Research on edge cloud load balancing strategy based on chaotic hierarchical gene Replication

Leilei Zhu [1], Zhao Ke [2], Zhichen Wu [3], Dan Liu [4], Wei Su [5] and Li Li*[6]

[1] College of Computer Science and Technology, Changchun University of Science and Technology, Jilin, Changchun 130022, China
E-mail: zhull01@ccucm.edu.cn
[2] College of Computer Science and Technology, Changchun University of Science and Technology, Jilin, Changchun 130022, China
E-mail: zk2611zhao@qq.com
[3] College of Computer Science and Technology, Changchun University of Science and Technology, Jilin, Changchun 130022, China
E-mail: 1258484050@qq.com
[4] College of Computer Science and Technology, Changchun University of Science and Technology, Jilin, Changchun 130022, China
E-mail: ld@cust.edu.cn
[5] College of Medical Information, Changchun University of Chinese Medicine, Jilin, Changchun 130117, China
E-mail: suwei@ccucm.edu.cn
[5] College of Computer Science and Technology, Changchun University of Science and Technology, Jilin, Changchun 130022, China
E-mail: ll@cust.edu.cn

Abstract. Edge cloud is usually used to dispose delay-sensitive business、 realize the processing and analysis of local real-time and short-cycle data. However, due to the large number of concurrent requests for edge intensive tasks, the resource allocation strategy will seriously affect the stability of nodes. To solve this problem, an adaptive resource allocation model (CRPSO model) based on chaotic hierarchical gene replication is proposed in this paper. In the model, the concept of chaotic replication ratio is proposed. Based on Kubernetes edge cluster, the resource allocation results of CRPSO model are verified from three aspects: CPU variance, memory variance and total variance of two kinds of resources. Experiments shows that the fitness of this model is much higher than that of the comparison algorithm on average, and the convergence rate of this model remains optimal even though the solution space is set to be exponential. In addition, the variance fluctuation range of CPU and memory is lower than that of Kubernetes clustering algorithm after resource allocation. Therefore, this model is suitable for edge large-scale task request scenario.

Keywords: Edge cloud, Resource allocation, Chaos theory, replication ratio, Kubernetes

1. INTRODUCTION

Due to the terminal application of the Internet of things is generally near the user. Therefore, the traditional cloud computing paradigm no longer meets the requirements of users [1,2]. Hence, the concept of edge cloud is born. Edge cloud is an extension of cloud computing. It can reduce processing time of business and meet the requirements of field computing. Base on the edge cloud, the local delay sensitive tasks can be implemented and processed, so as to form a distributed cloud architecture with the central cloud [3]. Fig. 1 shows the framework structure of edge cloud collaboration. Fig. 2 shows the architecture of the edge cloud.

