Hindawi
Scientific Programming
Volume 2020, Article ID 3967847, 9 pages
https://doi.org/10.1155/2020/3967847

Research Article

A Task Scheduling Strategy in Edge-Cloud Collaborative Scenario Based on Deadline

Shudong Wang[1], **Yanqing Li**[1], **Shanchen Pang**[1], **Qinghua Lu**[2], **Shuyu Wang**[1] and **Jianli Zhao**[3]

[1]*College of Computer Science and Technology, China University of Petroleum, Qingdao 266000, China*
[2]*Data61, Eveleigh, NSW, Australia*
[3]*College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266000, China*

Correspondence should be addressed to Shanchen Pang; pangsc@upc.edu.cn

Received 24 October 2019; Accepted 27 December 2019; Published 18 March 2020

Guest Editor: Jinghui Zhang

Task scheduling plays a critical role in the performance of the edge-cloud collaborative. Whether the task is executed in the cloud and how it is scheduled in the cloud is an important issue. On the basis of satisfying the delay, this paper will schedule tasks on edge devices or cloud and present a task scheduling algorithm for tasks that need to be transferred to the cloud based on the catastrophic genetic algorithm (CGA) to achieve global optimum. The algorithm quantifies the total task completion time and the penalty factor as a fitness function. By improving the roulette selection strategy, optimizing mutation and crossover operator, and introducing cataclysm strategy, the search scope is expanded. Furthermore, the premature problem of the evolutionary algorithm is effectively alleviated. The experimental results show that the algorithm can address the optimal local issue while significantly shortening the task completion time on the basis of satisfying tasks delays.

1. Introduction

With the rise of edge computing, the convergence of cloud computing and edge computing has become a major focus [1–3]. Especially when we make great strides towards the digital era of the Internet of Everything, edge-cloud collaboration has become an important application in many scenes such as CDN, industrial Internet, energy, intelligent transportation, and security monitoring. Cloud computing and edge computing need to work closely together to better match the various demand scenarios, thus maximizing the value of edge computing and cloud computing collaboration. Take the example of an IoT scenario. The devices in the Internet of Things generate a large amount of data, and the data are uploaded to the cloud for processing, which will cause great pressure on the cloud. To share the pressure of the central cloud node, the edge computing node can be responsible for data calculation and storage within its own scope [4–6]. Cloud computing excels in global, non-real-time, long-cycle big data processing and analysis and can play an advantage in long-term maintenance, business decision support, etc. Edge computing is more suitable for local, real-time, short-cycle data processing and analysis. Edge computing can better support real-time intelligent decision making and execution of local business. There are some high real-time performance applications, such as industrial system detection applications, control applications, executive applications, and emerging VR/AR applications. Some scenarios require real-time performance within 10 ms or even lower [7, 8]. If data analysis and processing are all implemented in the cloud, it is sometimes difficult to meet the real-time requirements of the service. It seriously affects the business experience of end customers. But, usually more studies usually consider the process of unloading, ignoring the assignment of tasks after unloading.

Tasks can be scheduled to the edge or the far cloud based on energy consumption and time delay. For the problem that needs to be processed in the cloud center, how to perform

proper scheduling to achieve the goal is worthwhile research question.

Task scheduling methods in the cloud center can be divided into heuristic algorithms (such as RR and SJF), metaheuristic algorithms (based on biological incentives and swarm intelligence), and hybrid task scheduling algorithms [9]. In the scheduling process, various performance-based performance indicators such as system utilization, execution time, load balance, network communication cost, delay, and the like are used [10]. The heuristic task scheduling algorithm can easily schedule tasks and provide the best solution. However, it does not guarantee the best results and is easy to fall into partial selection. The metaheuristic algorithm is an improved algorithm based on a heuristic algorithm, which is a combination of random algorithms and local search algorithm[11–13]. It enables the exploration and development of search space and handles a large amount of search space information. In addition, it can use learning strategies to acquire and master information to effectively find approximate optimal solutions. Among them, genetic algorithm (GA), particle swarm optimization (PSO), and ant colony algorithm (ACO) are the most widely used evolutionary algorithms in the task scheduling in recent years [14]. However, these algorithms usually converge prematurely and are prone to finite optimally. When approaching the optimal solution, it may also swing left and right, making the convergence slower [15]. In genetic algorithms, the crossover operators become the main operators because of its global search ability and mutation operator is to become the auxiliary operator because of its local search ability. Genetic algorithms have the ability to balance the global search space with the local search space. Genetic algorithms always search for global and local spaces through crossover and mutation operators. They cooperate with each other and monitor each other. How to effectively cooperate with the intersection and mutation operations, make the convergence faster, and jump out of the local optimum in the solution process is a valuable research content of the current genetic algorithm.

This paper proposes a task scheduling strategy for edge-cloud collaborative computing based on disaster genetic algorithm. Considering the meaning of the cross operation, the individual optimal retention, and the magnitude of the mutation probability in the evolutionary process, the ability to optimize convergence and the three genetic operators of the genetic algorithm are improved. A penalty factor determines the execution time objective function based on the time delay. At the same time, a catastrophic strategy was introduced to simulate the phenomenon of disasters in biological evolution. During the first 1/2 iterations, premature aging may occur and the best chromosomes of successive generations will not develop at all. Therefore, we increase the probability of mutation, break the monopoly of the original gene, make the individual away from the current optimal solution into the group, increase the diversity of genes, and create new survival individuals. The algorithm we proposed can jump out of the local optimum and effectively alleviate the problem of premature convergence.

