Hindawi
Mobile Information Systems
Volume 2022, Article ID 4341953, 9 pages
https://doi.org/10.1155/2022/4341953

*Research Article*

# A Genetic Algorithm for Multiedge Collaborative Computing Offloading Scheme

**Weixin Sui**,[1] **Yimin Zhou**,[2] **Sizheng Zhu**,[1] **Ye Xu**,[1] **Shanshan Wang**,[1] **and Dan Wang**[1]

[1]*Public Experiment Center, University of Shanghai for Science and Technology, Shanghai 200093, China*
[2]*School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, 516 Jungong Road, China*

Correspondence should be addressed to Weixin Sui; farmer@usst.edu.cn

Received 13 May 2022; Accepted 29 June 2022; Published 30 July 2022

Academic Editor: Muhammad Babar

The fast popularization of the Internet of Things (IoT) has caused the data scale to increase geometrically. The data of IoT devices is processed on the cloud, but the way of processing data in the cloud center gradually causes problems, such as communication delay, latency, and privacy leakage. Edge computing sinks some cloud center services to the edge of the device so that data processing is completed in the terminal network, thereby realizing rapid data processing. At the same time, since long-distance communication is avoided, user data is processed locally, so that user privacy data can be safely protected. A genetic algorithm is a type of heuristic algorithm that is based on the genetic development of organisms in nature and has a high global optimization capability. The basic aim and objective of this paper is to study the existing edge computing framework along with computing offloading technology. The genetic algorithm is investigated using multiedge computing-oriented collaborative computing offloading, which is helpful to the IoT's growth as well as the generation and the use of data. The use of a genetic algorithm based on a color graph for load balancing on several edge servers is also investigated. In terms of the study's performance evaluation, it is obvious that our proposed approach produces superior results than previous studies.

## 1. Introduction

As the Internet of Things (IoT) technology and mobile communication technology are rapidly evolving, intelligent mobile terminals such as smartphones, tablet computers, and smart home devices have entered thousands of households, which greatly facilitates people's lives. At the same time, the massive increase in mobile terminals will inevitably bring about a sharp increase in mobile traffic. The rapid development and application of the IoT have resulted in a significant increase in terminal data. The traditional computer architecture, which is based on a centralized data center and the Internet as we know it, is not well adapted to transferring continuously flowing rivers of real-world data. However, the process of transferring data to the cloud for evaluation has been plagued with faults. In one sense, the application scenarios that need high-speed and real-time data processing cannot be met by this method. For example, in the event of an emergency, response measures should be initiated immediately instead of being processed by the cloud center. Data regarding user privacy, on the other hand, that is sent to a cloud center may result in privacy leakage during processing. For instance, the camera must transfer user picture data to a cloud center for processing, which may compromise user privacy. With the innovation of various information services and terminal applications, users have increasingly stringent requirements for network service quality and network request delay.

In order to address these aforementioned challenges, businesses are gradually adopting edge computing architecture. Edge computing can leverage model-driven intelligent capabilities (storage, computing, converged networks, etc.) to enable the actual realization of autonomous and collaborative capabilities in the context of the intelligent distribution architecture and platform of the network edge. Not only that, but some gateways of edge computing can also connect the physical and digital world in the network through the mutual conversion of its network connection and related protocols, giving users more lightweight connection management, quick data analysis, and practical management tasks. Edge computing is the process of utilizing an open platform in order to give the closest end service by integrating application core capabilities on the side close to the object or data source, storage, computing, and network [1]. On the edge, edge computing applications grow increasingly beneficial in terms of rapid responses, application intelligence, privacy protection, and security, as well as addressing the industry's essential demands in real-time business. Edge computing is a type of computer that lies between or on top of physical objects and industrial connections. Edge computing is a critical component of IoT services [2–4]. Centralized cloud computing systems are becoming more incompetent for analyzing and evaluating the massive volumes of data generated from IoT devices because of restricted network performance for data transfer [5]. Preprocessing can considerably minimize the quantity of data that has to be delivered when edge computing pushes computing responsibilities from the centralized cloud to the edge side adjacent to IoT devices.

Edge computing is the process of allocating the central cloud server's processing capacity to edge nodes located close to the user. Edge computing offers two significant advantages over conventional cloud computing. First, edge nodes can preprocess a large amount of data, and then send it to the back-end cloud server. The other is to boost cloud server usage efficiency and optimize cloud assets by endowing edge node devices with computing power [6]. Computing offloading technology was originally proposed in mobile cloud computing. Mobile devices offload computing-intensive tasks to a remote cloud computing point for processing with the help of the core network, making up for the shortcomings of mobile devices on different levels including storage, computing power, and battery life. At the same time, it breaks through the limitations of heterogeneous software and hardware of mobile device terminals. However, because this technique necessitates sending a big quantity of data to a cloud computing center located far away from the mobile device, it consumes a significant amount of network resources and causes a significant request delay [7, 8]. Edge computing "sinks" cloud services to the edge of the network, so that edge servers are closer to mobile terminal devices and can provide nearby computing and storage resources. The processing of some or all of the activities is offloaded to the edge server, which can effectively minimize network resource occupancy and request delay, improving service quality and user experience.