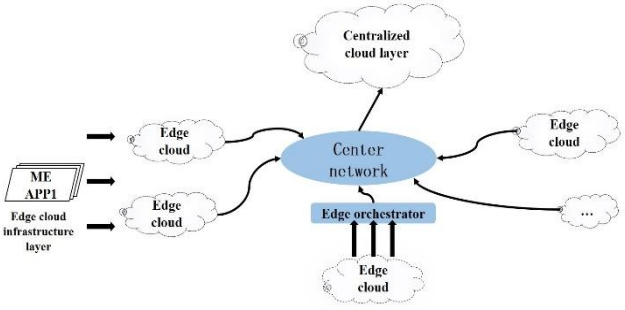

Fig.1 The framework of edge cloud collaboration

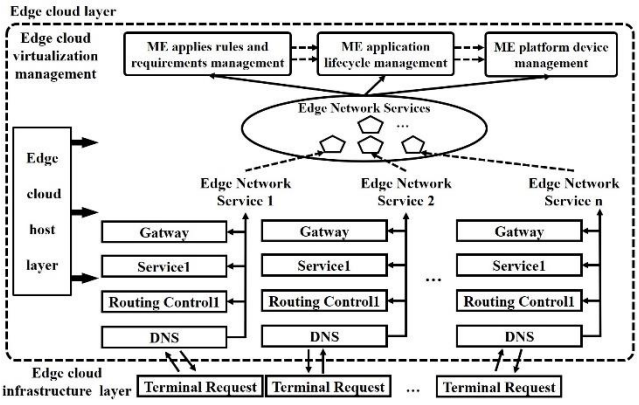

Fig. 2 The architecture of the edge cloud

However, since the edge cloud computing capability is relatively limited and it usually handles high concurrency delay-sensitive tasks, the unbalanced resource allocation strategy will cause node instability. Some studies have attempted to address these issues, but they are still based on the cloud computing paradigm. Reference [4] proposed a relaxed ant colony algorithm to reduce the consumption of task scheduling. Reference [5] proposed the mapping allocation strategy between tasks and virtual machines and optimized the mapping problem between them. Reference [6] referred to the paradigm of cloud computing, proposed a multi-task scheduling algorithm based on delay to improve the requirements of mobile

cloud computing. Reference [7] proposed a task scheduling method based on the improved chaotic bat algorithm to reduce the terminal energy consumption.

To sum up, the above research is still the research of cloud computing mode, which solves the problems of edge devices or the problem of the cloud center. without unloading the task to the edge server and considering the optimization of edge cluster. In this paper, a balanced resource scheduling policy is established under edge cluster. The main contributions are as follows.

(1) Established an adaptive resource scheduling model (CRPSO model) based on chaotic gene replication, and propose the concept of gene replication ratio.

(2) Theoretical verification of CRPSO model based on pattern theory.

(3) The multi-dimensional comparison experiment shows that the CRPSO model proposed in this paper can optimize the load balancing capability of edge nodes and maintain the stable long-term service operation of the cluster.

2. RELATED WORK

At present, some studies have considered the edge cloud environment. Reference [8] designed a task scheduling scheme based on game theory to reduce the energy consumption and computing cost of the central cloud. Reference [9] proposed a three-stage task completion time minimization model (FC-SDES model), and considered the total time of task placement and task scheduling, and finally studied the maximum task completion time in the FC-SDES model. Reference [10] converted the computing resources of edge servers according to the remaining completion time of tasks, so as to shorten the completion time of tasks and improve the unloading efficiency of the task. Reference [11] proposed an optimal resource management strategy to minimize the energy consumption of mobile devices in order to control the proportion of tasks beyond the delay constraint, but this model did not consider the computing resource constraints of edge devices. Reference [12] based on the improved grey Wolf optimization algorithm and its excellent global search ability, the problem of task scheduling efficiency and local optimal solution are solved. To sum up, the above studies have considered the interaction between the edge cloud nodes, but only the delay problem of the edge cloud environment is considered, and the load balancing and green energy saving problems of the edge cloud are not considered. The load balancing of the cluster can ensure the long-term stability and computing efficiency of the edge nodes. This paper considers the edge cloud multi-objective resource optimization method, based on the edge cloud characteristics of short delay, to achieve green resource scheduling.

3. CRPSO MODEL OF RESOURCES ALLOCATION

The experiment in this paper refers to the initial constraints of cloud computing task scheduling model. Considering a set of cloud tasks is C, $C=\{c_1,c_2,\cdots,c_n\}$. Given a set of compute nodes is N, $N=\{v_1,v_2,\cdots,v_m\}$. The distribution relationship between cloud tasks and nodes is shown in formula (1).

$$D=\begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix} \tag{1}$$

Each element d_{ij} of matrix D represents whether a cloud task c_i is placed on the node v_j with 0 or 1, and the constraint conditions such as formula (2) are obtained.

$$d_{ij}=\begin{cases} 1 & c_i \text{ is placed on the } v_j \\ 0 & c_i \text{ is not placed on the } v_j \end{cases} \tag{2}$$

Multiple computing resources in a node, denoted by r_k , v_j has an upper limit of L_{jk}. The computing resource requirement for each cloud task is c_{ik}. According to the resource constraints of the compute node, assuming that the cloud task c_i is placed on the v_j . the constraint conditions of formula (3) can be obtained.

$$\sum_{i=1}^{N} d_{ijc_{ik}} \le L_{jk} \tag{3}$$

The current utilization of r_k is u_{jk}. The utilization rate is calculated as shown in formula (4).

$$u_{jk} = \frac{\sum_{i=1} c_{ik}}{L_{jk}} \tag{4}$$

In this paper, standard deviation is used to measure the dispersion degree of a variable in the sample, that is, the standard deviation is minimized. Then the current load balancing degree B of the cluster is defined in formula (5).