The rest of this article is organized as follows. Section 2 introduces the related work. Section 3 introduces the task classification strategy. Section 4 introduces the task scheduling model in the cloud center. Section 5 introduces the CGA algorithm. Section 6 introduces the experimental and comparison results. Finally, Section 7 summarizes this paper.

2. Relevant Work

Research on edge-cloud collaboration is still in the initial stage, but many domestic and foreign scholars have carried out related research and achieved research results on the task scheduling problem at the edge or cloud. Ke et al. [16] proposed classifying tasks according to whether they meet the delay and energy consumption. In the scheduling of tasks in the cloud, genetic algorithms are widely studied for their adaptability to various task scheduling problems. The genetic algorithm is appropriate for various task scheduling problems. The improvement of the genetic algorithm is mainly to improve the genetic operator and to achieve the purpose of improving the convergence speed and the performance of the classical genetic algorithm. At present, many corking algorithms have been proposed successively after experimentation and demonstration by scholars. Keshanchi et al. [17] proposed an improved heuristic-based genetic algorithm, called N-GA. The N-GA is used for the static task scheduling in the cloud. Akbari et al. [18] improved the performance of genetic algorithm by significantly changing genetic operators to ensure the sample diversity and reliable coverage of the entire space. In [19], a hybrid metaheuristic algorithm is offered, which uses the HEFT (Heterogeneous Earliest Completion Time) algorithm combined with PSO and GA to improve performance. Johnson proposed a rule-based genetic algorithm (JRGA) [20] for a two-stage task scheduling in data centers. In [11], the authors proposed a task scheduling scheme for heterogeneous computing systems built on a genetic algorithm, which maps each task to the processor according to the assigned priority to shorten the manufacturing time as much as possible. Goyal and Agrawal [21] proposed a model for scheduling a group of independent tasks on multiple machines and solved the question by combined the GA and the electoral heuristic algorithm. The goal of this model is also intended to minimize the maximum time. Kumar et al. [22] put forward a new task scheduling method, which integrated min-min algorithm and min-max algorithm in a genetic algorithm. The goal of the research is to shorten the generation time and execution time to the greatest extent [23].

However, the methods mentioned above may still fall into a local optimum when solving a multimode problem [10]. Therefore, the algorithm needs some strategies to avoid this limitation. Literature [24–28] mentioned an integer genetic algorithm using a “catastrophe” operator. It is designed to help to jump out of the local extreme points. The bionic significance of “catastrophe” operator and the improvement of disaster genetic algorithm in solving the above problems are emphatically introduced. These operations can mitigate the phenomenon of falling into a local optimum and premature convergence.

In addition, there are few studies that achieve the least total time based on the delay of meeting each task. Therefore, based on the research of genetic algorithms, this paper raises a task scheduling algorithm called CGA based on cataclysm strategy [29], which mainly considers the time delay to achieve the minimum total execution time. And the effectiveness of the proposed algorithm is checked by experiments.

3. Task Classification

In the system, we consider a set of tasks to be performed, each of which comes from an edge device which is denoted as $N = \{1, 2, 3, \ldots, N\}$. The tasks include interactive gaming, natural language processing, image location, etc [16]. Each task should be completed within the deadline. Each task with three attributes is defined as $\text{Task}_i = [\text{data}_i, d_i, \exp T_i], i \in N$. For Task_i, d_i is the size of the input data for the computation, which may include program codes, input files, etc [16].$\exp T_i$ is the deadline for completion of a task. data_i is the length of the task. Therefore, we must first classify the tasks that need to be processed to determine whether to execute in the cloud. According to the ratio of the delay of the task and the length of the task, the sensitivity of the task is determined. And finally, the tasks in the cloud will be scheduled to reduce total execution time.

Let f_i^c represent the computing power assigned to the Task_i by the edge device. Thus, we can get the time of the local execution of Task_i as

$$T_{\text{local}}^i = \frac{\text{data}_i}{f_l^c}. \tag{1}$$

The time transferred to the cloud is defined as

$$T_{\text{tran}}^i = \frac{d_i}{\text{Rate}}. \tag{2}$$

Rate is the upload rate of tasks transferred to the cloud; here the upload rate is a fixed value.

In order to facilitate subsequent task scheduling in the cloud, tasks need to be sorted according to sensitivity. The task sensitivity can be defined as

$$\text{sen}T_i = \frac{\text{data}_i}{\exp T_i}. \tag{3}$$

The complete task classification process is illustrated in Algorithm 1.

4. Task Scheduling Model in the Cloud Center

The task scheduling problem in the cloud is how to reasonably arrange each task to multiple virtual machines so that all tasks can be completed in a shorter execution time and meet the delay as much as possible [10]. Here, the following assumptions are made:

(1) There is no interdependence between tasks and tasks

(2) The size of the task and the computing speed of the virtual machine are known

Definition 1. Virtual machines on physical machines:

$$\text{PM}_i = [\text{VM}_1, \text{VM}_2, \ldots, \text{VM}_{\text{NVM}}], \tag{4}$$

where PM_i represents the host machine, Nvm represents the number of virtual machines, and VM_k represents the kth virtual machine resource in the cloud environment.