Computing offloading (also called computing migration) [9] refers to the process of reasonably allocating the computing tasks of mobile terminals to remote devices with sufficient resources for processing according to a certain offloading strategy. As one of the most essential aspects of edge computing research, computing offloading is usually described as the incompetence of resource-intensive and delay-sensitive tasks on terminal devices with limited resources and computing power. Part of it is delivered to the cloud computing center or edge server for task computing, thereby enhancing the task processing capability of the mobile terminal device and improving the user experience quality [10, 11]. The computing offloading technology makes up for the terminal device's computing power, storage resources, and energy efficiency. Of course, the edge server has limited computing resources, and when a significant number of offloading requests cannot be satisfied during the high-load period, the 3-layer offloading calculation can also be performed through the joint cloud center [12]. In addition, due to the emergence of edge collaborative applications, when multiple users are in the same proximity, the results of task offloading calculations can be reused [13, 14].

In computing offloading technology, offloading decisions need to consider the influence of network link quality, terminal device performance, edge server performance, and other factors. Specific questions include whether the newly generated task can be offloaded, whether it is partially or completely offloaded, and where to offload to perform the computation. For the question of "partial or full offloading," according to whether the task can be divided, it is decided whether to offload all tasks to the edge server for execution or to offload some tasks and leave the rest for local execution. For the question of "where to unload to perform tasks," the nature of the tasks needs to be considered. The relationship between tasks in the same application can be either parallel or serial, and the sequence of tasks must be calculated based on the order of the tasks.

Through mathematical methodologies and computer simulation activities, a genetic algorithm converts the problem-solving mechanism into a mechanism comparable to the mutation and crossover of chromosomal genes in biological evolution. Better optimization outcomes may generally be attained faster than certain standard optimization techniques when handling more difficult combinatorial optimization challenges. Combinatorial optimization, machine learning, signal processing, adaptive control, and artificial life have all benefited from genetic algorithms.

The basic contributions of this paper are as follows: to study the existing edge computing framework along with computing offloading technology. The genetic algorithm is explored on the basis of multiedge computing-oriented collaborative computing offloading, which is beneficial to the growth of the IoT as well as the creation and usage of data. Also using the genetic algorithm based on the color graph for the load balancing on different edge servers is studied. For the performance evolution of this study, it is crystal clear that our proposed model gives a better result as compared to already existing work.

The remaining paper is organized as follows: Section 2 sheds light on the background information for the construction of a genetic algorithm system for a multiedge collaborative computing offloading scheme. Section 3 is about the proposed method and approaches toward load balancing among different edge servers. Section 4 presents the experimental and application analysis of the proposed method. Section 5 finally concludes the theme of the whole paper.

## 2. Background Information for the Construction of Genetic Algorithm System for Multiedge Collaborative Computing Offloading Scheme

*2.1. Process of Calculating Offload.* According to different indicators, computing offloading can be divided into static offloading and dynamic offloading, full offloading and partial offloading, single-node offloading, and multi-node offloading [15]. However, no matter what computing offloading strategy is adopted, in the environment of edge computing, the computing offloading basically follows the following steps, as shown in Figure 1.

(1) Node search: to offload intensive computing tasks on mobile terminal devices to edge servers, the first step is to search for available edge computing nodes in the edge computing environment. The number of computing tasks offloaded to edge servers can be divided into single-node offloading and multi-node offloading. Single-node offloading only needs to divide the computing task into two parts, which are locally processed on the mobile device and on the edge server respectively. Multi-node offloading needs to consider the load situation, computing power, and network communication status of the mobile terminal device on different nodes.

(2) Task segmentation: task segmentation is to use an appropriate algorithm to divide the tasks that need to be processed and divide them into the local execution part and the part that must be offloaded to the edge server for execution. In general, the locally executed part is generally program code that must be executed locally, such as user interface, program code for processing peripheral devices, and the like. The part executed on the edge server is usually some program code with little interaction with local device information and a large amount of computation.

(3) Unloading decision: the unloading decision is the core part of computing unloading, which determines whether to unload the program and which part of the program to unload to the edge computing node.

(4) Program transmission: the offloaded computing program is transferred to the edge computing nodes, and the timeliness of the transmission depends on the bandwidth conditions of the communication network.

(5) Execute the calculation: the edge computing node performs computation on the programs offloaded to the edge server. Usually, a virtual machine solution is adopted. The edge server will start a virtual machine for the offloaded task and the computing task will reside in the virtual machine and execute until the task operation ends.