$$B=\sum_{k=1}\left(\sum_{j=1}^{m}\sqrt{\frac{\left(u_{jk}-\bar{u}_k\right)^2}{m-1}}\right) \tag{5}$$

In formula (5), $\bar{u}_k$ indicates the average utilization of r_k in the cluster. The implication of this formula is that the sum of standard deviations of different resources in the cluster is the smallest. Indicator B is effective because it is minimized only if the standard deviation of utilization of all resources is at a small level. Then, the objective function is transformed into formula (6).

$$\min(B)=\min\left(\sum_{k=1}\left(\sum_{j=1}^{m}\sqrt{\frac{\left(u_{jk}-\bar{u}_k\right)^2}{m-1}}\right)\right) \tag{6}$$

Now, the description of the problem is changed to: find a distribution scheme D to minimize the load balancing degree B of the whole cluster. Its formulas are described as formula (7) -formula (8).

$$min\big(f(D)\big) = \underset{D}{\operatorname{argmin}}(B) = min\left(\sum_{k=1}\left(\sum_{j=1}^{m}\sqrt{\frac{\left(u_{jk}-\bar{u}_k\right)^2}{m-1}}\right)\right) \tag{7}$$

$$\text{s.t.}\begin{cases} \sum_{i=1}^{N} d_{ij}c_{ik} \le L_{jk} \\ \sum_{j=1}^{M} d_{ij}=1 \\ d_{ij}=\begin{cases} 1 & c_i \text{ is placed on the } v_j \\ 0 & c_i \text{ is not placed on the } v_j \end{cases} \end{cases} \tag{8}$$

In the traditional particle swarm optimization (PSO)

algorithm [13]-[14], the individual particle updates its position and velocity according to the global optimal particle in the current population and the historical optimal position of the current particle. After several iterations, the global optimal solution of the problem is obtained. The speed update and the position update formula are as shown in formula (9)- formula (10).

$$v(t+1)=\omega v(t)+c_1 r_1\left(pbest\text{-}x(t)\right)+c_2 r_2\left(gbest\text{-}x(t)\right) \quad (9)$$

$$x(t+1)=x(t)+v(t+1) \quad (10)$$

Here, *gbest* is the global optimal particle, *pbest* is the iterative optimal particle, and ω is the inertia weight, c_1 is the social acceleration factor, and c_2 is the cognitive acceleration factor. v(t+1) is the velocity of the particle at the next moment, and x(t+1) is the position of the particle at the next moment.

Based on discrete particle swarm optimization (DPSO model) [15]-[16], CRPSO model is proposed in this paper.

The extreme sensitivity of chaos indicates that a small deviation of the initial conditions of the nonlinear iterative process can lead to huge fluctuations in the iterative process, and a short number of iterations can produce a huge difference in the value of the expression [17]. The definition is: $f:R^n \rightarrow R^n$ is a self map on the set R^n, Here, n is the number of cloud tasks, marking the size of the solution space of the load balancing problem. If the $f(R^n) \subset E^n$, the subset E^n is the invariant set of E^n. Obviously, the orbital of x falls intR^no. $x \subset E^n$. For any two points: x_1 and x_2 ,they are in the set E^n, the distance between the two points satisfies that $d(x_1, x_2) < \delta$, there is always a positive number exists, assuming it is d_0. The limit of $d(x_1, x_2)$ is as shown in formula (11).

$$\lim_{n\to\infty} d\left(f^n(x_1), f^n(x_2)\right) \geq d_0 \quad (11)$$

This indicates that any neighborhood of any point in the iteration space can always reach the neighborhood of other point after sufficient iterations. In this model, a series of new solutions which are far away from the current solution vector can be quickly obtained by chaotic iteration, and these new solutions are widely distributed.

From the point of view of particle swarm optimization algorithm, by generating Logistics chaotic sequence and selecting new solutions in it, the search for global optimal solution can be carried out in a larger scope while taking into account the current search area. In the practical application of the algorithm, in the process of each iteration, a certain length of chaotic sequence is generated according to the optimal solution vector for further search, so as to generate new solutions to optimize the search process of the solution space.