Definition 2. Virtual machine resources:

$$\text{VM}_K = [\text{IDV}_k, \text{MIPS}_k], \tag{5}$$

where IDV_k is the serial number of the virtual machine and MIPS_k represents the computing power of the kth virtual machine.

Definition 3. Task sequence:

$$T = [\text{TAsk}_1, \text{TAsk}_2, \ldots, \text{TAsk}_i, \ldots, \text{TAsk}_{\text{Ntsk}}], \quad i \in GC, \tag{6}$$

where Ntsk represents the number of tasks that need to be performed in the cloud and Task_i represents the ith task in the task sequence.

Definition 4. Task expected completion time.

The ECT matrix is used to represent the completion time of all tasks on each virtual machine resource.

$$\text{ECT} = \begin{bmatrix} \text{ECT}_{1,1} & \text{ECT}_{1,2} & \ldots & \text{ECT}_{1,\text{Nvm}} \\ \text{ECT}_{2,1} & \text{ECT}_{2,2} & \ldots & \text{ECT}_{2,\text{Nvm}} \\ \ldots & \ldots & \ldots & \ldots \\ \text{ECT}_{\text{Ntsk},1} & \text{ECT}_{\text{Ntsk},2} & \ldots & \text{ECT}_{\text{Ntsk,Nvm}} \end{bmatrix}. \tag{7}$$

The execution time required for each task to run on a computing resource (virtual machine) is calculated as follows:

$$\text{ECT}_{i,k} = \frac{\text{data}_i}{\text{MIPS}_k}, \quad k = 1, 2, \ldots, \text{Nvm}; \; i = 1, 2, \ldots, \text{Ntsk}. \tag{8}$$

Let the task set be assigned to the kth virtual machine; then, the task completion time $\text{RT}(k)$ on the kth virtual machine is

$$RT(k) = \sum_{l \in N_l} \text{ECT}(l, k), \quad k = 1, 2, \ldots, \text{Nvm},$$
$$\forall l \in [1, \text{Ntsk}] \text{mapped to } k\text{th VM}, \quad k = 1, 2, 3, \ldots, \text{Nvm}. \tag{9}$$

AllNTime is the maximum completion time for each computing resource:

$$\text{AllNTime} = \max(\text{RT}(K)), \quad k = 1, 2, \ldots, \text{Nvm}. \tag{10}$$

Definition 5. Matching matrix A.

We can get the matrix A:

(1) Initialization
(2) Task set: N = {1, 2, 3, . . . , N};
(3) Categorized task sets: GC = GL = ϕ;
(4) For each task $i \in$ N do
(5) Calculate senT_i by data$_i$/exp T_i, respectively;
(6) If ($T_{\text{local}} >$ exp T_i) then
(7) $i \Longrightarrow$ GC;
(8) Else if ($T_{\text{local}} \leq$ exp T_i) then
(9) $i \Longrightarrow$ GL;
(10) End if;
(11) End for;
(12) Output: GC (Sort by sensitivity in ascending order), GL.

Algorithm 1: The algorithm for classifying the tasks.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,\text{Nvm}} \\ a_{2,1} & \cdots & \cdots & \cdots \\ \cdots & a_{i,j} & \cdots & \cdots \\ a_{\text{Ntsk},1} & \cdots & \cdots & a_{\text{Ntsk,Nvm}} \end{bmatrix}. \quad (11)$$

Among them, $a_{i,j} = \{0, 1\}$. And the value of $a_{i,j}$ indicates whether the task numbered i is executed on the virtual machine numbered j, and if it is 1, it is executed.

5. CGA Algorithm

5.1. Algorithmic Thought. The three genetic operations of the genetic algorithm affect the convergence speed of the algorithm. This paper mainly considers satisfying the delay and minimizing the total execution time, and improves the selection operation and the crossover operation as well as the mutation operation of the genetic algorithm to generate a new generation of the population while simulating biological evolution in the iterative process. The catastrophic phenomenon in the process makes the algorithm increase individual diversity without expanding the population size, and it is easier to get rid of the optimal local trap. The algorithm flow chart is shown in Figure 1:

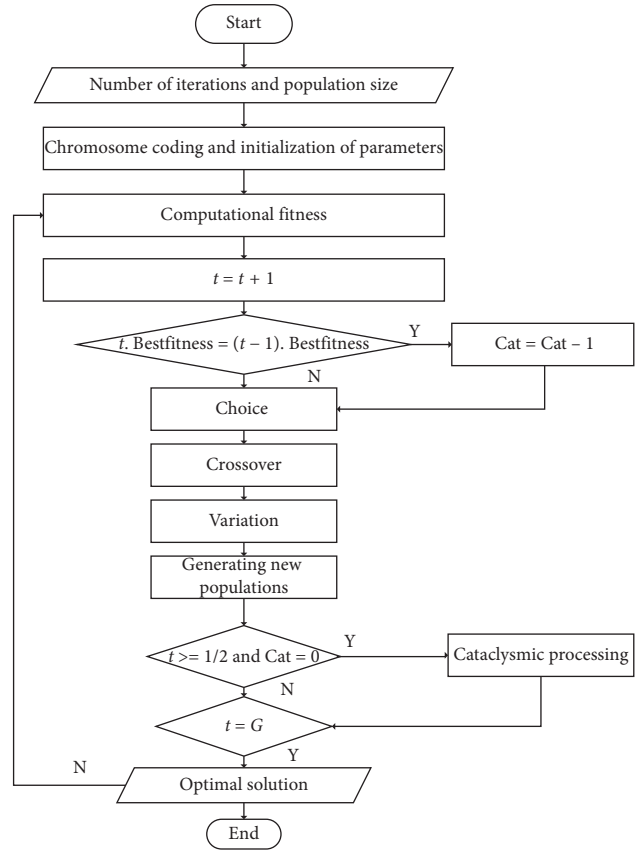

Figure 1: Algorithmic flow.

5.2. Basic Operations of the Algorithm

5.2.1. Encoding. In cloud computing scheduling problem, the encoding of solutions usually uses binary coded and real coded, where real coded is multi-to-one mapping pairing encoding. The task of this paper and the virtual machine are coded by the mapping pairing method [2]. For example, if there are M vms, that is, $\{v_1, v_2, v_3, \ldots, v_M\}$, and N tasks, that is, $\{\text{Tak}_1, \text{Task}_2, \text{Task}_3, \ldots, \text{Task}_n\}$, the length of the code will be N and the value of each gene will come from 1 to M, as shown in Figure 2:

Task	t_1	t_2	t_3	t_4	t_5	t_6
Vm	v_1	v_3	v_2	v_4	v_5	v_5
Code	1	3	2	4	5	5

Figure 2: Encoding.

5.2.2. Fitness Function. The fitness function represents the degree of an individual's fitness in the evolutionary process. The greater the fitness is, the easier it is to be retained in the evolutionary process. The fitness function will directly affect the performance of the algorithm and whether it can achieve the goal. In this paper, we need to consider the effect of time delay and execution time on individual fitness.

The difference between execution time and deadline for each task:

$$\sum_{i\in[1,i]} a_{i,j}\mathrm{ECT}_{i,j} + T^{i}_{\mathrm{tran}} - \exp T_i. \tag{12}$$

Penalty factor based on whether delay is satisfied:

$$\mathrm{punish} = \begin{cases} 0, & \text{if } \sum_{i\in[1,i]} a_{i,j}\mathrm{ECT}_{i,j} + T^{i}_{\mathrm{tran}} - \exp T_i \le 0, \\ \left| \sum_{i\in[1,i]} ba_{i,j}\mathrm{ECT}_{i,j} + T^{i}_{\mathrm{tran}} - \exp T_i \right|, & \text{if } \sum_{i\in[1,i]} a_{i,j}\mathrm{ECT}_{i,j} + T^{i}_{\mathrm{tran}} - \exp T_i > 0. \end{cases} \tag{13}$$

Because the goal is to minimize the total execution time of the task scheduling while meeting the deadline of tasks, the fitness function of this paper is designed as

$$\mathrm{fitness} = \frac{1}{\mathrm{AllNTime} + \sum_{i=1}^{\mathrm{Ntsk}} \mathrm{punish}}. \tag{14}$$

5.2.3. Improve Roulette Choice. The roulette selection method is also called the proportional selection method. The basic idea is that the larger the individual's adaptability is, the easier it is to be selected. The traditional roulette method can select the best individual, but it cannot guarantee that the best individual will remain to the next generation, and the subsequent crossover operation may destroy the best individual. Therefore, this paper combines roulette with the best individuals to save individuals with the greatest fitness in each generation directly to the next generation and does not participate in the crossover operation or mutation operation. The remaining individuals use traditional roulette to select the progeny population. The probability $Ps(j)$ of individual selection in traditional roulette is

$$Ps(j) = \frac{\mathrm{fitness}(j)}{\sum_{i=1}^{N} \mathrm{fitness}(i)}. \tag{15}$$

5.2.4. Crossover. The crossover operation of the traditional genetic algorithm is to select the number of individuals to cross according to the crossover rate, to generate a crossover operation for each of the intersecting individuals using the random function $\mathrm{rand}(1, n)$, and to map the two chromosomes to the segments after the location point are exchanged. Traditional crossover operations are prone to the situation of the high similarity of crossover fragments, at which time the crossover meaning becomes smaller. To this end, this paper sets a cross threshold, and only if the threshold is exceeded, the cross is considered meaningful. Otherwise, no crossover occurs. The threshold size represents the proportion of similar genes in the total gene. This operation is mainly based on the principle of preventing inbreeding and optimizing offspring in the process of human evolution. In this paper, we set the threshold to 0.8 and the crossover probability higher than 0.7 to avoid slowing down the speed of convergence rate caused by abandoning the cross operation because the similarity is too high. The specific crossover operation is shown in Figure 3:

5.2.5. Variation. A mutation operator is a very important operation. There are two purposes for introducing mutations into genetic algorithms: one is to make the genetic algorithm have local random search ability. When the genetic algorithm is close to the optimal solution neighborhood through the crossover operator, the local random search ability using the mutation operator can accelerate the convergence to the optimal solution [30]. In this case, the mutation probability should take a smaller value. The second is to enable the genetic algorithm to maintain group diversity to prevent immature convergence. At this time, the mutation probability should take a larger value. The probability of variation usually takes a small value and generally does not exceed 0.1.

