Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 1059524, 8 pages
https://doi.org/10.1155/2020/1059524

Research Article

Task Offloading Strategy of 6G Heterogeneous Edge-Cloud Computing Model considering Mass Customization Mode Collaborative Manufacturing Environment

Hang Zhou,[1] Yong Xiang ,[1] Hao-Feng Li,[2] and Rong Yuan[3,4]

[1]*School of Computer Engineering, Chengdu Technological University, Chengdu 611731, China*
[2]*Department of Computer Science, Tianfu New Area General Aviation Profession Academy, Chengdu 611731, Sichuan, China*
[3]*School of Mechanical Engineering, Chengdu University, Chengdu, China*
[4]*College of Architecture and Environment, Sichuan University, Chengdu 610017, China*

Correspondence should be addressed to Yong Xiang; zhouchcdsc@163.com

Received 21 July 2020; Accepted 5 September 2020; Published 15 September 2020

Academic Editor: José A. F. O. Correia

With the continuous integration of cloud computing, edge computing, and Internet of things (IoT), various mobile applications will emerge in future 6G network. Driven by real-time response and low energy consumption requirements, mobile edge-cloud computing (MECC) will play an important role to improve user experience and reduce costs. However, due to the complexity of applications, the computing capacity of devices cannot meet the low-latency and low energy consumption requirement. Meanwhile, subject to the limited supplement of power and energy system, the heterogeneous multilayer mobile edge-cloud computing (HetMECC) is proposed to join cloud server, edge server, and terminal devices for data calculation and transmission. By dividing computing tasks, terminal applications can receive reliable and efficient computing services. The simulation results show that the proposed model can achieve the low-latency requirement of data calculation and transmission and improve the robustness of architecture.

1. Introduction

The 5G communication technology has three important application prospects: enhanced mobile broadband (EMB), massive machine-type communication (EMTC), and ultrareliable and low latency communication (URLLC) [1].

EMTC can exchange information based on large-scale data between machines without human intervention. In the future 6G era, massive multisource data will occupy a lot of network bandwidth resources and call for more reliable transmission and real-time processing; this is a challenge to the power and energy system and network system. In engineering production, real-time data acquisition and processing is important for reducing the loss caused by failure. The cdn-p2p system based on Hadoop was proposed by Shaikh et al. [2]. However, this method occupies a lot of network bandwidth and computing resources. Georgakopoulos et al. [3] integrated the edge device into the data center. However, there is a long transmission delay to upload the task to the data center. Subramanya et al. [4] put forward MEC, which is a practical edge computing architecture. However, the computing power of the edge server is limited compared with that of the cloud computing center, so it is difficult to process computing intensive tasks.

It is complicated to fulfill business requirements such as abnormal judgment, emergency scheduling optimization, and overall joint control [5–8]. So, intelligent prediction is needed to grasp the completion of tasks [9–22].

The data involved in the process management and task scheduling present the multisource and heterogeneous characteristics. So, it is necessary to achieve multisource heterogeneous data fusion [23–26]. Time series database (TSDB) is a database for storing time series data. It supports the functions of fast data writing and multidimensional

aggregation query. However, it still has the problems of high cost and energy consumption [27].

The HetMECC model based on heterogeneous multilayer edge computing is proposed in this paper. It combines cloud computing with multilayer edge computing to upload unhandled tasks from the lower edge server to the upper edge server. According to data generation rates, the robustness of the system is analyzed from the perspective of computing resources and transmission resources. The computing power can be fully used to avoid network congestion and reduce the system latency.

2. Proposed Methodology

An analytical methodology is proposed to analyze system latency of the HetMECC model, as shown in Figure 1. The dynamic analysis process is shown in Figure 2: first of all, the raw data are obtained through the monitoring of the IOT technology in application scenarios. Secondly, the task offloading model of HetMECC network processes the input dynamic raw data through computation of the system latency method, computation of energy consumption method, and fitness computing method. Finally, PSO algorithm is used for optimized the task offloading plan. Task completion and robust analysis of HetMECC network are given in case study.