According to formula (11), an iteration sequence can be generated, as shown in formula (12).

$$X = \{x_0, x_1, \cdots, x_n\} \quad (12)$$

Because of the topological transitive nature of chaotic mapping, its iterative orbits are dense everywhere in the solution space. That is, for any two points such, as x_1 and x_2 in the set E and their respective neighborhoods U_1 and U_2, there exists a sufficiently large n to produce formula (13).

$$f^n(U_1) \cap U_2 \neq \varnothing \quad (13)$$

This shows that for particle swarm optimization algorithm, adding chaotic search step can walk to any point in the solution space within a certain number of times of search, thus can effectively avoid the local optimal solution.

At the end of each iteration, the CRPSO model will generate a chaotic sequence based on $gbest$ particle. The particle with the highest fitness is selected from the sequence and compared with the historical optimal particle to obtain the current optimal particle.

Algorithm : *chaotic hierarchical gene Replication model construction algorithm*

Step1*: Initialize the population, given the population partition threshold are* θ_1, θ_2.

Step2*: The fitness of each particle in the population is calculated, and the optimal individual is found in the initial population as the initial state of the historical optimal individual* ($gbest$) *, and an individual is randomly selected as the initial state of the iterative optimal particle* ($pbest$).

1. **while** $t < MAXITERATION$:
2. *Sort(particls,key=fitness);*
3. **for** *i in particles:*
4. ***Step3****: Update the velocity of particle[i].*
5. ***Step4****: Update the coordinates of particle[i].*
6. **if** $i > \theta_2$:
7. $perform\ \sigma(particle[i], gbest)$;
8. **else if** $i>\theta_1 \wedge i<\theta_2$:
9. $perform\ \sigma(particle[i], pbest)$;
10. **else**
11. $continue$;
12. $new_sSeries = getChaosSeries(gbest.solution)$;
13. $cgbest = find\_best(new_sSeries)$;
14. **if** $cgbest.fitness > gbest.fitness$
15. $pbest = cgbest$;
16. **if** $pbest > gbest$
17. *gbest=pbest;*
18. **output:** *gbest;*

Step 1: Initialize the population and given the population partition threshold θ_1, θ_2 ; **Step2(1-17):** Construction method of CRPSO model, According to the threshold value of particle , whether the current particle is the global optimal solution is determined; **Step3- Step4(4-17):** These two steps represent an operation to update the velocity and coordinates of particle; According to the judgment that the updated information of particle and it is added to the chaotic sequence (12-17), whether the particle is the global optimal solution is determined.

4. RESULTS AND ANALYSIS

The environment in this experiment is aimed at the small-scale cluster characteristics of the edge cloud. The physical host model is Intel(R) Core(TM) I7-9700 @3.00GHZ, 4 Core CPU, 32G memory. Moreover, CentOS 7 Linux virtual operating system is installed inside the physical machine, and the Kubernetes cluster

environment is configured in the virtual machine. The node range is set to 3-7 nodes (1 master node and the rest are slave nodes). The hardware configuration information is shown in Table 1. The specific verification index includes five aspects: algorithm fitness, convergence speed, CPU variance, memory variance, and total resource variance.

Table. 1 The hardware configuration information

Node	CPU	Memory
master	2 Core	3072 MB
worker	2 Core	2560 MB

In the experiment, five data sets of different sizes were set, as shown in Table 2. The data set was selected from random tasks automatically generated by the Kubernetes . DPSO (discrete particle swarm optimization), CPSO (chaos optimization particle swarm optimization), ACO algorithm (ant colony algorithm) [18]-[19] and GA algorithm (genetic algorithm) [20] are selected as the comparison algorithm.

Table. 2 Mapping between data set numbers and tasks

Number of the data set	Information for each data set
1	3 node——12 tasks
2	4 node——18 tasks
3	5 node——20 tasks
4	6 node——24 tasks
5	7 node——28 tasks

4.1. Efficiency analysis of CRPSO model

The results of each algorithm are shown in Fig. 3 . The number of iterations for each algorithm is 300, where the horizontal axis of the stack diagram represents the number of the data set, and the vertical axis represents the fitness fluctuation of each algorithm with the increase solution space.

