International Journal of Scientific Engineering and Science
Volume 6, Issue 7, pp. 1-5, 2022. ISSN (Online): 2456-7361

Research on Task Offloading Strategy Based on Genetic Algorithm

Xiulan Sun[1], Yue Wang[2], Wenzao Li[1,3]
[1]College of Communication Engineering, Chengdu University and Information Technology, Chengdu, Sichuan, China
[2]Educational Informationization and Big Data center, Education Department of Sichuan Province, Chengdu, Sichuan, China
[3]Network and Data Security Key Lab. of Sichuan Pro.University of Electronic Science and Technology of China, Chengdu, Sichuan, China

Abstract— *Existing mobile edge computing research focuses on optimizing latency and energy consumption while ignoring the impact of server load imbalance. In this paper, we consider the terminal device uploads the computing task to the nearest AP, and the AP offloads the task to the MEC server through multi-hop communication. Aiming at the edge server load problem of the system and the extra delay that may be caused by multi-hop communication, we established a multi-objective optimization problem with edge server load balancing and average offloading delay as the optimization goals. We propose an offloading decision based on a genetic algorithm, effectively optimizes the load balancing of task offloading. The effectiveness of the proposed algorithm is verified by a large number of simulations.*

Keywords— *Mobile edge computing, multi-hop communication, edge server load balancing, average offloading delay.*

I. Introduction

With the rise of computing-intensive applications and delay-sensitive applications, in order to overcome the problems of large delay and network congestion that may be caused by mobile cloud computing (MCC), researchers have proposed mobile edge computing (MEC). By studying the offloading strategy of MEC, the offloading delay, system energy consumption, etc. can be reduced, but there are inevitably some problems that need to be resolved. First, due to the uneven distribution of tasks in time and space, it may cause unbalanced load among MEC servers, which may lead to a sharp drop in Quality of Service and Quality of Experience [1,2]. Therefore, it is necessary to solve the problem of load balancing among MEC servers. Thus, it is vital to resolve the problem of load balancing among edge servers. Secondly, in a multi-hop MEC system, during the offloading process, when wireless access point offloads large-scale tasks to the neighboring MEC server, it is necessary to utilize the wireless link between APs for task offloading. But multi-hop communication may bring additional transmission delays [3].

In multi-server and multi-task scenarios, H. Zhang et al. designed an optimization problem with the goal of minimizing the completion time of all tasks [4]. J. Xue et al. established an integrated computing and transmission system with MEC visible light communication (VLC) as the main body, and formulated the overlapping-based low-latency flexible system design as an optimization problem [5]. These efforts in the literature focus on addressing system energy consumption and offload latency, while ignoring the impact and consequences of unbalanced MEC server load.

Intensive user tasks can lead to an unbalanced load among servers, resulting in unreasonable resource utilization, which requires appropriate algorithms to solve this problem. Yu M. et al. [6] and Lu H et al. [7] proposed task offloading strategies based on reinforcement learning and deep reinforcement learning, respectively, to achieve the purpose of optimizing load balancing. In [8] study, Li S. L. et al. achieved the purpose of optimizing the weighted sum of total delay and energy consumption of mobile devices and load balancing among MEC servers through the study of offloading strategies. Above literature consider optimizing load balancing, they do not consider task offloading in an environment where edge server deployment is restricted.

The main contributions of this paper are summarized as follows:

- We consider task offloading in scenarios with limited deployment of edge servers in wireless metropolitan area networks, a multi-objective optimization problem is constructed to optimize the load balancing of edge servers and the average delay of task transmission.
- Transform the scenario of deploying edge servers on some APs in the wireless metropolitan area network into an undirected and unweighted graph, an offloading strategy based on genetic algorithm is proposed.
- The algorithm is simulated using the real base station geographic location data set. The experimental results show that the proposed offloading strategy based on the genetic algorithm has a good effect on the optimization problems.

The rest of this article is organized as follows. The second part elaborates on the system model and problem construction. Detailed solutions are provided in Section 3. Section 4 evaluates the performance of the proposed algorithm based on the simulation results. Section 5 concludes this paper and future work.

II. System Model

In this section, we propose some formal concepts to derive optimization models for offload time and load balancing in edge computing scenarios.