(6) Return the operation result: the result after the execution of the calculation processing is returned to the mobile device terminal, so far, the uninstallation process ends.

*2.2. Network Structure of Edge Computing.* Figure 2 shows the typical architecture of the current edge computing network. In the vertical direction, data is transmitted across the wireless network and the core network using cloud-edge-end 3-layer collaborative computing, while distributed horizontal collaborative computing is done amongst devices in the same layer. Cloud computing can provide computing power and service resources on-demand, regardless of the amount of data to be calculated, the time tolerance of tasks, and the quality of network connections. Although the service resources provided by edge computing are limited, compared with remote cloud computing, it can use devices with storage functions and computing distributed on the path of the data source and cloud center to realize data preprocessing and store the preprocessed data. The data is uploaded to the cloud center or the calculation result is returned to the mobile terminal device.

## 3. Proposed Method and Approaches

*3.1. Color and Unicolor Graphs.* In graph theory, graph coloring is the process of allocating colors to the vertices and edges of a graph [16]. In this work, we look at the graph coloring issue, which entails giving colors to vertices in such a way that no two neighboring vertices in a graph have the same color. When assigning colors to vertices, the goal of graph coloring is to use as few colors as possible.

Petersen graphs, a special type of undirected graph with ten vertices and fifteen edges, are shown in Figure 3 (with colors and without colors). In Figure 3, there are two graphs: A and B. The A shows a graph without coloring while B shows graphs with three different colors including blue, green, and dark red. There is one condition that should be satisfied while coloring this graph that no neighbor vertices should include the same color and also by keeping the number of colors as minimum as possible.

In our proposed strategy, we use graph coloring in order to balance the load. By analogy, a graph's vertices may be thought of as terminal devices, and the graph's edges can be thought of as nearby discoverable terminal devices. Edge servers are distinguished by their different graph colors. The goal of this strategy is to spread terminal device workloads over numerous edge servers while reducing the amount of the necessary edge servers. If we assign work from terminal devices that are close together to superfluous edge servers that are geometrically far away from the terminal devices, we meet the real-time condition.

*3.2. GA (Genetic Algorithm).* A genetic algorithm is a kind of bionic optimal strategy for finding the optimal solution set based on the assumption of evolution theory, and its essence is a kind of efficient, parallel, and all-around search method. A genetic algorithm is self-learning, self-adapting, and self-optimizing algorithm based on Darwin's theory of evolution,

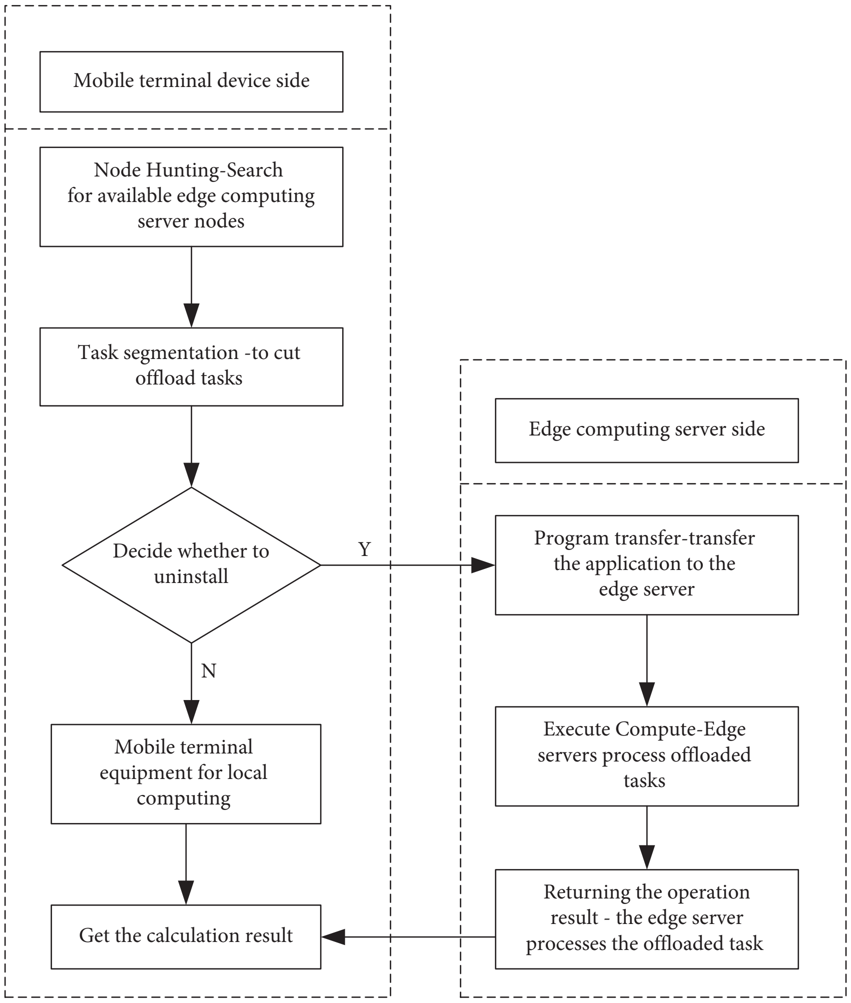
Mobile terminal device side
Node Hunting-Search for available edge computing server nodes
Task segmentation -to cut offload tasks
Decide whether to uninstall
Y
N
Mobile terminal equipment for local computing
Get the calculation result
Edge computing server side
Program transfer-transfer the application to the edge server
Execute Compute-Edge servers process offloaded tasks
Returning the operation result - the edge server processes the offloaded task