In this paper, two variability values are set. When the number of iterations reaches 2/3, the mutation probability is reduced by 0.02. Determine the number of individuals that need to be mutated based on the probability of mutation, randomly select two locations on the chromosome, and exchange the values of the genes. The genic value may have not changed after the mutation operation was executed, which is equivalent to no mutation operation, and the variation operation is improved in order to ensure that the variation operation can be executed even if it is already a small probability event. If two genic values of mutation are the same, add the first random number to 1 and let it perform mutation operation with another gene point. If the first random number is still the same, increment the value by one until the value is different to ensure the mutation operation (see Figure 4).

5.2.6. Catastrophe. After many generations of evolution, the group may obtain a locally optimal solution. At this time, the group implies a large amount of information related to the local optimum, tending to premature convergence and the possibility of jumping out by operators such as crossover operation and mutation operation. It is possible to introduce "catastrophe" strategy, obtain some useful global information, and obtain a solution far away from the original locality with a large probability so that a larger diversity can be obtained at smaller group size. It can provide more opportunities to get rid of the original local optimal solution. However, the catastrophe cannot go through evolution all the time. We should consider avoiding the problem of destroying the optimal solution and reoptimizing in the later stage.

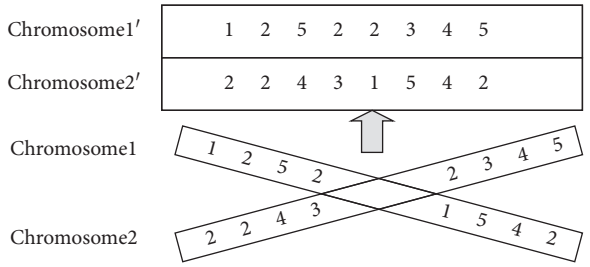

Figure 3: Crossover.

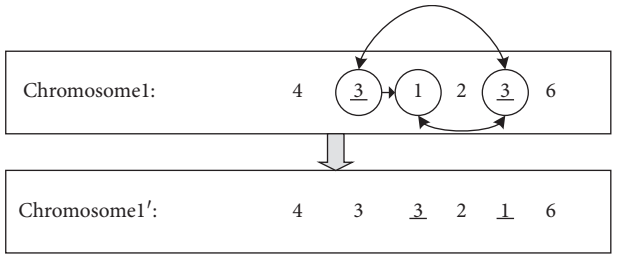

Figure 4: Variation.

The genetic algorithm has the disadvantages of easy to fall into local optimum and premature convergence [31]. Once it falls into local optimum, it will be difficult to jump out. For this reason, we add the catastrophic strategy mentioned in the literature [28] to this paper. By increasing the mutation probability to stay away from the current optimal, the solution that is far from the current optimal solution is included in the population to jump out of the optimal local solution. Catastrophic operation is shown in Algorithm 2.

5.3. Task Classification and Scheduling Description

Step 1: classify all tasks from different devices according to Algorithm 1.

Step 2: for tasks that need to be uninstalled to the cloud, sort by sensitivity. The initial coding is optimized according to the computing power of virtual machine.

Step 3: chromosome coding and initialization of parameters.

Step 4: calculate fitness.

Step 5: superposition algebras plus one.

Step 6: judge whether the optimal individual fitness of $(t-1)$th generation is equal to that of the tth generation, and if so, the catastrophe threshold is reduced by one; otherwise, it will continue.

Step 7: perform selection operation, cross operation, and mutation operation.

Step 8: generate the descendant population and determine whether the catastrophe threshold cat is equal to 0 (before t/2 iterations). If equal to 0, carry on the catastrophe operation.

Step 9: if the number of iterations reaches the maximum, output; otherwise, turn to step 4.

```
(1) Input: Catastrophe threshold cat;
(2)     cat = a;
(3)     For (t = 0; t < G/2; t++)
(4)       {
(5)          If (t. Bestfitness = (t−1). Bestfitness)
(6)            {
(7)               cat = cat−−;
(8)            }
(9)          If (cat = 0)
(10)              The first third variation;
(11)         Else
(12)              Continue circulation;
(13)           }
```

Algorithm 2: Catastrophic operation.

6. Evaluation

In this experiment, for tasks that need to be processed in the cloud, we used CloudSim 3.0 to implement the algorithms, by adding the bindCloudletToVM method in the DAtacenterBroker class; the CGA algorithm based on the catastrophe genetic algorithm is added to carry out the simulation experiment. Data such as resource computing power and task calculations are derived from data randomly generated in MATLAB. We choose the different number of tasks, and the experimental data of different iteration times are analyzed and compared with the time-based differential evolution algorithm (TDE) and simple genetic algorithm under the same data conditions. The TDE algorithm is based on differential evolution (DE) task scheduling algorithm that minimizes the completion time. The differential evolution algorithm is also a population-based heuristic search algorithm. There is a great similarity between differential evolution algorithm and genetic algorithm. They all include mutation, crossover, and selection operations, but the specific definition of these operations is different from the genetic algorithm. The experimental results are shown in Figures 5–9.