The main devices in HetMECC model are classified into three categories: cloud computing center, edge server, and edge device. The servers are sorted form the upper layer (cloud computing server was defined as layer 0) to the lower layer (defined as layer 1 to layer n). Edge devices are located in the lowest layer (defined as layer $n+1$). The variable $M_n(i)$ denotes the number of devices connecting to the device i in layer n.

2.1. Computation Task Offloading Model. Computation task offloading strategy was classified into three types: local computation, edge computation task offloading, and multilayers computation task offloading, as shown in Figures 3–5. The number (1–5) denotes the allocated computation tasks. These tasks can be processed by edge devices, edge server, or cloud server.

The edge device is located in the lowest layer in the HetMECC model, such as multifunction sensors, computer numerical control machine (CNC), and other smart communication devices. The device i is taken as example in Figure 3. It can process the raw data generated by itself and transfer computation result to cloud server. $v_{n+1}(i)$ denotes the raw data generation rate of edge device i in layer $n+1$. $b_{n+1}(i)$ denotes the number of local computation cycles. $Q_{n+1}(i)$ denotes quantities of computation resources for device i. $\theta_{n+1}(j,i)$ denotes quantities of the allocated transmission resources for device i from edge server j.

In Figure 4, computation tasks were transferred to edge server j; allocated transmission resources $\theta_{n+1}(j,i)$ and raw data generation rate $v_{n+1}(i)$ were considered for the edge device i.

In Figure 5, the computation task was offloaded by the multilayer scheduling method. The computation result of current layer and other layers, some unprocessed raw data, should be taken into consideration. $v_n(i,j)$ denotes raw data arrived rate of edge server j from edge device i in layer N. $Q_n(j)$ denotes quantities of computation resources for edge server j. $\theta_n(k,j)$ denotes quantities of the allocated transmission resources for edge server j from upper server k.

2.2. Computation of System Latency Method. In the HetMECC model, the system latency consisted of computation time, raw data transmission time, and temporary results receiving/sending time. It is assumed as follows:

(1) All computation tasks can be divided

(2) The quantity of allocated computation tasks will not exceed maximum computation capacity of devices

(3) All allocated transmission tasks can be completed

The latency $T_n(i)$ of device i in layer n can be calculated by

$$T_n(i) = \frac{s_n(i) * b_n(i)}{Q_n(i)} + \frac{\rho * s_n(i) * v_n(i) + [1 - s_n(i)] * v_n(i) + \beta_n(i)}{\theta_n(j)}, \tag{1}$$

where $s_n(i)$ denotes task offloading ratio, ρ denotes the data volume compression ratio, $\rho * s_n(i) * v_n(i)$ denotes the quantity of raw data for transmission, $[1 - s_n(i)] * v_n(i)$ denotes the quantity of processed data in layer n, and $\beta_n(i)$ denotes the quantity of temporary results received.

For any device in layer n, $T_{\text{edge}}(n)$ can be calculated by

$$T_{\text{edge}}(n) = \sum_{j=1}^{n-1} \sum_{i \in M_n(j)} T_n(i), \tag{2}$$

where $T_{\text{edge}}(n)$ denotes the total system latency of edge servers and edge devices in layer n.

Assume the computation time T_{server} of cloud computing server is known. The total system latency of the Het MECC model can be calculated by

$$T = T_{\text{server}} + \sum_{n=1}^{n+1} T_{\text{edge}}(n). \tag{3}$$

2.3. Computation of Energy Consumption Method. The energy consumption of devices is classified as follows:

(1) Tasks were all offloaded to edge device. $P_{\text{device}}(i)$ denotes the power of edge device i in layer $n+1$. The energy consumption $E_{\text{device}}(i)$ can be calculated by

$$E_{\text{device}}(i) = \frac{s_{n+1}(i) * b_{n+1}(i)}{Q_{n+1}(i)} * P_{\text{device}}(i). \tag{4}$$