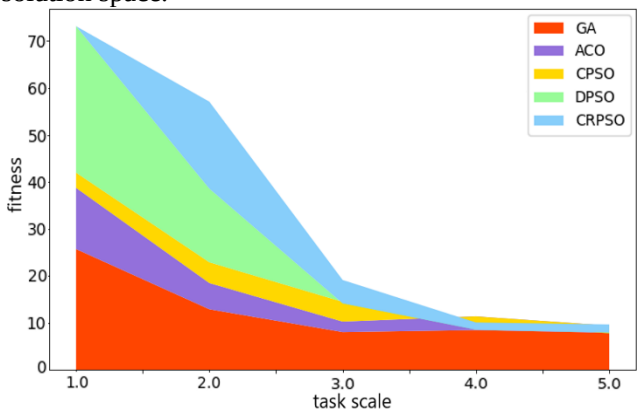

Fig. 3 The fitness of each algorithm with the increase solution space

As shown in Fig. 3, CRPSO algorithm has the highest performance among all algorithms, showing leading advantages in data sets of different sizes. The performance of CPSO algorithm lags behind DPSO algorithm when the data set size is small, and gradually exceeds DPSO algorithm as the data set size increases. The performance of ACO algorithm and GA algorithm is lower than the above three algorithms, because they are easy to fall into the local optimal solution, and GA algorithm is easy to lose the good solution due to the destruction of crossover operator.

Next, the number 2 data set was selected (4 nodes were placed with 18 tasks), and the above 5 algorithms were repeated for 200 times, and then the different quantiles were calculated and the boxplot was drawn. The ordinate indicates the load balance degree. Shown as in Fig. 4.

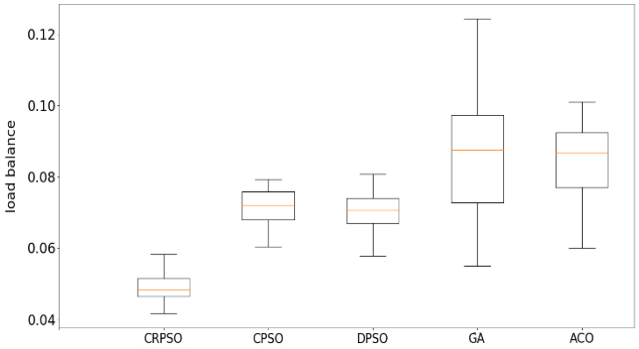

Fig. 4 The load balancing results of each algorithm in the same solution space

Fig.4 shows that the load balance of the solution obtained by CRPSO algorithm is much higher than that of the other four algorithms under the number 2 dataset, indicating that CRPSO shows leading performance under this problem. The upper and lower limits of the boxplot corresponding to the CRPSO algorithm are shorter than the other four algorithms, which indicates that the stability of the CRPSO algorithm is also good. Among them, the performance and stability of CPSO algorithm and DPSO algorithm are approximately the same in this solution space, because their search ability is sufficient to solve problems of this scale. The performance of GA algorithm and ACO algorithm is similar, but the stability of GA algorithm is very poor, because the crossover operator and mutation operator are easy to destroy the good structure of the solution without optimization, while ACO algorithm is easy to fall into the local optimal solution.

Next, a further comparison is made based on the average fitness and standard deviation of each algorithm. Shown as in Fig. 5.

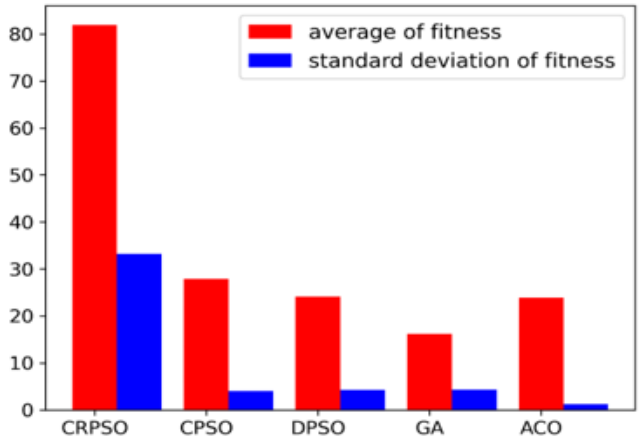

Fig. 5 The average fitness and standard deviation of each algorithm

It can be seen from Fig. 5 that the average fitness level of CRPSO is much higher than the other four algorithms, which indicates that the optimal solution of CRPSO algorithm is better than the other four algorithms, and the convergence speed is fast, because in the iteration, the fitness of the algorithm is at a high level. However, although CRPSO algorithm has a low fitness at the beginning of iteration, it has a large standard deviation.But the speed of finding the high fitness region is very fast, which indicates that the efficiency of the algorithm to jump out of the local optimal solution is very high.

