Edge cloud task scheduling model based on layered excellent gene replication strategy

Leilei Zhu1,2, Ke Zhao1, Huaze Lin1, Dan Liu1, and Li Li1,*
1Changchun University of Science and Technology, Jilin, Changchun 130022, China
2Changchun University of Chinese Medicine, Jilin, Changchun 130117, China
*Corresponding authors: Li Li (e-mail: ll@cust.edu.cn)

Abstract. With the development of the Internet of Things and 5G. Edge cloud technology has gradually become a research hotspot. When facing the massive and concurrent tasks of terminal users, reasonable resource scheduling strategy is a key technology. Because edge cloud needs to respond quickly to real-time tasks and ensure the stability of nodes at the same time, the optimal task scheduling strategy needs to be selected to meet the low latency requirements of edge users. In view of the above problems in resource allocation of edge cloud, this paper established a layered excellent gene replication strategy (HEGPSO model), in which the optimal replicator is added, and an evolutionary particle swarm optimization algorithm is proposed. In each iteration, the population is divided into three layers based on individual fitness. After that, the optimal replication factor is added to each layer of individuals to enhance the global search ability of the algorithm and ensure the good convergence of the algorithm. Finally, a balanced resource allocation model is established. Experiments show that the HEGPSO model proposed in this paper has high fitness and fast convergence speed, and is suitable for large-scale task access scenarios.

1. Introduction

Edge cloud is an important part of the development of distributed cloud architecture, which is mainly used to process the explosive growth of edge data in multiple scenarios. Each edge cloud realizes independent regional autonomy and maximizes resource utility by combining its own resources and management characteristics. However, due to the obvious difference in computing and storage capacity among edge nodes, it is necessary to implement a reliable resource allocation strategy when processing heterogeneous data in multiple scenarios. Thus, on the basis of ensuring the efficient operation of the edge cloud, the edge cloud collaborative service is finally realized. Figure 1 shows the framework for edge resource allocation when terminal tasks arrived.
Based on the traditional cloud computing resource allocation strategy and considering the characteristics of edge cloud resources, it is of great significance to establish an adaptive resource allocation strategy for reducing resource consumption of edge nodes, maintaining node stability and improving QoS of edge cloud services.

Taleb T et al. sorted out the architecture and business process of edge cloud computing under 5G and adopted heuristic algorithm to solve the problem of multi-step task segmentation [1]. Gascon-s J et al. designed a comprehensive adaptive cloud edge middleware, providing a strong guarantee for the study of the overall edge cloud [2]. Zhang Haibo et al. designed the task unloading and resource optimization method of edge computing in super-dense networks [3]. Shang G W et al. use mixed integer program to solve the optimal solution, improving access delay and load balancing effect [4]. The above research is mainly based on improving the unloading efficiency of edge tasks and reducing the response time and stability of edge nodes.

2. Related work
Because edge computing nodes mainly process data sources generated by edge users, some edge computing nodes may suffer from uneven load, resulting in redundant nodes and degrading service performance of nodes. In view of this problem, relevant scholars have done the following research.

Based on the central mode, You C et al. proposed an energy-saving resource configuration scheme for mobile edge computing offloading, but the scheme did not involve the waiting time and congestion of edge cloud computing nodes [5]. On the basis of multi-user and multi-task, Chen M H et al. added the allocation and processing cost of computing resources, and also ignored the serious delay caused by server congestion [6]. Weihua H et al. proposed a load balancing implementation strategy based on fuzzy clustering of weight preference [7]. Alnusairi T S et al. designed an optimization algorithm based on mixed particle swarm optimization, but this algorithm ignored the special case that the workload was dependent [8]. Puthal D et al. proposed a new load balancing method from the perspective of improving the security of edge data centers, but the real-time network needs to be improved [9].