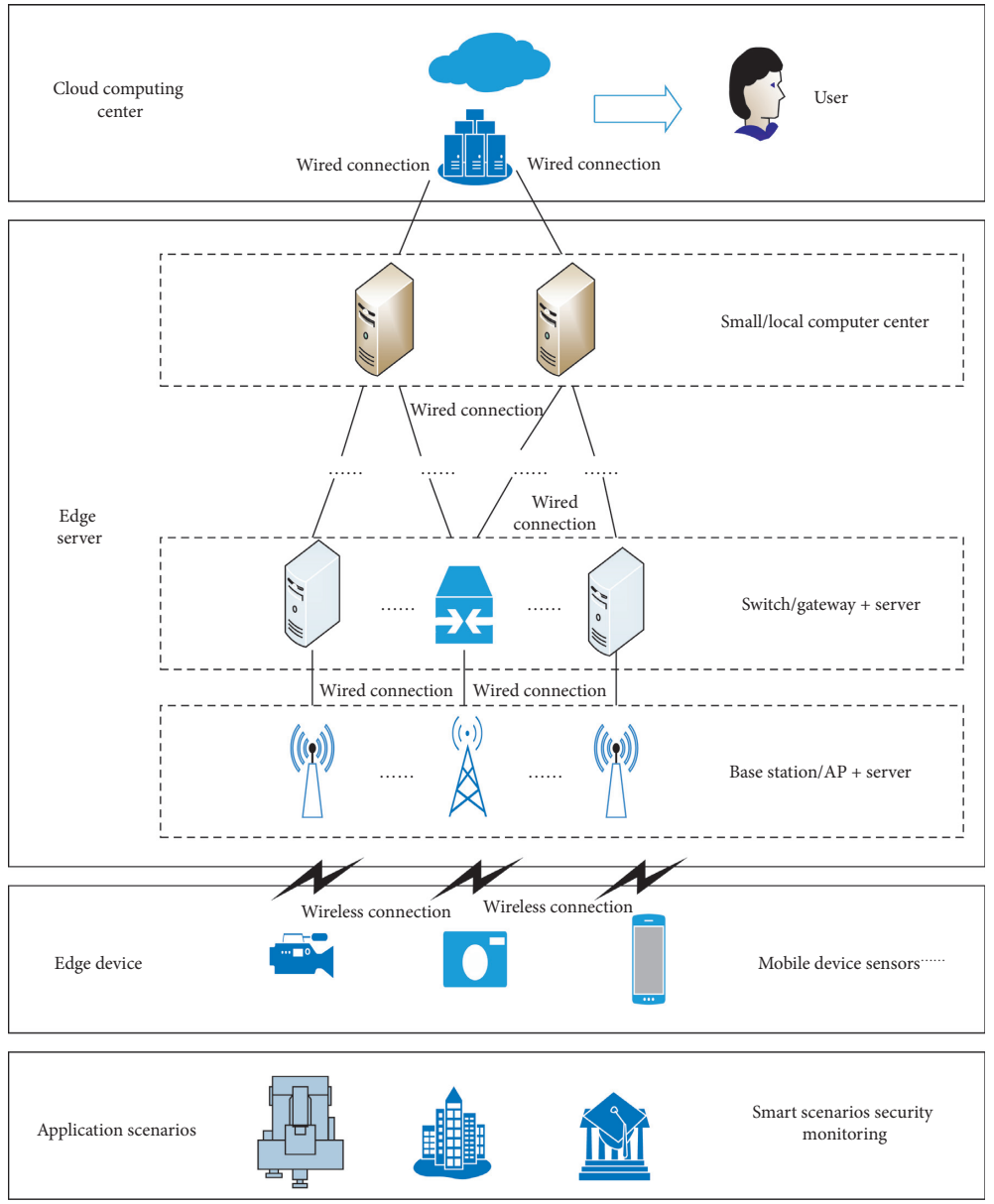

Figure 1: Heterogeneous multilayer edge-cloud computing network architecture.