The low standard deviation of CPSO, DPSO, GA and ACO algorithm indicates that they are trapped in the local optimal solution.Because CPSO algorithm cannot retain good individuals, the randomness of search is large.The search scope of DPSO algorithm is limited and randomness is large, so the overall mean fitness is relatively small.GA algorithm has the characteristic of fast convergence, and it is difficult to find the optimal solution for this problem effectively.ACO algorithm also has the characteristics of fast convergence, so it is difficult to jump out of the local optimal solution.

As can be seen from Fig. 6, CRPSO algorithm has found the optimal solution when the iteration reaches half of the maximum number of iterations. A longer box body means that the local optimal solution has been jumped out, so that the fitness difference between the early and late iterations is large. The DPSO and ACO algorithms fall into the local optimal. Although CPSO has a wide search range, it is difficult to retain excellent individuals, so the overall fitness is low. GA also fails to jump out of the local optimal solution because it is easy to destroy excellent individuals.

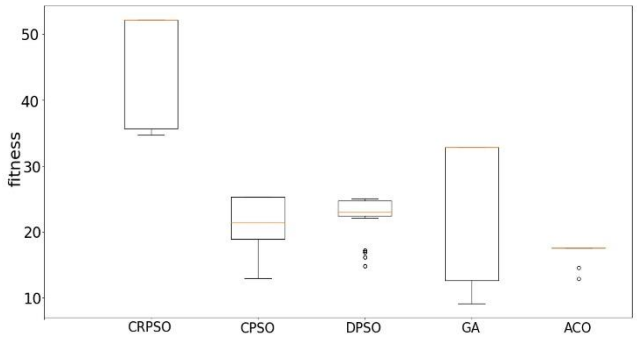

Fig. 6 The Boxplot of standard deviation fitness of each algorithm

4.2. Analysis of resource allocation results of CRPSO model in edge cluster

This paper selects Kubernetes as the edge cloud platform. Next, the CRPSO algorithm is connected to the Kubernetes cluster, and the number 2 data set is selected to carry out five groups of experiments. In this scenario, the number of feasible solutions in each group is 4^{18}, that is, the number of solutions conforms to the edge cloud high concurrent task solving scenario. In each experiment, CRPSO model deployment container group and Kubernetes automatic deployment container group were used for two comparison experiments respectively. The resource requests are shown in Table 3.

Table. 3 Mapping between data set numbers and tasks

Resources	Task (Units: Group)								
CPU	1	2	3	4	5	6	7	8	9
	0.13	0.279	0.065	0.29	0.096	0.237	0.06	0.056	0.2
	10	11	12	13	14	15	16	17	18
	0.05	0.07	0.204	0.146	0.06	0.04	0.117	0.06	0.27
Memory	1	2	3	4	5	6	7	8	9
	116	82	141	65	101	101	70	190	104
	10	11	12	13	14	15	16	17	18
	161	178	113	179	129	193	329	100	92

By comparison, the variance of CPU and memory are shown in Fig. 7 and Fig. 8.