The above research results show that edge cloud has the characteristics of distributed parallel processing architecture, but due to the real-time, dynamic and unpredictable scale of cloud services, it is easy to cause uneven load of edge cloud, reduce the service performance of edge cloud, and cause unbalanced use of resources.

Jiang Tongquan et al. proposed the session migration method of static single type streaming media edge cloud based on dynamic threshold allocation, but the effect of this method is limited when dynamic data oscillates violently [10]. Kang Y et al. proposed a lightweight scheduler, Neurosurgeon, which
reduces the power consumption, system response time and throughput of mobile devices [11]. Lan L et al. proposed an energy-aware task unloading mechanism in multi-user mobile edge cloud computing, which can make unloading decisions for each mobile user under multi-channel wireless interference [12]. Kaur K et al proposed a traffic scheduling algorithm based on energy efficiency delay and bandwidth [13]. Chunlin Li et al. proposed an adaptive resource allocation algorithm based on edge cloud architecture [14]. Mahbuba A et al. proposed a multi-objective resource allocation strategy for intelligent robot workflow in the Internet of Things, transformed the actual workflow resource allocation problem into a constrained multi-objective optimization problem, and introduced the concept of middle-layer edge cloud [15]. Tang H et al proposed a weighted optimal matching algorithm, but when the data scale is too large and the resources and equipment are greatly different, the algorithm performance will fluctuate to some extent [16].

The above research results show that resource allocation strategy of edge cloud has a great impact on its service performance. However, the execution efficiency of existing research content and optimization of comprehensive resource indicators have certain limitations. In this paper, an evolutionary particle swarm optimization (PSO) algorithm based on hierarchical good gene replication (HEGPSO model) is proposed by adding optimal replicators. Finally, experiments prove that the proposed algorithm has good efficiency and stability in dealing with high-dimensional edge cloud resource scheduling problems.

3. HEGPSO model establishment

In this paper, the physical machine set in edge clustering is defined as \( H = \{H_1, H_2, ..., H_3\} \); Each host in the cluster has the number of M VMS to handle concurrent tasks. The VM set in the host is defined as \( VMList = \{VM_1, VM_2, ..., VM_M\} \). Therefore, this problem studies that given n cloud tasks and M virtual machines, the optimal allocation scheme can be worked out to achieve the optimal balance of resource utilization of the entire virtual machine cluster, that is, there is no node overload or underload phenomenon, and the difference of resource utilization of each node in the cluster is not too large. This article considers that a VM has two types of resources: CPU (computing resources) and memory capacity. The VM definition is shown in Formula 1.

\[
\text{Cloudlet} = \{CPU\_request, Mem\_request\} \tag{1}
\]

Given n cloud tasks, the list is shown in Formula 2.

\[
\text{CloudletList} = \{C_1, C_2, ..., C_N\} \tag{2}
\]

Where the assignment of cloud task \( i \) in VM \( j \) is represented by \( a_{ij} \) then an assignment strategy can be represented by matrix A, as shown in Formula 3.

\[
A = \begin{pmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{pmatrix} \tag{3}
\]

If \( a_{ij} = 1 \), it indicates that cloud task \( i \) is placed on VM \( j \); otherwise, \( a_{ij} = 0 \). Because each task cannot be divided, it can be assigned to only one compute node at a time. Therefore, Formula 4 must be met.

\[
\sum_{i=0}^{n} a_{ij} = 1 \tag{4}
\]

That is, under allocation policy A, the total utilization of resource \( k \) on VM \( j \) is Formula 5.

\[
resource_{k\_usage} = \sum_{i=1}^{n} a_{ij} \times C_{ik} \tag{5}
\]

Where \( C_{ik} \) represents the quantity required for resource \( k \) in task \( j \). If the capacity of resource \( k \) on VM \( j \) is \( capacity_{jk} \), the constraints shown in Formula (6) must be met. In order to facilitate unified calculation, the above formula is written in a standardized form, that is, utilization rate, \( percent_{jk} \) of
resource \( k \) on virtual machine \( j \), then the above formula can be written as Formula (7).