Parameter setting: crossover probability crossover = 0.8, maximum evolution algebra = 200, and mutation probability is 0.03, and in order to avoid errors as much as possible, this paper will perform ten times for each group of experiments and finally get the total task completion time. The experimental values are taken as the average of ten experiments.

When the number of tasks is small, the optimal effect is not obvious. However, the optimization of the algorithm is more obvious when the number of tasks is large. But the more tasks there are, the fewer tasks that are unloaded into the cloud, because as the number of tasks increases, the task takes longer to execute. With the increase of evolutionary algebra, the proposed algorithm can converge more quickly and save more time. Figures 5 and 6 show the changes of total task execution time and adaptive value of CGA algorithm, classical genetic algorithm, and TDE algorithm under different iterations. It can be seen that the effect of the classical genetic algorithm is the worst. The CGA algorithm uses less evolutionary algebra than other algorithms to get

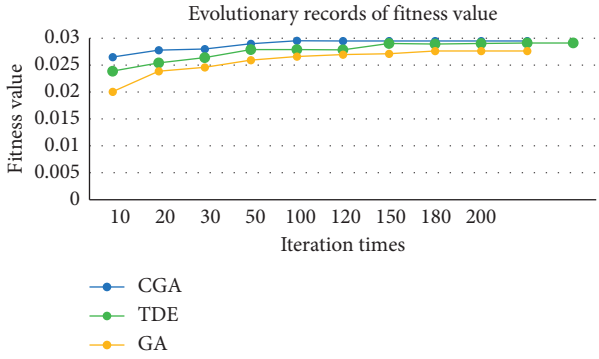

Figure 5: Evolution—200 iterations.

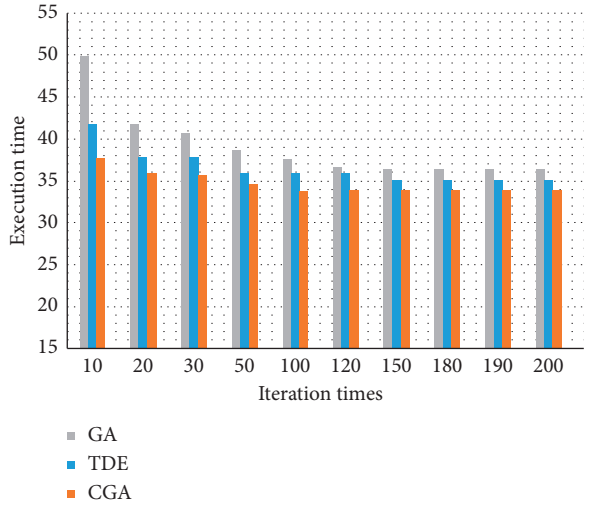

Figure 6: Execution time—50 tasks.

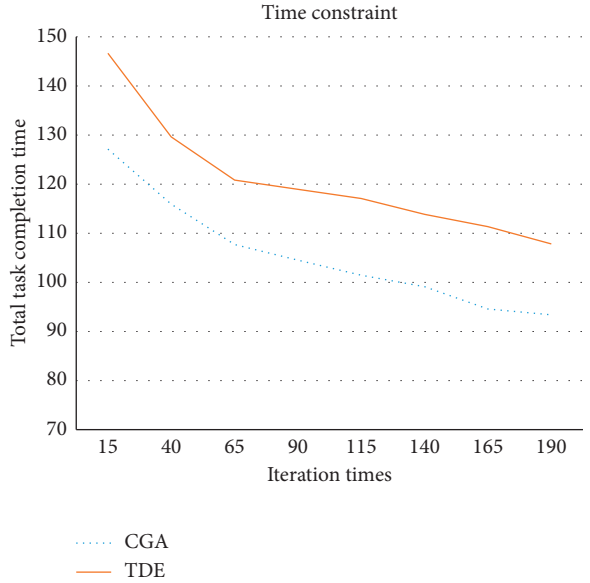

Figure 7: Execution time—200 iterations.

better average fitness. Among them, this paper also optimizes the initial population, and CGA algorithm can find the optimal solution faster. As we all know, the solution found

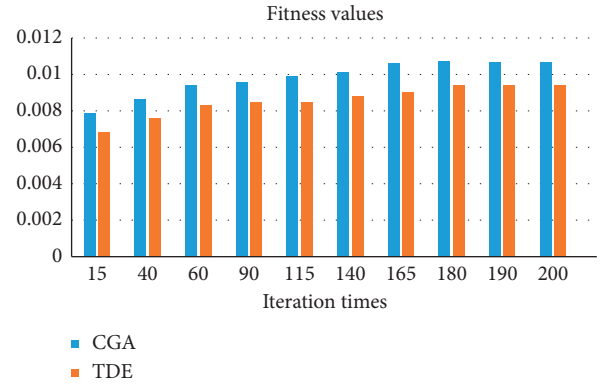

Figure 8: Evolution—200 iterations.