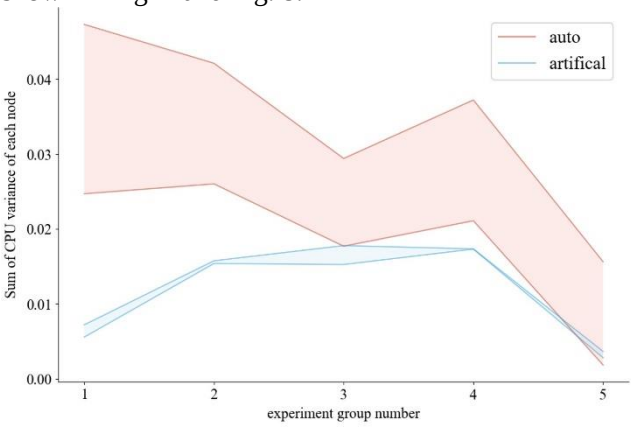

Fig. 7 The variance of CPU in number 2 data set

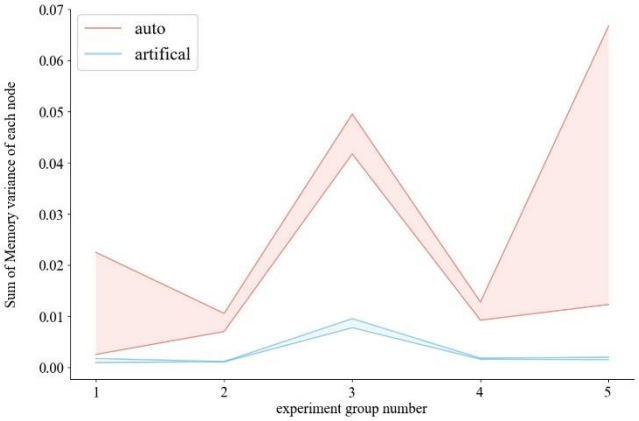

Fig. 8 The variance of memory in number 2 data set

According to Fig. 7 and Fig. 8, under the same conditions, CRPSO algorithm can achieve container group scheduling in a more balanced manner. In the third experiment, due to the increase of task resource requests on each node, the results of CPU and memory of Kubernetes automatic deployment showed large fluctuations between the proposed algorithm and Kubernetes automatic deployment. The resource variance of the proposed algorithm is still lower than that of the latter. Kubernetes internal scheduling algorithm is more based on the local optimal algorithm to implement resource scheduling. When a sudden large task request occurs, the task is processed according to the order of the request queue. As a result, the resource usage of a node is high and the CPU and memory variance is large, which is not good for the long-term and stable operation of the node.

Next, the total variance of each resource in the experiment was observed. Shown in Fig.9.

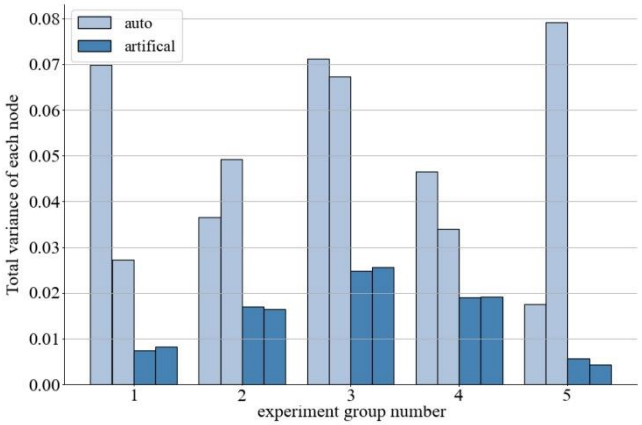

Fig. 9 The total resource variance in number 2 data set

As shown in Fig. 9, the total variances of two times of CPU and memory of CRPSO model in each group of experiments are significantly lower than those of Kubernetes, and the total variances of this model are close to each other in each deployment. After calculation, the CRPSO model can improve the total variance of resources by 69.74%. Therefore, the operation results are stable.

5. CONCLUSION

In this paper, a CRPSO adaptive model is proposed based on the edge cloud task request and Kubernetes cluster environment characteristics. Combining with chaotic sequence generation theory, the fitness and load balancing results of CRPSO model are obtained by designing the scene of increasing solution space. Experiments show that this model has obvious advantages over the comparison algorithm in terms of fitness, and the convergence rate of this model remains optimal even if the solution space is set to be exponential. Through the multi-objective solution operation, compared with the Kubernetes automatic deployment algorithm, the proposed model can improve the total variance of CPU and memory resources by 69.74%. In this paper, the statistical method of experimental results is manual, but the code of automatic statistical results has been written. In the future, this method will be applied to improve the experimental efficiency.

ACKNOWLEDGEMENTS

This work was supported in part by the science and technology project of "13th Five-Year" planning of the Education Department of Jilin Province (JJKH20200801KJ); the key project of "13th Five-Year" planning of the Education Science of Jilin Province (ZD19019); Scientific research projects of higher education in Jilin Province (JGJX2020D63).