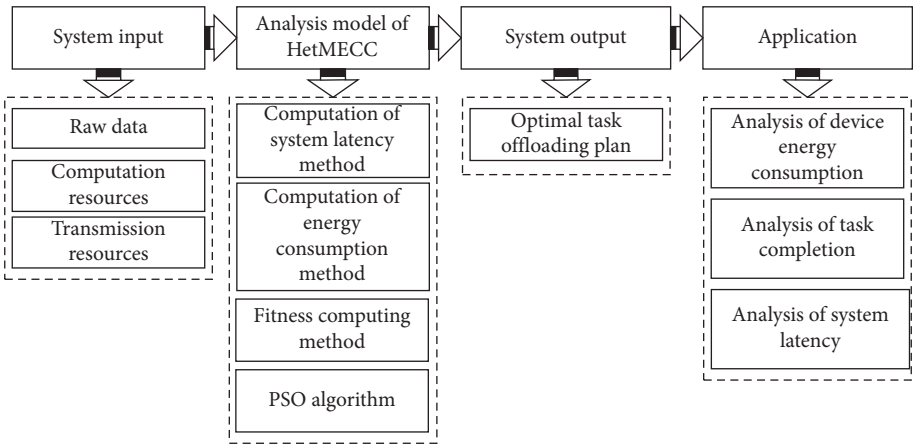

Figure 2: Dynamic analysis process of the HetMECC model.

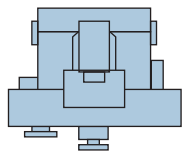

Figure 3: Local computation of edge device.

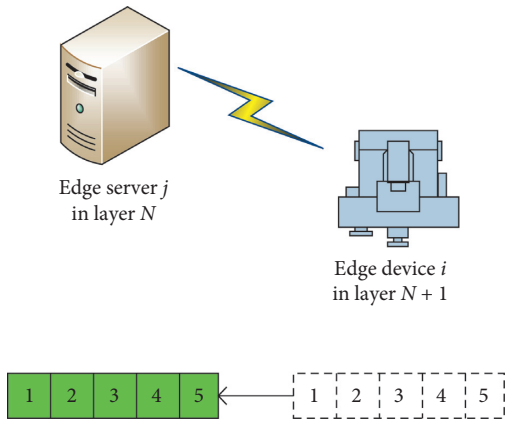

Figure 4: Edge computation task offloading.

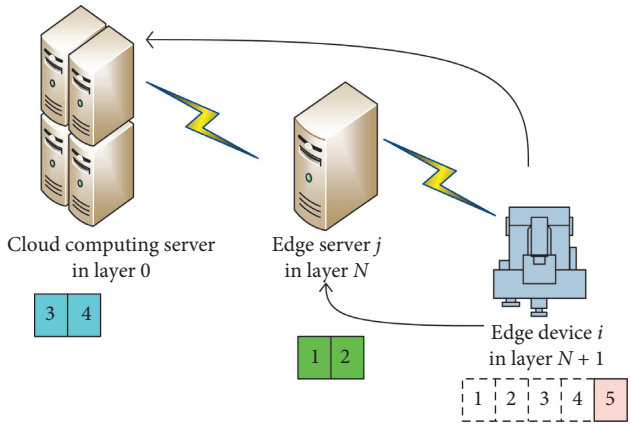

Figure 5: Multilayers computation task offloading.

The total energy consumption E_{device} can be calculated by

$$E_{\text{device}} = \sum_{i=1}^{n} E_{\text{device}}(i). \tag{5}$$

(2) Tasks were offloaded only to multilayer servers, such as edge servers and cloud computing servers. So, the energy consumption consisted of many parts and can be calculated by

$$E_{\text{edgeserver}}(i) = \frac{\rho * s_n(i) * v_n(i)}{\theta_n(j)} * P_{\text{send}}(i) + \frac{[1 - s_n(i)] * v_n(i)}{\theta_n(j)} * P_{\text{transmission}}(i) + \frac{\beta_n(i)}{\theta_n(j)} * P_{\text{receive}}(i). \tag{6}$$

The total energy consumption of edge servers can be calculated by

$$E_{\text{edgeserver}} = \sum_{j=1}^{n-1} \sum_{i \in M_n(j)} E_{\text{edgeserver}}(i). \tag{7}$$

The total energy consumption of cloud computing servers can be calculated by

$$E_{\text{cloudserver}} = \sum_{j=1}^{n-1} \sum_{i \in M_n(j)} E_{\text{cloudserver}}(i). \tag{8}$$