A. Network Model

As shown in figure 1, we consider a multi-hop mobile edge computing scenario consisting of multiple APs and multiple servers. This paper uses a connected undirected graph $G =$

$\{A \cup S, E\}$ to represent the network, A represents the set of wireless APs, $A = \{a_1, a_2,,, a_n\}$, E represents the set of links between base stations; when APs When $a_i (i \in A)$ and $a_j (j \in A)$ links are connected, there is an edge $(i, j) \in E$; S represents the set of edge servers, $S = \{s_1, s_2,,, s_m\}$. where $n = |A|$ and $m = |S|$ respectively represent the number of APs and edge servers, and the value of m must be much smaller than the value of n. Some APs in the network are deployed with edge servers with the same capacity. If an AP is deployed with an edge server, the AP and the server are collectively referred to as edge computing nodes. Each AP receives a certain number of task requests at the same time, where the task request set on AP ai is represented as $T_{(i)} = \{t_1, t_2,,, t_v\}$, each task is indivisible, and d_v represents the data size of task t_v, v_i represents the number of all tasks of AP a_i. h_{ij} represents the number of tasks that AP ai offloads to the server s_j. In this paper, each AP is linked through wireless links with geographically adjacent APs, and the number of links of each AP does not exceed 3. In intelligent edge computing, after a large number of computing tasks are offloaded from mobile devices to nearby base stations, edge computing nodes need to determine how to allocate computing resources for execution[9]. The network's intelligent manager receives the servers' load information from the edge computing nodes and formulates the offloading strategy. Based on the reference information obtained, the intelligent manager designs the most appropriate offloading scheme for each APs task and sends the information back.

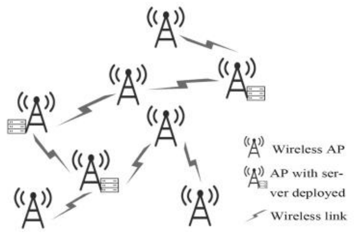

Fig. 1. This is system model.

B. Load Balance Analysis

Due to the uneven distribution of intensive tasks and edge servers, some edge servers may be overloaded, causing network congestion. An important purpose of researching load balancing is to help improve resource usage, to assure that no single node is overloaded, to decrease mobile users' waiting time, and to improve mobile users' experience [9]. By referring to the research literature [10-12], this paper mainly studies the computing load balancing of edge servers in mobile edge computing, mainly balancing the task load that has reached the AP but has not started to execute, and does not consider the communication model of task uploading.

This research uses standard deviation to evaluate the workload balancing of edge servers. We know that m edge servers are located among APs, then we calculate the workload of each edge server j as w_j and the load of server s_j as:

$$w_j = \sum_{i=1}^{k_j} d_i \tag{1}$$

where $k_j = |K_j|$ represents the number of tasks calculated by server s_j, K_j represents the task set calculated by server s_j, $K_j = \{t_1, t_2,,, t_k\}$. The average workload of all edge servers is expressed as follows:

$$w_{ave} = \frac{1}{m} \sum_{j=1}^{m} w_j \tag{2}$$

The standard deviation of the workload can be calculated as follows:

$$w_B = \sqrt{\frac{\sum_{j=1}^{m} (w_j - w_{ave})^2}{m}} \tag{3}$$

It is straightforward to know that the smaller the value of the standard deviation, the more balanced the workload of each edge server.

C. Offloading Time Mode

Offloading time is one of the basic elements of the offloading strategy, and a shorter offloading time can enable users to achieve better service quality. Since the task upload process is not considered in the model in this paper, when the task is offloaded to the edge server for processing, we mainly consider the time when the task is transmitted to the target server through the AP.

The tasks on each AP are offloaded to the edge server for calculation through a wireless link. If the AP is deployed with an edge server, we believe that the received tasks on the AP will have the smallest transmission delay, which is regarded as 0. If the AP is not deployed with an edge server, then the received task of the AP needs to be forwarded to the nearby edge server through the linked adjacent base station for calculation, and the cumulative delay of the multi-hop transmission of each task constitutes the task transmission delay.

The offloading time of the task t_v on the wireless AP a_i is the cumulative delay of the transmission path of the task, namely:

$$t_v^{tran} = \frac{d_v}{\alpha} \times h_c \tag{4}$$

In the formula, d_v is the data size of the task t_v, α is the rate of data transmission between APs, and h_c is the number of links passed in the process of the task being offloaded to the target server through the initial AP. The average transmission delay of all tasks in the mobile edge computing network can be expressed as:

$$t_{ave} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{v_i} t_j^{tran}}{\sum_{i=1}^{n} v_i} \tag{5}$$

Here t_j^{tran} represents the transmission delay of task t_j, and v_i represents the number of task requests received by AP a_i.