REFERENCES:

[1] Pan J, McElhannon J (2018) Future edge cloud and edge computing for internet of things applications. IEEE Internet Things J 5(1):439–449.

[2] Sharma SK, Wang X (2017) Live data analytics with collaborative edge and cloud processing in wireless IoT networks. IEEE Access 5(99):4621–4635.

[3] Du B, Huang R, Xie Z et al (2018) KID model-driven things-edge-cloud computing paradigm for trafc data as a service. IEEE Netw 32(1):34–41.

[4] Xu J, Hao Z, Zhang R et al (2019) A method based on the combination of laxity and ant colony system for cloud-fog task scheduling. IEEE Access 7:116218–116226.

[5] Meena V, Arvind V, Vijayalakshmi P et al (2018) Optimized task clustering for mobile cloud computing using Workfowsim. In: 2018 2nd International Conference on Inventive Systems and Control (ICISC), pp 1000–1005.

[6] Wang T, Wei X, Tang C et al (2018) Efcient multi-tasks scheduling algorithm in mobile cloud computing with time constraints. Peer-to-Peer Netw Appl 11(4):793–807.

[7] Tang C, Xiao S, Wei X et al. (2018) Energy efcient and deadline satisfed task scheduling in mobile cloud computing. In: 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), pp 198–205.

[8] Guo X, Liu L, Chang Z et al (2018) Data ofoading and task allocation for cloudlet-assisted ad hoc mobile clouds. Wirel Netw 24:79–88.

[9] Zeng D, Gu L, Guo S et al (2016) Joint optimization of task scheduling and image placement in fog computing supported software-defned embedded system. IEEE Trans Comput 65(12):3702–3712.

[10] LI Bo, HUANG Xin, NIU Li, et al. Task offloading decision in vehicle edge computing environment [J]. Microelectronics & Computer, 2019, 36(2): 78-82.(in Chinese).

[11] YOU Changsheng, ZENG Yong, ZHANG Rui, et al. Asynchronous Mobile-Edge Computation Offloading: Energy-Efficient Resource Management [J].IEEE Transactions on Wireless Communications, 2018,17(11): 7590-7605.

[12] Jiang K, Ni H, Sun P et al (2019) An improved binary grey wolf optimizer for dependent task scheduling in edge computing. In: 2019 21st International Conference on Advanced Communication Technology (ICACT), pp 182–186.

[13] Kennedy J,Eberhart R.Particke swarm optimization[C]//Proco IEEE International Conference on Neural Networks.Piscataway,NJ:IEEE.Press,2011:1942-1948.

[14] Y. Xie et al., "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment," Future Gener. Comput. Syst., vol. 97, pp. 361–378, Aug. 2019, doi: 10.1016/j.future.2019.03.005.

[15] Xiaohui Wang,Haoran Gu,YuXian Yue. The optimization of virtual resource allocation in cloud computing based on RBPSO[J]. Concurrency and Computation: Practice and Experience,2020,32(16).

[16] Jian C, Li M, Kuang X (2018) Edge cloud computing service composition based on modified bird swarm optimization in the internet of things. Cluster Compute 1-9. https://doi.org/10.1007/s10586-017-1630-9.

[17] Sayed. Gehad. Ismail.et al. A novel chaotic salp swarm algorithm for global optimization and feature selection [J]. Applied Intelligence, 2018, 48 (10): 3486-3481.

[18] A. Dalvandi, M. Gurusamy, and K. C. Chua, ‘‘Application scheduling,placement, and routing for power efficiency in cloud data centers,’’ IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 4, pp. 947–960, Apr. 2017,doi: 10.1109/TPDS.2016.2607743.

[19] F. Ahamed, S. Shahrestani, and B. Javadi, ‘‘Security aware and energy efficient virtual machine consolidation in cloud computing systems,’’in Proc. IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, Aug. 2016,pp. 1516–1523.

[20] WEN Tao,SHENG Guo Jun, GUO Quan,LI Ying Qiu.(2013)Web Service Composition Based on Modifed Particle Swarm Optimization. CHINESE JOURNAL OF COMPUTERS,36(5) :1031-1046.