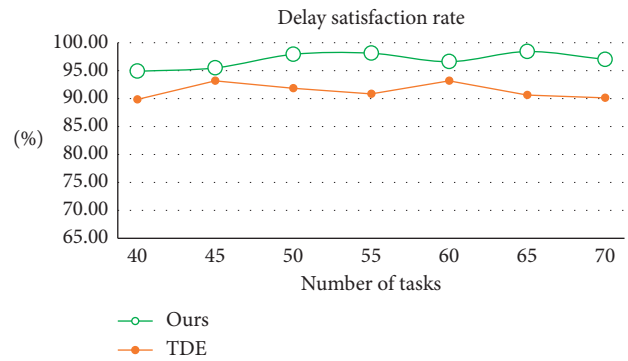

Figure 9: Delay satisfaction rate.

by genetic algorithm may not be optimal, but the experimental results show that the CGA algorithm is better than the other two algorithms, and it is easier to jump out of the local optimum and find the optimal solution. Figures 7 and 8 are comparisons between CGA algorithm and TDE algorithm. We can see that CGA algorithm can achieve the goal of this paper better.

According to the experimental results, it can be seen from Figure 9 that the delay satisfaction rate of the experiment is above 95%, which can meet the demand. And the performance of CGA algorithm is better than the TDE algorithm. In addition, the CGA algorithm is superior to the TDE algorithm in the task completion time and convergence speed of the evolutionary process, and its convergence speed is significantly better than the TDE algorithm and the traditional genetic algorithm. As the number of iterations increases, the CGA algorithm can find the optimal solution better and make the convergence rate faster. The mutation strategy called cataclysm policy is designed to help the population jump out of the local extreme points [27]. It can be seen that the catastrophic strategy in this paper does not slow down the convergence rate and destroy the optimal direction. Instead, it can help the operation to continuously optimize the population and is not easy to fall into the local optimum.

7. Conclusion and Future Work

The task scheduling in edge-cloud collaborative scenario is considered to be one of the critical challenges. Whether the

task is executed in the cloud and how it is scheduled in the cloud is an important issue. In the past, many heuristics and metaheuristic task scheduling strategies have been used in cloud computing or edge computing. Genetic algorithms have unique advantages that traditional methods do not have in solving complex problems such as big space, nonlinearity, and global optimization. They have been widely used in more and more fields. In this paper, we proposed a task scheduling strategy under deadline constraint, where tasks on edge devices could select the execution place including cloud and local devices. And the goal is to minimize the execution time of all tasks. The CGA algorithm as an alternative method to solve the task scheduling problem; this algorithm adds cataclysm strategy to it. We have considered the constraint of time [5] and optimized the task scheduling. The algorithm CGA was inspired by the behavior of the extinction in the Ice Age, and it is used as a global optimization algorithm [10].

The CGA algorithm we proposed was simulated in the CloudSim environment, and the main objective was to minimize the execution time and meet delay. The results are compared with the results of existing heuristic methods such as the traditional genetic algorithm (GA) and the time-based differential evolution algorithm (TDE). From the experimental results, we can also get the conclusion that the proposed CGA can efficiently schedule the tasks to the VM and achieve our goals.

In the future, we will consider improving the algorithm under conditions that are closer to the actual environment so that the algorithm can be applied to dynamic and real-time task scheduling in edge-cloud collaboration. Besides, we want to build a multi-objective version of CGA for optimizing the task scheduling problem in the cloud. Study of workflow scheduling using CGA is another future investigation. And we can also mine or forecast its potential relationships [32–34]. In addition, the method of task scheduling can consider many other parameters, such as the use of memory, peak of the demand, and overloads [10]. Besides, we can combine the Markov chain with the parallel computing framework and apply it in our model [35, 36].

Data Availability

Because this paper only deals with time and static tasks, we used randomly generated data to export it as a dataset for the length of tasks.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank International Networks Service and Bio-Computing Innovation Team from the college of Computer Science and Technology in China University of Petroleum (East China) for their discussion and technical support. This study was supported by the National Natural Science Foundation of China (nos. 61572522, 61873281, and 61572523).

References

[1] M. Satyanarayanan, "Edge computing," *Computer*, vol. 50, no. 10, pp. 36–38, 2017.

[2] Y. Mao, C. You, J. Zhang et al., "Mobile edge computing: survey and research outlook," 2017, http://arxiv.org/abs/1701.01090.

[3] Y. Yu, "Mobile edge computing towards 5G: vision, recent progress, and open challenges," *China Communications*, vol. 13, no. 2, pp. 89–99, 2017.

[4] G. Ananthanarayanan, P. Bahl, P. Bodik et al., "Real-time video analytics: the killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[5] J. Liu, Y. Mao, J. Zhang et al., "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT)*, Barcelona, Spain, July 2016.

[6] Z. Ke, Y. Mao, S. Leng et al., "Optimal delay constrained offloading for vehicular edge computing networks," in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017.

[7] Y. Yin, W. Xu, Y. Xu, Li He, and L. Yu, "Collaborative qos prediction for mobile service with data filtering and slopeone model," *Mobile Information Systems*, vol. 2017, Article ID 7356213, 14 pages, 2017.

[8] Y. Yin, F. Yu, Y. Xu, L. Yu, and J. Mu, "Network location-aware service recommendation with random walk in cyber-physical systems," *Sensors*, vol. 17, no. 9, p. 2059, 2017.

[9] T. Jena and J. R. Mohanty, "Disaster recovery services in intercloud using genetic algorithm load balancer," *International Journal of Electrical and Computer Engineering*, vol. 6, no. 4, p. 1828, 2016.