2.4. Fitness Computing Method. The traditional qualitative security analysis cannot provide sufficient information in application scenarios. This paper proposed a quantitative security analysis model. It is used as the quality evaluation index of the task offloading model. The safety factor of the device is defined as $F(i)$, and the task is offloaded only once. The safety factor F_{sum} of the whole task offloading model is derived as follows:

$$F_{\text{sum}} = \sum_{i=1}^{n} F(i). \tag{9}$$

where $F(i)$ denotes the safety factor and the value of $F(i)$ ranges from (0, 1]. Tasks were offloaded to edge devices or cloud computing servers when the value is 1. Otherwise, tasks were offloaded to edge servers. So, the sum of $F(i)$ should be larger in the task offloading model. This paper proposed a fitness function to evaluate the task offloading model under the time constraint [10, 15] as follows:

$$\text{fitness} = \frac{p1 * E_{\text{sum}} + p2 * 10 * E_{\text{sum}} * (T_{\text{sum}}/T_c)}{F_{\text{sum}}}, \tag{10}$$

where $p1$ and $p2$ denote device type parameters under determined task offloading strategy, E_{sum} denotes the total energy consumption, T_{sum} denotes the total system latency, and T_c denotes the time constraint.

3. TOMO Algorithm Design

Based on the HetMECC model, the task offloading model optimization (TOMO) algorithm and the particle swarm optimization (PSO) algorithm [28] are used to optimize the task offloading plan and reduce the conflict probabilities of network resource. TOMO algorithm is shown in Table 1.

Table 1: TOMO algorithm.

Algorithm: TOMO
Input: maximum number of iterations Iter, task number T, time constraint T_c, cloud computing server CCs, edge servers ESs, edge devices EDs
Output: optimal offloading scheme p_{best}
(1) for $i = 1$ to k do
(2) random initialization of device type parameters pi and particle speed v_i;
(3) setting initialization pi as optimal device type parameters;
(4) end for
(5) for $t = 1$ to Iter do
(6) for $i = 1$ to k do
(7) update data type parameters pi and particle speed v_i;
(8) in the process of execution, multiple tasks request the same resource at the same time, and the Multiple layer resources optimal algorithm is used to reduce the conflict;
(9) calculate total system latency T_{sum} considering device type parameter pi;
(10) calculate total energy consumption E_{sum} considering device type parameter pi;
(11) calculate total safety factor F_{sum} considering device type parameter pi;
(12) calculate fitness value according equation (10);
(13) if (fitness(pi) < fitness(p_{best})) then
(14) setting pi as the best offloading scheme p_{best};
(15) end if
(16) end for
(17) The offloading scheme with the lowest fitness value is selected as the global optimal offloading scheme p_{best};
(18) end for
(19) return value of p_{best}

Table 2: Configuration of computation resources and transmission resources of HetMECC.

Node type	Computation resources (Mcps)	Transmission resources (Mbit·s^{-1})
Edge device	0.13	Null
Lowest edge server	0.4	1.6
Midedge server	1.7	4.2
Top edge server	5.3	6.7
Cloud computing server	20	20