Figure 1: Flowchart of computational offloading.

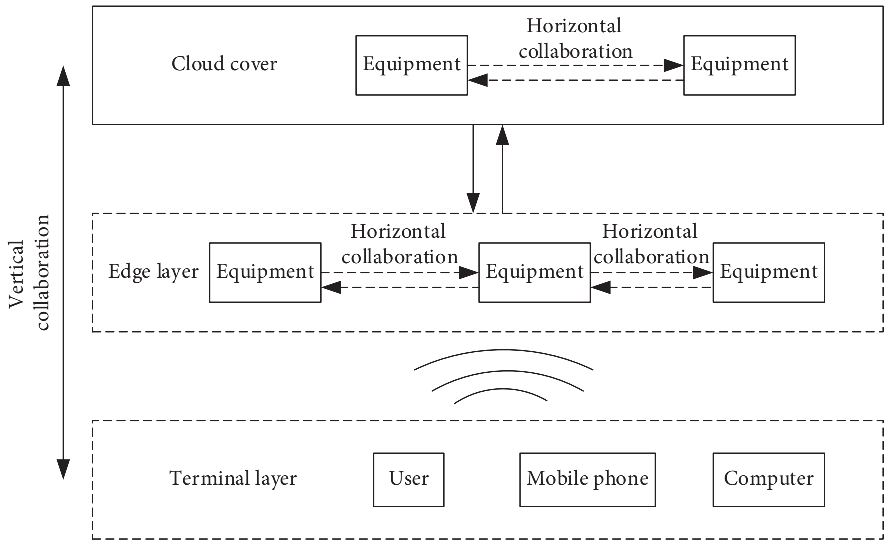

Figure 2: Edge computing network architecture diagram.

which can solve complicated problems [17]. A genetic algorithm's essential idea is [18] that it encodes a problem's solution to chromosomes abstractly in such a way that one chromosome corresponds to one answer. The person is then ranked according to their fitness using an evaluation criterion for each chromosome. People who are physically fit

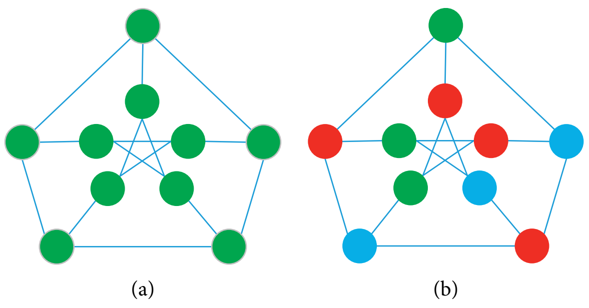

Figure 3: Petersen graphs. (a) Without color. (b) With colors.