D. Problem Formulation

In the research, our focus is to ensure the load balance of edge servers; at the same time, minimize the average offloading delay of all tasks. Then our objective function is articulated as follows:

$$Minmize\ [w_B, t_{ave}] \tag{7}$$

$$s.t.\ \sum_{j=1}^{m} h_{ij} = v_i \quad i \in A, j \in S \tag{8}$$

$$\sum_{j=1}^{m} k_j = \sum_{i=1}^{n} v_i \quad i \in A, j \in S \tag{9}$$

Constraint (7) guarantees that all task requests from AP a_i will be allocated to the server for processing, and no tasks remain unprocessed. Constraint (8) guarantees that all AP tasks are offloaded to the server for calculation.

III. Solving Method

In response to the problems raised in the second part, we propose a task offloading strategy based on genetic algorithm. In order to solve the problem in the edge computing network with restricted server deployment in the paper, the genetic algorithm needs to make some adjustments to the algorithm model, including genes, chromosomes, selection operators and fitness functions.

Chromosome: In this paper, the length of the chromosome is set as the number of APs, and the chromosome code is encoded by an integer. Each gene in the chromosome can be any integer in the server serial number. A chromosome $X = \{x_1, x_2, ,, x_M\}$ means that all tasks are offloaded to the corresponding edge server and its values are randomly initialized. Here is an example: There are 8 APs in the current network, 3 APs are deployed with servers, and the AP numbers where the servers are deployed are one of 1,3, and 7, thus the chromosome length is 8, and each gene in the chromosome is one of 1,3, and 7. Suppose there is a chromosome [1,1,1,7,1,3,3,7], the first, second, third, and fifth execute calculations in the server. At the same time, the transmission path of the task is retrieved by the Dijkstra algorithm. The input of the algorithm is the initial node, the target node and the adjacency matrix of the graph, and the output is the shortest path between the two nodes.

Three operators: (1) Selection operator: we sort all chromosomes according to the fitness value in ascending order, and then select the half population with the smaller fitness value, and add it twice to the parent population to eliminate the other half of the population with a large fitness value. This selection operator can make the algorithm converge quickly and significantly reduce the number of iterations of the population. (2) Crossover operator: The two-point crossover operator is used here, that is, two points are randomly selected in a single chromosome, and then gene exchange is executed. The crossover probability is denoted as C_p, and after extensive preliminary testing and convergence analysis, C_p is set to 0.6. (3) Mutation operator: Uniform mutation is used here, that is, each gene in the chromosome is mutated according to a certain probability. The mutation probability is denoted as M_p. According to preliminary test results, M_p is set to 0.0001.

Fitness function: For a certain chromosome, the fitness value portrays the quality of the task offloading decision expressed by the chromosome, and formula (9) is used as the fitness function f. The smaller the fitness value, the better the fitness of the chromosome. Consequently, our goal is to identify the chromosome with the smallest fitness value during the GA iteration.

$$f = \beta \times \frac{w_B}{w_{B(max)}} + (1-\beta) \times \frac{t_{ave}}{t_{ave(max)}} \qquad (9)$$

In the formula, β and $(1-\beta)$ represent the weights of standard deviation of workload and average offloading delay, respectively. In this paper, let $w_{B(max)}$ and $t_{ave(max)}$ be the w_B maximum and t_{ave} maximum obtained at the first iteration of the algorithm.

At the same time, we added an elite retention strategy to the genetic algorithm to escape the algorithm falling into the local optimum.

IV. Simulation Analysis

This section illustrates how to conduct simulation experiments to determine the effectiveness of our solution, which is simulated using python.

In the simulation of this paper, we randomly selected the actual coordinates of 20 base stations and set the number of servers to 5. We conduct experiments with different total tasks, and the total tasks received by all APs are 3000, 4000, 5000, 6000, 7000 and 8000. The data size of each task is [7,40] *Mbit* [13], and the data transfer rate between APs is 20*MB/s* [14].

In the network scenario, the K-means bisection algorithm is used to divide multiple AP points into m clusters, and the value of m is equal to the number of servers. After clustering, we find the AP closest to the cluster center in each cluster and set this AP as the AP where the server is deployed. After the location of the server is determined, according to the principle of proximity, each AP node is limited to be connected to three AP nodes at most, and the topology map is randomly generated.