[10] M. A. Elaziz, S. Xiong, K. P. N. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowledge-Based Systems*, vol. 169, pp. 39–52, 2019.

[11] Y. Xu, J. K. Li, and K. Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues," *Information Sciences*, vol. 270, pp. 255–287, 2014.

[12] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[13] J.-F. Li and J. Peng, "Task scheduling algorithm based on improved genetic algorithm in cloud computing environment," *Jisuanji Yingyong/Journal of Computer Applications*, vol. 31, no. 1, p. 184186, 2011.

[14] J. Xue, L. Li, S. Zhao, and L. Jiao, "A study of task scheduling based on differential evolution algorithm in cloud computing," in *Proceedings of the International Conference on Computational Intelligence Communication Networks*, Bhopal, India, November 2014.

[15] C. Zhu and J. Ni, "Cloud model-based differential evolution algorithm for optimization problems," in *Proceedings of the Sixth International Conference on Internet Computing for Science Engineering*, Henan, China, April 2012.

[16] Z. Ke, Y. Mao, S. Leng et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, no. 99, pp. 5896–5907, 2016.

[17] B. Keshanchi, A. Souri, and N. J. Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing," *Journal of Systems and Software*, vol. 124, pp. 1–21, 2017.

[18] M. Akbari, R. Hassan, and S. H. Alizadeh, "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems," *Engineering Applications of Artificial Intelligence*, vol. 61, pp. 35–46, 2017.

[19] K. Amin and A. Ghaffari, "Hybrid task scheduling method for cloud computing by genetic and de algorithms," *Wireless Personal Communications*, vol. 97, no. 4, pp. 6301–6323, 2017.

[20] Y. Xiong, S. Huang, M. Wu, J. She, and K. Jiang, "A johnson'-rule- based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing," *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 597–610, 2017.

[21] T. Goyal and A. Agrawal, "Host scheduling algorithm using genetic algorithm in cloud computing environment," *International Journal of Advances in Engineering & Technology*, vol. 1, no. 1, pp. 7–12, 2013.

[22] P. Kumar and A. Verma, "Independent task scheduling in cloud computing by improved genetic algorithm," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 5, pp. 111–114, 2012.

[23] H. Aziza and S. Krichen, "Bi-objective decision support system for task-scheduling based on genetic algorithm in cloud computing," *Computing*, vol. 100, no. 2, pp. 65–91, 2018.

[24] L. Xin-Rong and G. Yang, "Application of catastrophic adaptive genetic algorithm to reactive power optimization of power system," in *Proceedings of the 2010 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 2, pp. 450–454, IEEE, Sanya, China, October 2010.

[25] M. Wang, B. Li, Z. Wang, and X. Xie, "An optimization strategy for evolutionary testing based on cataclysm," in *Proceedings of the Computer Software Applications Coirference Workshops*, Seoul, South Korea, July 2010.

[26] Z. X. Cai, "Application of grey theory in forecasting the diaphania pyloalis cataclysm," *Science of Sericulture*, vol. 35, pp. 869–871, 2009.

[27] S. X. Lv, Yu R. Zeng, and L. Wang, "An effective fruit fly optimization algorithm with hybrid information exchange and its applications," *International Journal of Machine Learning Cybernetics*, vol. 9, no. 10, p. 16231648, 2018.

[28] S. C. Xiao, S. D. Sun, and H. Guo, "Cataclysm genetic algorithm for solving vechicle scheduling problem with time windows," *Application Research of Computers*, vol. 31, no. 12, 2014.

[29] Z. Wang and Y. Chen, "Binary decision diagram variable ordering based on catastrophe genetic algorithm," *Computer Engineering Applications*, vol. 51, no. 3, pp. 55–60, 2015.

[30] A. W. Mohamed, H. Z. Sabry, and T. Abd-Elaziz, "Real parameter optimization by an effective differential evolution algorithm," *Egyptian Informatics Journal*, vol. 14, no. 1, pp. 37–53, 2013.

[31] K. Chandrasekaran and U. Divakarla, *Load Balancing of Virtual Machine Resources in Cloud Using Genetic Algorithm*, National Institute of Technology Karnataka, Surath Ned, Mangalore, Karnataka, 2013.

[32] H. Gao, W. Huang, X. Yang, Y. Duan, and Y. Yin, "Toward service selection for workflow reconfiguration: an interface-based computing solution," *Future Generation Computer Systems*, vol. 87, pp. 298–311, 2018.

[33] H. Gao, D. Chu, Y. Duan, and Y. Yin, "Probabilistic model checking-based service selection method for business process modeling," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 06, pp. 897–923, 2017.

[34] Y. Yin, L. Chen, y. xu, and J. Wan, "Location-aware service recommendation with enhanced probabilistic matrix factorization," *IEEE Access*, vol. 6, pp. 62815–62825, 2018.

[35] S. Pang, T. Ding, A. Rodrfguez-Patdn, T. Song, and Z. Phen, "A parallel bioinspired framework for numerical calculations using enzymatic p system with an enzymatic environment," *IEEE Access*, vol. 6, pp. 65548–65556, 2018.

[36] H. Gao, S. Mao, W. Huang, and X. Yang, "applying probabilistic model checking to financial production risk evaluation and control: a case study of Alibaba's Yu'e Bao," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 3, pp. 785–795, 2018.