\[
\text{capacity}_j^k > \text{resource}_k \text{ usage}\
\]

\[
\text{percent}_j^k < 1
\]  

In order to measure the effectiveness of allocation strategy \( A \) on node clusters, this paper uses the load balance degree to describe cluster load balancing. The definition of balance is shown in the Formula (8).

\[
\text{balance} = \sum_{k=1}^{r} \sum_{j=1}^{m} \frac{\left(\text{percent}_j^k - \text{percent}_k\right)}{m-1}
\]

Therefore, this problem is transformed into: given a cloud task list \( \text{CloudletList} \) and a virtual machine list \( \text{VMList} \), search for an optimal allocation strategy \( A \) to minimize the balance, namely the Formula (9). Among them, constraint conditions such as formula need to be satisfied the Formula (10).

\[
A = \text{argmin}_A \left( \sum_{k=1}^{r} \sum_{j=1}^{m} \frac{\left(\text{percent}_j^k - \text{percent}_k\right)}{m-1} \right)
\]

\[
\left\{ \begin{array}{l}
\text{percent}_j^k \leq 1 \\
\sum_{i=0}^{n} a_{ij} = 1
\end{array} \right.
\]

In this paper, a discrete particle swarm algorithm (HEGPSO model) is proposed based on pattern theory and discrete particle swarm optimization (DSO). The speed update formula and position update formula of traditional particle swarm optimization algorithm are as Formula (11) and Formula (12).

\[
v(t + 1) = \omega v(t) + c_1 r_1 (p_{\text{best}} - x(t)) + c_2 r_2 (g_{\text{best}} - x(t))
\]

\[
x(t + 1) = x(t) + v(t + 1)
\]

Let the form of the target solution be an \( n \)-dimensional vector, and the value range of vector elements is \([0, m]\). In each iteration, the population was divided into three layers based on individual fitness, and the updating formula of the individuals at each layer after the division was different, making the population more self-organized.

**Definition 1**: solution space for \( S \), shown as Formula (13). For each \( s \in S \), elements of vector \( s \) meet \( \{s_i|s_i \in 0, 1, 2, \ldots, m - 1\} \). Cloud task \( C_1 \) is assigned to compute node \( V_{s_{i1}} \), cloud task \( C_2 \) is assigned to compute node \( V_{s_{i2}} \), and cloud task \( c_1 \) is assigned to compute node \( v_{s_{1i}} \), where \( N \) is the number of particles in the particle swarm.

\[
s_i = \{s_{i1}, s_{i2}, \ldots, s_{im}\}, i = 1, 2, \ldots N
\]

**Definition 2**: Remembering the operation \( \sigma \) on space \( S \) (Formula 14). Binary copy operation for equal length vectors \( s_i \) and \( s_j \). The operation is: given the length \( n \) of vectors \( s_i \) and \( s_j \), the replication ratio is \( c \), where \( 0 < c < 1 \); From the random position \( a(0 \leq a < n) \) Starting, the number of \( cn \) vector elements in the copy of vector \( s_j \) are overwritten into vector \( s_i \). Since this operator is cyclic, if an index exceeds the length of the vector, the copy index is mod \( n \).

\[
\sigma(s_i, s_j, c)
\]

For HEGPSO model, the excellent structure of \( C \) can converge to the global optimal solution. Meanwhile, for each generation of \( g_{\text{best}} \) particles, the length of its copy should be greater than that of \( l_{\text{best}} \). In view of the PSO algorithm's characteristics of wide search range and prematurity, the actual selection of parameters should focus on global optimization. Therefore, 0.7 and 0.4 are preferable for \( g_{\text{best}} \) and \( l_{\text{best}} \). Based on the above definition, the updating formula of HEGPSO is shown in Formula (15).
\[ x(t) + v(t+1), t < \theta_1 \]
\[ \sigma(x(t)+v(t+1), \text{lbest}, c_1), \theta_1 < t < \theta_2 \]
\[ \sigma(x(t)+v(t+1), \text{gbest}, c_2), t > \theta_2 \]