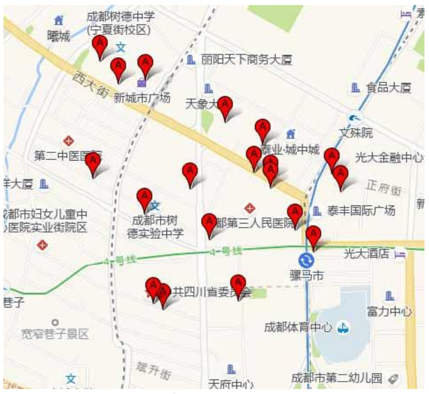

Fig. 2. The real location map of some base stations randomly selected in Jinniu District, Chengdu.

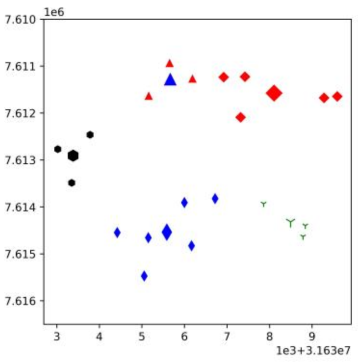

Fig. 3. This is the AP node clustering result

Figure 3 shows the result of the AP location after the bipartite K-means clustering algorithm. Points with the same mark belong to the same cluster, and the greater mark in the cluster represents the cluster center. Figure 4 is a corresponding network topology diagram generated randomly after selecting an AP node to deploy a server according to the clustering result. The small dots in the picture represent APs, the numbers on them represent the serial numbers of APs, and the triangle-shaped dots represent APs with deployed servers.

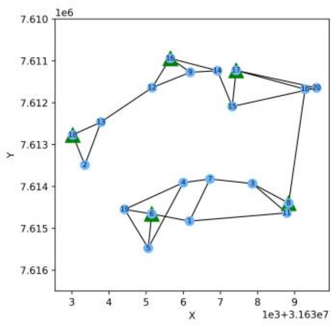

Fig. 4. This is the network topology map corresponding to the clustering result.

In the simulation experiments, we primarily compared the results of the following algorithms. Selecting the nearest server algorithm (SNSA): Each AP finds the server with the least number of hops in the task offloading process according to the Dijkstra algorithm, and all tasks received by each AP are offloaded to this server. (2) Genetic-based optimization algorithm (GA): The details of the algorithm have been outlined in the previous section. In this section, we mainly observe the situation where the workload standard deviation is more important, and the workload standard deviation and the average transmission delay are equally important. Therefore, we select the cases where the weight β of the standard deviation of the workload is 0.7 and 0.5, respectively, and observe the results of the algorithm.

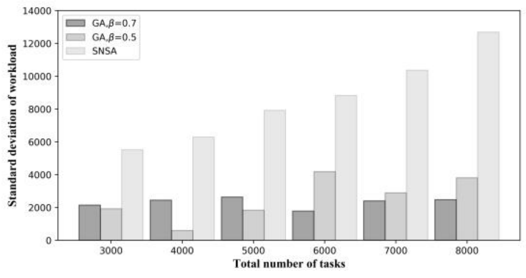

Fig. 5. This is the workload standard deviations value of various algorithm offloading strategies.

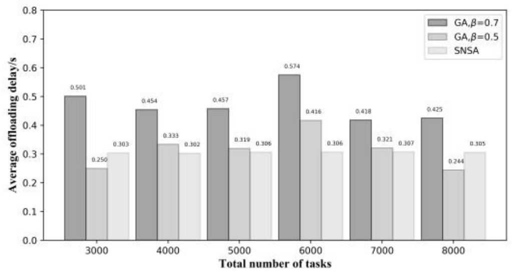

Fig. 6. This is the average offloading delay value of various algorithm offloading strategies.

We draw a comparison of the results of offloading strategies obtained by various algorithms under different total number of tasks. Figures 5 and 6 are comparison graphs of workload standard deviation values and average offload delay values, respectively.

Looking at Figure 5, we notice that the genetic algorithm can get the offloading strategy with the lowest standard deviation of the workload. When the total number of tasks is less than 5000, the genetic algorithm with β of 0.5 obtains the lowest standard deviation of the workload; when the total number of tasks is more than 5,000, the genetic algorithm with β of 0.7 obtains the lowest standard deviation of the workload. We infer that this is related to the larger number of tasks and data volumes on the AP. Under different total number of tasks, the standard deviation of the workload obtained by the genetic algorithm with β of 0.5 is 52%-76% lower than that of selecting the nearest server algorithm; the standard deviation of the workload obtained by the genetic algorithm with β of 0.7 is 61%-80% lower than that of selecting the nearest server algorithm.