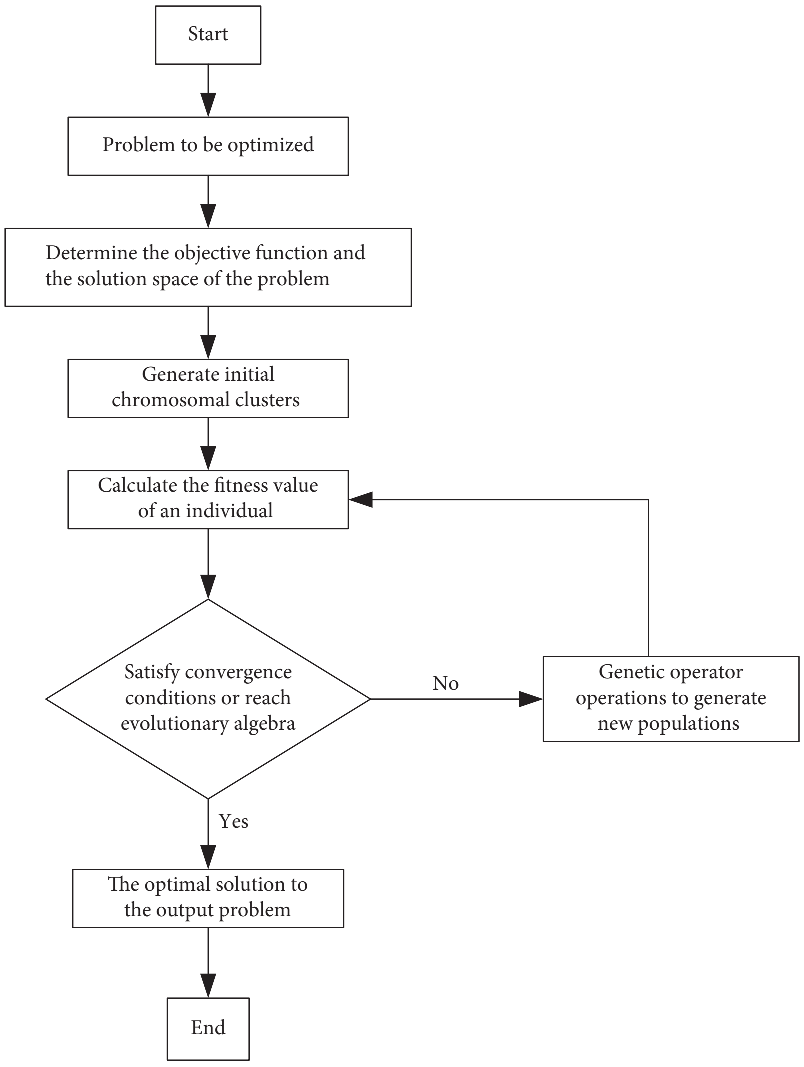
Start
Problem to be optimized
Determine the objective function and the solution space of the problem
Generate initial chromosomal clusters
Calculate the fitness value of an individual
Satisfy convergence conditions or reach evolutionary algebra
No
Genetic operator operations to generate new populations
Yes
The optimal solution to the output problem
End

Figure 4: Basic flow chart of genetic algorithm.

are more likely to get chosen. The next-generation population is chosen by the selection, crossover, mutation, and so on until the fitness of the emerging individuals matches the demands of the algorithm, and the final individual obtained is the optimal solution to the problem we are looking for. Figure 4 depicts the overall flow of the genetic algorithm.

Selection operation, crossover operation, and mutation operation are the three basic processes of a genetic algorithm. Only by scientifically and rationally linking these operation phases and constructing matching operation plans based on the individual issues to be solved, the algorithm's performance can be maximized and the best answer can be obtained fast and correctly.

```
Input to the algorithm: G such that G is the tuple of V and E
Output to the algorithm: A colored graph denoted by $G_{color}$ and equal to ($V_{color}$ E)
total_color ← ran_num
fit_fun ← generate_fit_fun()
cond ← F
max_gen ← retrieve_parameters(gen)
call lifeBegin(total_color)
func lifeBegin(total_color)
  counter = 0
  for $V_i$ = G do
    $V_{i.color}$ ← generateRandomColor(total_color)
  end for
  while (cond = = F and counter ≤ max_gen)
    call colorGraph(G, total_color)
    cond ← checkstopcondition()
    counter ← counter + 1
  end while
  if cond = = true then
    for all $V_i$ = G do
      $V_{i.color}$ ← pickColor(fit_fun, total_color)
    end for
    total_color ← total_color -
    lifeBegin(total_color)
  else
    total_color = total_color + 1
    lifeBegin(total_color)
  end if
end func
func colorGraph(G, total_color)
  evaluateFitness(G, fit_fun)
  selection(G, fit_fun)
  crossover(G, fit_fun)
  mutation(G, fit_fun)
end func
```

Algorithm 1: Balancing distribution.

*3.3. Characteristics of GA.* In the field of natural heuristic algorithms, the genetic algorithm (GA) is a rather well-known method. It was initially proposed in the 1960s by John Holland of the University of Michigan [17]. A genetic algorithm, as the name implies, is an algorithm that is created and executed using the principles of biological evolution and inheritance in nature. With the passage of time and the advancement of science and technology, computer performance has increased significantly and GAs have progressively made their way from the theoretical to the practical realm. GAs have grown in popularity as a framework technique for solving large optimization problems as a result of scholarly study and are now widely employed in domains such as computer science, commerce, and finance.

Scholars have presented numerous sorts of intelligent optimization algorithms for various types of optimization issues in various domains, such as particle swarm optimization and simulated annealing. These algorithms are built on various theoretical foundations and are appropriate for various domains, each with its own set of benefits and drawbacks. The legacy algorithm has the following features as a sophisticated problem-solving algorithm.

First, the GA is a stochastic optimization algorithm. It does not have a lot of mathematical prerequisites for solving the optimization issue. The inherent properties of the problem do not need to be taken into account during the search process because of its evolutionary characteristics such as its ability to directly operate on the structural object, discrete or continuous, linear or nonlinear, and the application scenarios and scope are relatively broad.