The population is divided into three levels. Given three properties of particle \( P \), velocity, position and fitness, they are the speed, coordinate and fitness value of particle \( P \) respectively. For a given \( p \) particle, its updating formula is shown as Formula (16).

\[ v(t + 1) = \omega v(t) + c_1 r_1 (\text{gbest. position} - x(t)) + c_2 r_2 (\text{gbest. position} - x(t)) \]
\[ x(t+1) = \begin{cases} 
  x(t) + v(t+1), & t < \theta_1 \\
  \sigma(x(t)+v(t+1), \text{lbest}, c_1), & \theta_1 < t < \theta_2 \\
  \sigma(x(t)+v(t+1), \text{gbest}, c_2), & t > \theta_2 
\end{cases} \]

4. Experimental results and analysis

4.1. Experimental environment Description

In order to prove the effectiveness of HEGPSO algorithm proposed in this paper for solving resource scheduling in edge environment, we have carried out many experiments. In this paper, Pyhton is used to conduct simulation experiments and randomly generate tasks. In order to prove the advantages of HEGPSO, this paper selected the discrete particle swarm optimization (DPSO) algorithm from [17] literature (plus references) and the ant colony algorithm (ACO algorithm) from [18] literature. The experiment mainly compared the load balance degree, fitness and convergence speed between the algorithms. The experiment was divided into two groups. The first group set the scene as a fixed number of tasks and observed the convergence speed and fitness score between algorithms. The first group is to set the number of tasks with equal interval growth and observe whether the algorithm is suitable for large-scale task access scenarios. The experimental environment was as follows: Intel(R) Xeon(R) CPU E3-1220 V2 @ 3.10GHz 3.50GHz, 12 GB ddr3 1333MHz memory, 64 bit Windows 10 OS, and pycharm 2018.

4.2. Result verification and analysis

4.2.1 The equivalent task request scenario solution

In this group of experiments, 800 tasks were set and compared with DPSO algorithm and ACO algorithm, as shown in Figure 2.

![Figure 2. Fitness results of three algorithms in equal number of tasks](image)
In this group of experiments, the ant colony algorithm converges to the local optimal solution prematurely. Although DPSO algorithm is more extensive than ACO algorithm search, it is easy to converge to the local optimal solution prematurely in the process of solving. ACO algorithm, on the other hand, tends to lead to a large probability gap between different choices and cannot jump out of the local optimal solution. Due to the introduction of the mechanism of pattern processing, HEGPSO can retain the excellent structure of the previous generation and ensure the randomness of the search, so as to expand the search, and the continuous search ability is stronger, so as to get closer to the global optimal solution, and the convergence speed is faster. Therefore, HEGPSO algorithm can effectively avoid convergence to the local optimal solution and solve the combinatorial optimization problem better.

4.2.2 The Incremental task request solution
In this group of experiments, the above algorithms are still used for comparison, and the longitudinal axis scoring results are marked according to the fitness function. The number of task requests is 200, 400, 600, 800 and 1000, respectively, so as to meet the random and large-scale concurrent task request scenario in the edge environment. The result is shown in Figure 3.

As can be seen from Figure 3, HEGPSO algorithm achieves the best effect in solving this problem, and far exceeds the other two algorithms in the case of large number of cloud tasks. However, DPSO algorithm and ACO algorithm are far behind HEGPSO algorithm in solving problems when there are too many tasks. The reason is that DPSO algorithm is unable to make adjustments to excessive cloud tasks due to the lack of effective adaptive mechanism. It cannot accurately search solution space region when solving combinatorial optimization problems and is easy to fall into local optimal solutions. ACO algorithm itself has strong convergence, but it cannot search widely in a large solution space.