Looking at Figure 6, the average offloading delay values obtained by each algorithm are relatively small: the values obtained by these algorithms are around 300ms-500ms. We infer that it has a lot to do with the task offloading path in the algorithm that selects the shortest path. In most cases, selecting the nearest server algorithm obtains the lowest average offloading delay, and the average offloading delay values obtained by the genetic algorithm with B of 0.5 and B of 0.7 are 4%-20% and 26%-40% higher than those obtained by the selecting the nearest server algorithm, respectively. But this strategy resulted in the highest number of workload standard deviations, with a very imbalance workload per edge server.

Consequently, we can conclude that the genetic algorithm has a good effect in optimizing the workload standard deviation.

V. CONCLUSION

In this paper, we in a network with limited deployment of edge servers, utilize multi-hop communication to offload tasks to edge servers, while optimizing server workload and offloading latency. After constructing the multi-objective optimization problem, an offloading strategy based on genetic algorithm is proposed to solve the problem. Simulation results show that the proposed algorithm can adjust the weight of workload and offload delay according to the experimental needs. Subsequent work will conduct a more careful study of server location determination and offloading strategies.

ACKNOWLEDGMENT

We thank all the reviewers and editors who have contributed to the quality of this paper. At the same time, we also appreciate the support by the fund from the Network and Data Security Key Laboratory of Sichuan Province, UESTC (NO. NDS2021-7), Sichuan Province General Education Scientific Research (NO.2019514).

REFERENCES

[1] Guo, H., Liu, J., & Zhang, J. Computation offloading for multi-access mobile edge computing in ultra-dense networks. *IEEE Communications Magazine*, vol. 56, issue 8, pp. 14–19, 2018.

[2] Fan, Q., & Ansari, N. Towards workload balancing in fog computing empowered IoT. *IEEE Transactions on Network Science and Engineering*, vol. 7, issue 1, pp. 253–262, 2018.

[3] Al-Abiad, M. S., Hassan, Z., & Hossain, J.Task Offloading Optimization in NOMA-Enabled Multi-hop Mobile Edge Computing System Using Conflict Graph. *arXiv e-prints*, vol. arXiv-2104, 2021.

[4] Zhang, H., Yang, Y., Huang, X., Fang, C., & Zhang, P. Ultra-low latency multi-task offloading in mobile edge computing. *IEEE Access*, vol. 9, pp. 32569-32581,2021.

[5] Xue, J., Ye, Z., Zhang, H., & Zhu, Y. Flexible Design of Low-Delay MEC-VLC Integrating Network Based on Attocell Overlap for IIoT. *Electronics*, vol. 11, issue 6, pp. 924,2022.

[6] Li, S. L., Du, J. B., Zhai, D. S., Chu, X. L., & Yu, F. R. Task offloading, load balancing, and resource allocation in MEC networks. *IET Communications*, vol. 14, issue 9, pp. 1451-1458, 2020.

[7] Yu M., Tang J., & Li J. A Multi-node MEC Computing Resource Allocation Scheme Based on Reinforcement Learning [J]. *Communication Technology*, vol. 12, 2019.

[8] Lu, H., Gu, C., Luo, F., Ding, W., & Liu, X. Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning. *Future Generation Computer Systems*, vol. 102, pp. 847-861, 2020.

[9] Xu, X., Li, Y., Huang, T., Xue, Y., Peng, K., Qi, L., & Dou, W. An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks. *Journal of Network and Computer Applications*, vol. 133, pp. 75-85, 2019.

[10] Mondal, S., Das, G., & Wong, E. A game-theoretic approach for non-cooperative load balancing among competing cloudlets. *IEEE Open Journal of the Communications Society*, vol. 1, pp. 226-241, 2020.

[11] He, J., Zhang, D., Zhou, Y., & Zhang, Y. (2019, August). An Online Computation Offloading Mechanism for Mobile Edge Computing in Ultra-Dense Small Cell Networks. In *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, pp. 826-833, 2020.

[12] Dong, Y., Xu, G., Ding, Y., Meng, X., & Zhao, J. (2019). A 'joint-me' task deployment strategy for load balancing in edge computing. *IEEE Access*, vol. 7, pp. 99658-99669, 2019.

[13] You, Q., & Tang, B. Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things. *Journal of Cloud Computing*, vol. 10, issue 1, pp. 1-11, 2021.

[14] Fan, W., Liu, Y. A., Tang, B., Wu, F., & Wang, Z. Computation offloading based on cooperations of mobile edge computing-enabled base stations. *IEEE Access*, vol. 6, pp. 22622-22633, 2017.