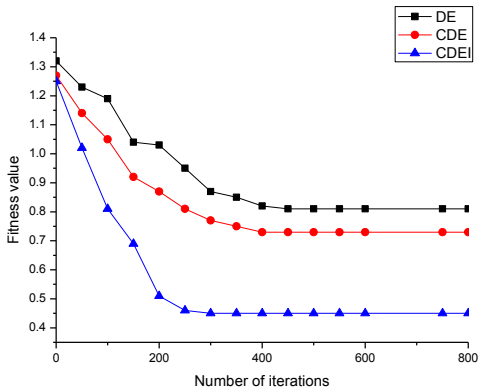
Table 2. The Ratio of Iteration Times

benchmark function	Index type	DE	CDEI
Schwefel	Max	605	236
	Min	431	253
	Mean	547	241
Rosenbrock	Max	482	302
	Min	313	223
	Mean	424	249
Noncontinuous Rastrigin	Max	589	290
	Min	461	231
	Mean	527	256
Griewank	Max	712	326
	Min	406	237
	Mean	539	271

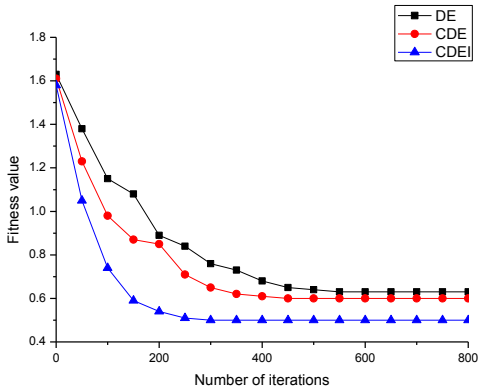
Table 2 can be found that the convergence of CDEI compared to DE has been greatly improved by the comparison, the maximum value, minimum value and mean value in the ratio of the iteration times have certain advantages, and the difference between the three is smaller. It can be seen that the diversity of the population is supplemented in the later stage of the algorithm, and it will not produce too much jump in the algorithm iteration, the individual distribution is more uniform, and the convergence speed is accelerated accordingly.



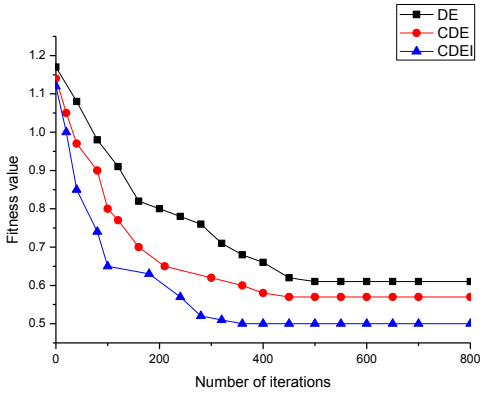
(a) Comparison Result of *Schwefel* Convergence



(b) Comparison Result of *Rosenbrock* Convergence



(c) Comparison Result of *Noncontinuous Rastrigin* Convergence

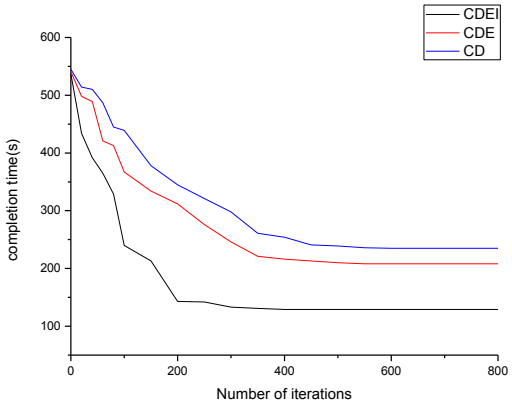


(d) Comparison Result of *Griewank* Convergence

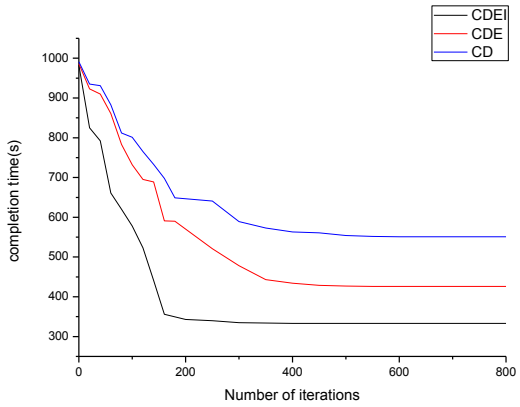
Figure 1. Convergence Comparison of Benchmark Function

The comparison of the three algorithms about the convergence in Figure 1, CDEI has certain advantages, especially for the non-optimized DE algorithm, but the gap between CDEI and CDE is not so bright as before, mainly the traditional DE algorithm is a combination of the chaotic strategy, the diversity of the population is guaranteed, eliminating the problem of individual oscillation. In this paper, F、CR are dynamically adjusted, and the parameters are adjusted so that the individual can't fall into the local optimal constraint in the algorithm, which accelerates the convergence.