5. Conclusion
This paper proposes an adaptive HEGPSO model based on the characteristics of concurrent task requests in edge cloud. Based on the elite preservation strategy, the model replicates some information of the superior individuals of the previous generation in different degrees according to the superior level of particles. Combined with BPSO algorithm, the concept of the superior replication operator is determined by mode theory. Considering the real-time response requirements of edge environment, the algorithm is compared with the comparison algorithm from two aspects of load balance fitness and convergence speed. By designing equal task request space and increasing task request space, the actual task request scene is simulated. Experimental results show that the proposed algorithm has some advantages in optimizing edge cluster load balancing. The model will be further improved by adding location influencing factors in the future.
Acknowledgements
This work was supported in part by the science and technology project of "14th Five-Year" planning of the Education Department of Jilin Province (JJKH20200801KJ); the key project of "14th Five-Year" planning of the Education Science of Jilin Province (ZD19019)

Reference
[1] Taleb T, Samdanis K, Mada B, et al. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Architecture & Orchestration[J]. IEEE Communications Surveys & Tutorials, 2017:1-1.
[2] Gascon-S J, et al. Poster: Towards a Distributed and Self-Adaptable Cloud-Edge Middleware[C]. 2018 IEEE/ACM Symposium on Edge Computing. ACM, 2018.
[3] Zhang Haibo et al. Task Unloading and Resource Optimization in Ultra-dense Networks based on Mobile Edge Computing [J]. Journal of Electronics & Information Technology, 2019(5).
[4] Shang G W, et al. Edge server placement in mobile edge computing[J]. Journal of Parallel and Distributed Computing, 2018.
[5] You C, et al. Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading [J]. IEEE Transactions on Wireless Communications, 2017, 16(3):1397–1411.
[6] Chen M H, et al. Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point[J]. IEEE Conference on Computer Communications, 2017, 09(1).
[7] Weihua H, et al. Fuzzy Clustering with Feature Weight Preferences for Load Balancing in Cloud [J]. International Journal of Software Engineering and Knowledge Engineering, 2018, 28(05):593-617.
[8] Alnusairi T S, et al. Binary PSOGSA for Load Balancing Task Scheduling in Cloud Environment[J]. International Journal of Advanced Computer Science & Applications, 2018, 9(5).
[9] Puthal D, et al. Secure authentication and load balancing of distributed edge datacenters[J]. Journal of Parallel and Distributed Computing, 2018.
[10] Jiang Tongquan et al. Session Migration Strategy of Streaming Media Edge Cloud Based on Dynamic Threshold Allocation [J]. Computer Engineering, 2017.
[11] Kang Y, et al. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge[J]. ACM Sigplan Notices, 2017, 52(4):615-629.
[12] Lan L, et al. An Energy-Aware Task Offloading Mechanism in Multiuser Mobile-Edge Cloud Computing[J]. Mobile Information Systems, 2018, 2018:1-12.
[13] Kaur K, et al. Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay[J]. IEEE Communications Magazine, 2018, 56(2):44-51.
[14] Chunlin Li, et al. Adaptive resource allocation based on the billing granularity in edge-cloud architecture[J]. Computer Communications, 2019.
[15] Mahbuba A, et al. Multi-objective resource allocation for Edge Cloud based robotic workflow in smart factory[J]. Future Generation Computer Systems, 2019, 97(8):119-130.
[16] Hengliang T, et al. Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud-edge environment[J]. Computer Communications, 2019, 134(1):70-82.
[17] Ali Husseinzadeh Kashan, Behrooz Karimi. (2009) A discrete particle swarm optimization algorithm for scheduling parallel machines. Computers & Industrial Engineering, 56: 216-223.
[18] ZHANG H Q, ZHANG X P, WANG H T, et al. Task Scheduling Algorithm Based on Load Balancing Ant Colony Optimization in Cloud Computing[J]. Microelectronics & Computer, 2015, 32(5):31-35.