The second is that the GA directly uses the objective function as the search heuristics and also the individual is evaluated by using the fitness function [19], without any other complicated derivation or supplementary data, and on this basis, performs genetic operations to realize the individual in the population. Information exchanges between them, so that it is less reliant on problem-solving and has a lot of flexibility.

The GA's third feature is that it uses a multi-point parallel search approach rather than a single-point search, which effectively prevents the search from convergent to the local optimal solution. Simultaneously, it is exactly because of the parallel settlement properties of genetic algorithms that we may increase the algorithm's operating speed via

Table 1: Comparison of distribution balance.

| Number of assignments (times) | The distribution balance of the experimental group | Control group distribution balance |
|---|---|---|
| 3 | 1.33 | 0.58 |
| 6 | 1.40 | 0.73 |
| 9 | 1.37 | 0.61 |
| 12 | 1.40 | 0.52 |
| 15 | 1.61 | 0.65 |
| 18 | 1.35 | 0.57 |
| 21 | 1.42 | 0.62 |
| 24 | 1.63 | 0.51 |
| 27 | 1.41 | 0.60 |
| 30 | 1.27 | 0.66 |

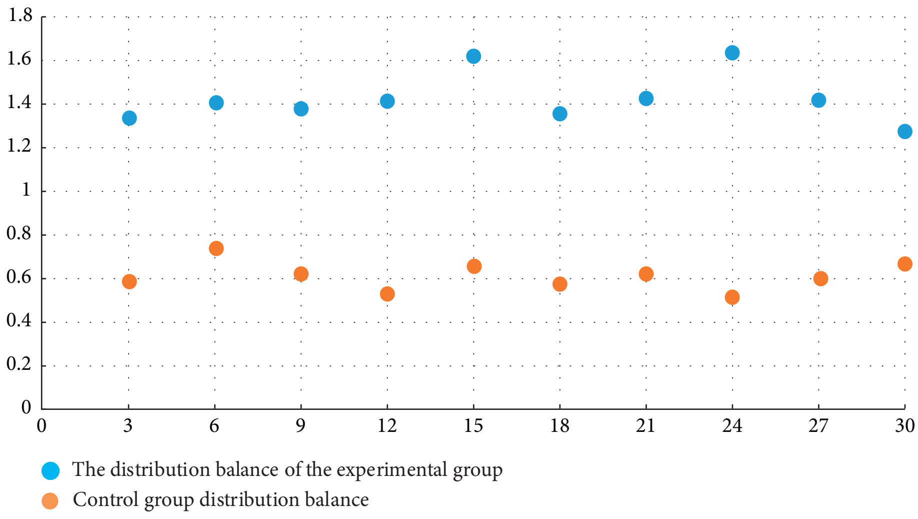

Figure 5: Comparison of distribution balance.

wide-ranging parallel computing, allowing for quick optimization in real-time.

Fourth, rather than working with the parameters themselves, the GA works with the coding of parameters, and its optimization principles are based on chance rather than determinism. In essence, it is a comprehensive framework for solving system optimization difficulties with a lot of space for development, rather than merely an optimization approach.

*3.4. Balancing Distribution.* The distribution balancing strategies in edge cloud computing setting for edge servers are elaborated in Algorithm 1. Here, *G* denotes the graph which is the tuple of vertices and edges and it is input to the algorithm. The output of the algorithm is a color graph which is denoted by $G_{\text{color}}$.

Edge servers also verify whether they are capable of executing the offloaded tasks while installing the suggested load balancing algorithm. If the offloaded tasks are supported by the edge server, task execution is scheduled. Otherwise, it sends the offloaded duties through extra network traffic to the primary cloud server. This strategy avoids task failures that aren't essential.

The proposed approach takes into account the waiting time for edge servers and the forwarding time for shifting work from edge servers to the central cloud server when determining whether to move workloads from edge servers to the central cloud server. The suggested approach does not send offloaded work to the central cloud when the edge server waiting time is smaller than the forwarding time for offloading workloads from edge servers to the central cloud server. Other than that, the offloaded duties are sent to the main cloud server.

Here, in the balancing distribution algorithm, the term ran_num means the random number which assigns to the total_color variable. The term fit_fun means the fitness function and the value of generate_fit_fun() is assigned to it. The value of the cond variable is F which means false.