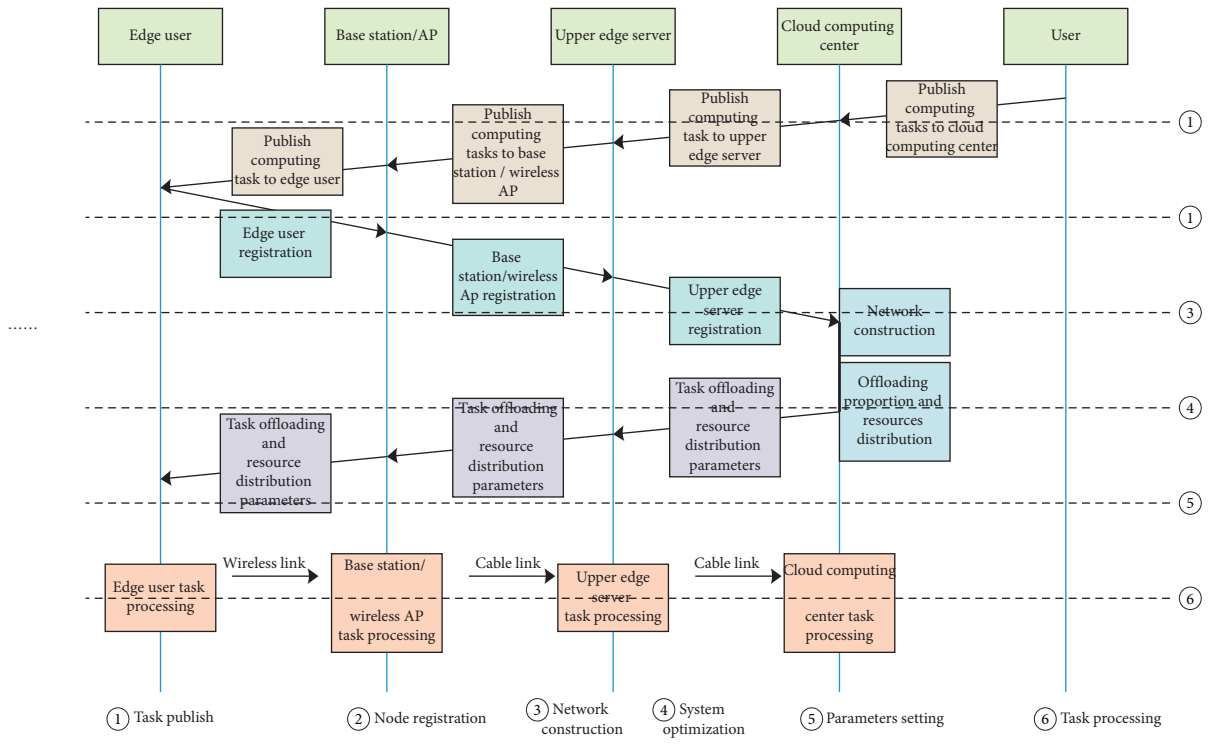

Figure 6: Simulation sequence diagram.

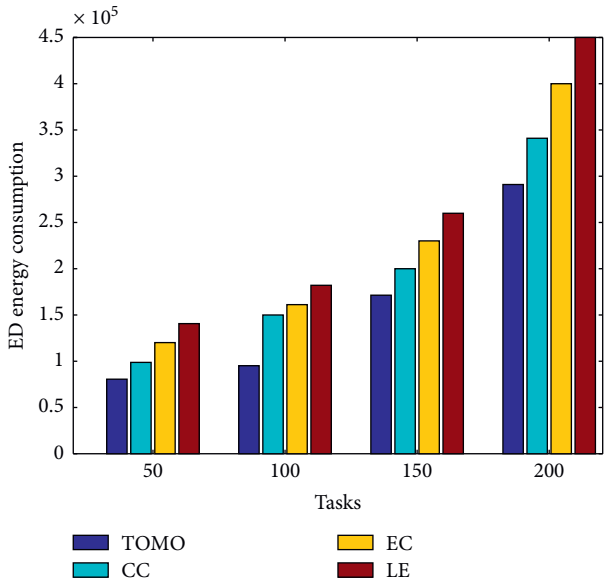

Figure 7: End device energy consumption.

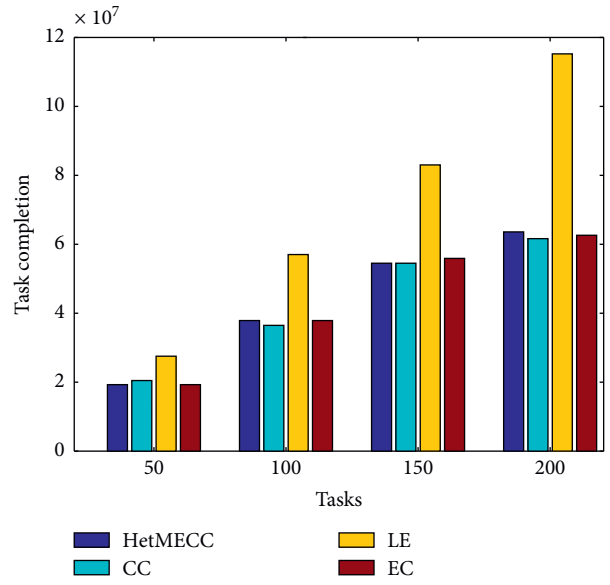

Figure 8: Task completion time comparison.