In order to investigate the task-time ratio of the platform, two schemes were set up: (1) the number of task is 400, the number of resource is 10; (2) the number of task is 800, the number of resource is 10; the compare the legend as follows:



(a) The Contrast when the Number of Task is 400



(b) The Contrast when the Number of Task is 400

Figure 2. The Ratio of Task-Time in Platform

It can be seen from Figure 2 that the improved algorithm CDEI proposed in this paper is obviously lower than the other two algorithms in terms of time consumption. From the data we can see that with the iteration of the algorithm, the value of CDEI has been rising steadily, and there is little oscillation. On the contrary, the DE algorithm will be a relatively large amplitude oscillation, which is mainly the early maturity of the algorithm, resulting in rapid convergence phenomenon. The set value of the scaling factor and the cross probability are constant, which leads to the decline of the late search ability and can't quickly get the optimal solution. It can be seen that the improvement effect of the algorithm is obvious and the resource deployment is reasonable.

6. Conclusion

Based on the chaos theory, an optimal differential evolution algorithm(CDEI) is proposed on the basis of the traditional differential evolution algorithm, which is used to solve the resource scheduling problem in cloud computing. In the early stage, chaos strategy was introduced to generate a new individual in the differential evolution, which was used to stagnate individual replacement operation to enhance the diversity of the population; at the same time, the scaling factor and crossover probability are dynamically adjusted, and the principle of high to low is adopted to improve the convergence of the algorithm. The experimental results show that the optimized algorithm has a good ability of resource scheduling.

References

[1] S. O. David, A. Amit and K. Yogesh, "A nucleic filter to enhance the security in cloud computing environment", Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development, New Delhi, India, **(2016)**, pp. 3762-3765.
[2] G. Shivam and C. M. Subhas, "Compliance, network, security and the people related factors in cloud ERP implementation", International Journal of Communication Systems, vol. 29, no. 8, **(2016)**, pp. 1395-1419.
[3] N. Farrukh and Q. Rizwan, "An Early Evaluation and Comparison of Three Private Cloud Computing Software Platforms", Journal of Computer Science and Technology, vol. 30, no. 3, **(2015)**, pp.639-654.
[4] J. A. Mohammed and M. Sharon, "Evaluating metrics performance of a dynamic scaling algorithm in cloud computing", Proceedings of the 30th International Conference on Computers and Their Applications, Honolulu, HI, USA, **(2015)**, pp.175-181.
[5] S. D. Bum, J. Y. Boo, L. S. Hee and L. K. Ho, "Cloud computing for ubiquitous computing on M2M and IoT environment mobile application", Cluster Computing, vol. 19, no. 2, **(2016)**, pp. 1001-1013.
[6] M. Preeti, S. P. Emmanuel, V. Vijay and T. Udaya, "Intrusion detection techniques in cloud environment: A survey", Journal of Network and Computer Applications, vol. 77, **(2016)**, pp. 18-47.

[7] D. Falco, I. S. Umberto and T. Ernesto, "Mapping of time-consuming multitask applications on a cloud system by multi-objective Differential Evolution", Parallel Computing, vol. 48, **(2015)**, pp. 40-58.
[8] U. Roberto and C. Stefano, "Differential evolution based human body pose estimation from point clouds", GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference, Amsterdam, Netherlands, **(2013)**, pp. 1389-1396.
[9] F. Nuno, P. J. Henrique, D. Costa and J. Fontes, "The adoption of cloud computing services by Portuguese companies: The impact of marketing efforts", Revista Iberica de Sistemas e Tecnologias de Informacao, vol.18, **(2016)**, pp. 33-48.
[10] K. Guney and D. Karaboga, "New narrow aperture dimension expressions obtained by using a differential evolution algorithm for optimum gain pyramidal horns", Journal of Electromagnetic Waves and Applications, vol. 18, no. 3, **(2004)**, pp. 321-339.
[11] W. A. Augusteen, R. Kumari and R. Rengaraj, "Economic and various emission dispatch using differential evolution algorithm", 2016 3rd International Conference on Electrical Energy Systems, Chennai, India, **(2016)**, pp.74-78.
[12] B. E. Moreira, B. A. Moreira, S. Pérez, J. Manuel, G. Pulido, J. Antonio, V. Rodríguez and M. Angel, "A hybrid differential evolution algorithm for solving the terminal assignment problem", Lecture Notes in Computer Science, vol. 5518, no. 2, **(2009)**, pp. 179-186.
[13] Z. D. Liu, H. G. Liqun and L. Steven, "An improved differential evolution algorithm for the task assignment problem", Engineering Applications of Artificial Intelligence, vol. 24, no. 4, **(2010)**, pp. 616-624.
[14] A. R. Khaparde, M. M. Raghuwanshi, Liqun and L. G. Malik, "A new distributed differential evolution algorithm", International Conference on Computing, Communication and Automation, Greater Noida, India, **(2015)**, pp. 558-562.
[15] M. D. Asafuddoula, R. Tapabrata and S. Ruhul, "An adaptive hybrid differential evolution algorithm for single objective optimization", Applied Mathematics and Computation, vol. 231, no. 4, **(2014)**, pp.601-618.
[16] S. Mukhopadhyay and S. Banerjee, "Global optimization of an optical chaotic system by Chaotic Multi Swarm Particle Swarm Optimization", Expert Systems with Applications, vol. 39, no. 1, **(2012)**, pp. 917-924.

Authors

Y. R. Li, received the Master's degree in computer application technology from University of Science and Technology Liaoning, in 2008. Currently, he is a lecturer at School of applied technology college at University of Science and Technology Liaoning. His research interests include Distributed computing and data mining.

C. N. Zhang, received the Master's degree in computer application technology from University of Science and Technology Liaoning, in 2007. Currently, she is a lecturer at School of Software Engineering at University of Science and Technology Liaoning. Her research interests include Distributed computing and data mining.