## 4. Experiment and Application Analysis

Using ECLDEVGSDEF Luna-SR 3 as a development IDE implemented by big data center resource allocation code [20], a large data center is selected as the experimental object, and two real data sets in the large data center are randomly selected as the resources of the big data center, namely, the TYHUSGE data set and CFISNJA data set. To verify the

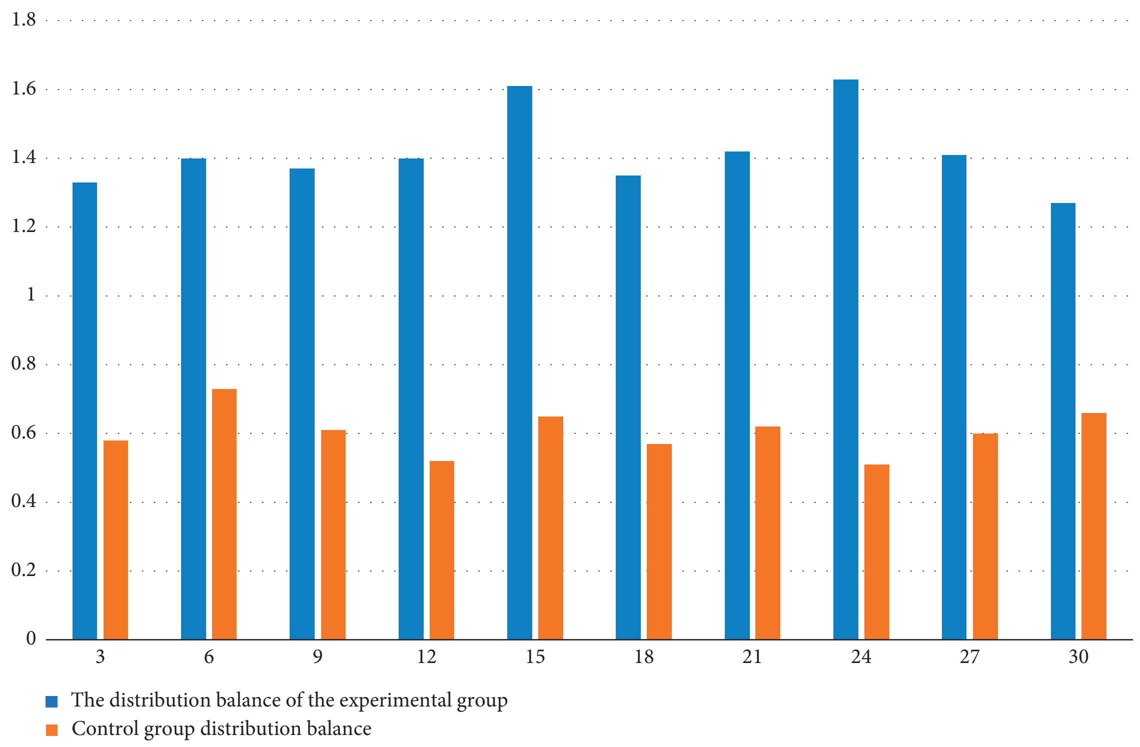

FIGURE 6: Comparing distribution balance of two groups.

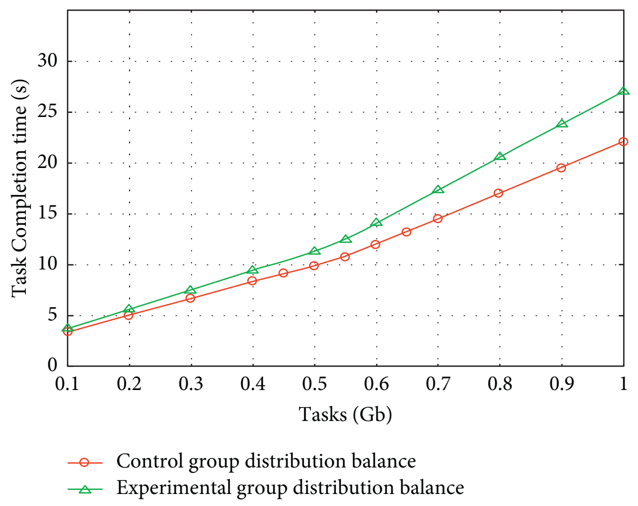

FIGURE 7: Comparing load distribution balance of two groups, experimental and control.

superiority and advancement of the proposed method of this paper, The experimental item is chosen in a circular region with an 8-kilometer side length, which belongs to a wireless network, and 250 communication nodes are set up.

Firstly, the genetic algorithm system based on the multiedge collaborative computing offloading scheme in this paper is used to allocate the resources of the big data center, and the distribution balance is calculated by Kerterly software, which is recorded as the experimental group. Then, the traditional allocation method is used to allocate the resources of the big data center. The distribution equilibrium was measured by Kerterly software and recorded as the control group. The content of the experiment is to test the distribution balance of the two distribution methods. The higher the distribution balance, the higher the distribution rationality of the distribution method. The number of assignments is set to 30 and every 3 times it is used as a recording node to record the experimental results.

Table 1 shows the outcomes of the concluding experiments. The distribution balance of the experimental group is higher than that of the control group, as shown in Table 1. A comparison chart of the experimental data, as shown in Figures 5 and 6, is produced to more intuitively highlight the difference between the two distribution strategies. Figure 5 shows that the distribution balance of the genetic algorithm system for the multiedge collaborative computing offloading scheme designed in this paper is significantly higher than that of the control group, indicating that the genetic algorithm system for the multiedge collaborative computing offloading scheme designed in this paper can achieve a balanced distribution of large data center resources and has practical application value.

Figure 7 compares the load distribution of two groups, namely, the control group and the experimental group. It can be seen from Figure 6 that the distribution balance strategy proposed in this work clearly wine the control group.

## 5. Conclusion

Edge computing has the potential to perform various functionalities such as low latency, high bandwidth, support for massive connections, and information privacy and security protection. Edge computing-based IoT architecture offers substantial benefits in terms of boosting communication security and lowering system resource usage. In this paper, the genetic algorithm based on the multiedge collaborative computing offloading scheme and color graph is studied, and the results show that the algorithm system has the value of promotion. Computing offloading is one of the critical technologies of edge computing, which mainly solves the shortcomings of mobile terminal equipment in terms of computing performance, resource storage, and energy efficiency. Computing offloading decision-making is the key topic of computing offloading technology research, and it primarily focuses on the feasibility and utility of computing offloading to users within the constraints of request time and energy usage. Computing offloading can realize the offloading of computing tasks on mobile devices to edge servers with rich computing resources, thereby improving computing efficiency and reducing the burden on mobile devices. Making good use of genetic algorithms and edge collaborative computing offload is beneficial to our lives and allows us to make better use of the network.

## Data Availability

The datasets used and analyzed during the study are available from the corresponding author upon reasonable request.

## Conflicts of Interest

The authors declare that they have no conflicts of interests.

## References

[1] J. Fu, Z. Shi, and Z. Zhang, "An edge computing framework for digital grid," in *Proceedings of the 2020 IEEE 3rd International Conference on Electronic Information and Communication Technology*, pp. 670–673, (ICEICT), Shenzhen, China, November 2020.

[2] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2017.

[3] Y. Zhang, J. Ren, J. Liu, C. Xu, H. Guo, and Y. Liu, "A survey on emerging computing paradigms for big data," *Chinese Journal of Electronics*, vol. 26, no. 1, pp. 1–12, 2017.

[4] J. Guo, H. Fadlullah, Z. M. Kato, and N. Kato, "Energy consumption minimization for FiWi enhanced lte-A hetnets with UE connection constraint," *IEEE Communications Magazine*, vol. 54, no. 11, pp. 56–62, 2016.

[5] J. Liu, H. Guo, H. Nishiyama, H. Ujikawa, K. Suzuki, and N. Kato, "New perspectives on future smart FiWi networks: scalability, reliability, and energy efficiency," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1045–1072, 2016.

[6] T. X. Hajisami, A. Pandey, P. Pompili, and D. Pompili, "Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[7] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12313–12325, 2018.

[8] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[9] Z. Jianmin, F. Yang, Z. Wu, Z. Zhengkun, and W. Yuwei, "Multi-access edge computing (MEC) and key technologies," *Telecommunications Science*, vol. 35, p. 160, 2019.

[10] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 134–144, 2019.

[11] M. Noreikis, Y. Xiao, and A. Ylä-Jaäiski, "QoS-oriented capacity planning for edge computing," in *Proceedings of the 2017 IEEE International Conference on Communications*, pp. 1–6, (ICC), Paris, france, May 2017.

[12] B. Wu, J. Zeng, L. Ge, X. Su, and Y. Tang, "Energy-latency aware offloading for hierarchical mobile edge computing," *IEEE Access*, vol. 7, pp. 121982–121997, 2019.

[13] H. Liu and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber-wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[14] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: a survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[15] C. V. N. Index, *Cisco Visual Networking index: Global mobile Data Traffic Forecast Update*, pp. C11–C52086, Tianjin, China, 2017.

[16] A. Artacho and J. Francisco, "Rubén Campoy, and Veit Elser. "An enhanced formulation for solving graph coloring problems with the Douglas–Rachford algorithm," *Journal of Global Optimization*, vol. 77, no. 2, pp. 383–403, 2020.

[17] X.-P. Wang and L.-M. Cao, *Genetic Algorithm: Theory, Application and Software Implementation*, pp. 68-69, Xi'an Jiaotong University Press, Xi'an, China, 2002.

[18] X. Meng, J. Bradley, B. Yavuz et al., "Mllib: machine learning in Apache spark," *Journal of Machine Learning Research*, vol. 17, pp. 1235–1241, 2016.

[19] S. V. B. Peddi, *Cloud Computing Frameworks for Food Recognition from Images*, University of Ottawa, Canada, 2015.

[20] X. Chen, L. Cheng, C. Liu et al., "A WOA-based optimization approach for task scheduling in cloud computing systems," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3117–3128, 2020.