4. Case Study

4.1. Simulation Environment and Parameter Setting. Computer hardware includes enhanced processor and bulk memory. MATLAB is used for simulation software. In the experiment, each layer publishes 50–200 tasks. The task load is a random value that follows normal distribution. The allocation of computing resources and transmission resources is shown in Table 2. Assume that data transmission bandwidth is 30 Gbps and 3000 Gbps in LAN and WAN. The time constraint is set to 20%~100% of the mean task completion time on 3.6 GHz CPU, and the value of node safety factor $F(i)$ is (0.5, 1].

The sequence diagram of simulation experiment for HetMECC model is shown as Figure 6.

4.2. Experimental Results and Analysis. The task offloading strategy TOMO algorithm is compared with local execution (LE), edge server execution (EC), and cloud computing center execution (CC) from aspects of end device energy consumption, task completion, system latency, fitness value, and robustness analysis. Then, the comparison result is obtained.

The end device energy consumption of the four task offloading models is shown in Figure 7. LE has the highest energy consumption value. The reason why EC has higher energy consumption is that all tasks are offloaded and executed in the edge server node. In the HetMECC model, some tasks can be executed in the cloud. The execution ability of the edge server is weaker than that of the cloud server. So, the energy consumption in CC increased.

The task completion time of the four task offloading models is shown in Figure 8. The completion time of LE is the highest and exceeds the time constraint due to the low execution speed of end devices. When the number of published tasks is 50 in each layer, the completion time of HetMECC is lower than EC. When the number of published tasks is 200 in each layer, the completion time of CC is smaller than HetMECC and EC. But, the completion time gap is small.

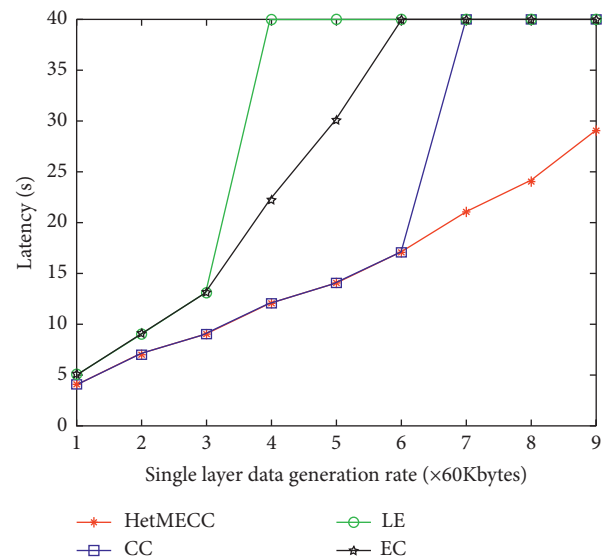

Figure 9: System latency comparison of single layer.

The system latency comparison is shown in Figures 9 and 10. The TOMO algorithm based on the HetMECC model can reduce the system latency more efficiently. It is useful in the case of high data generation rate under high computing pressure. When the data generation rate $v_{n+1}(i)$ is greater than 6, other task offloading models will have network congestion. But, TOMO can reduce the system latency by using multilayer edge servers and cloud computing centers for calculation and transmission. When the data generation rate of double layers in HetMECC network has increased to

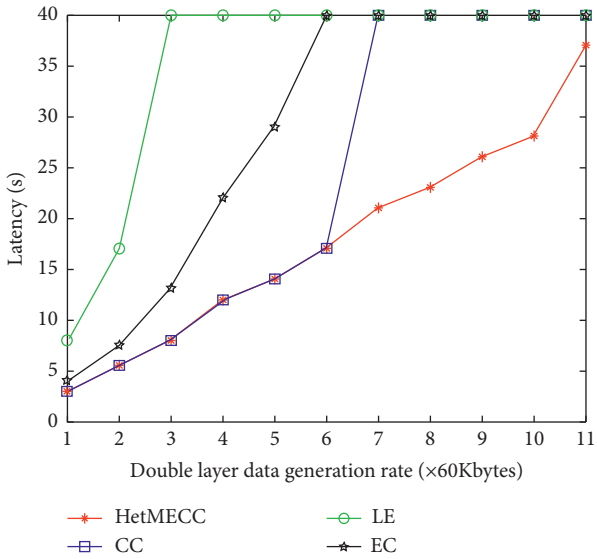

Figure 10: System latency comparison of double layers.

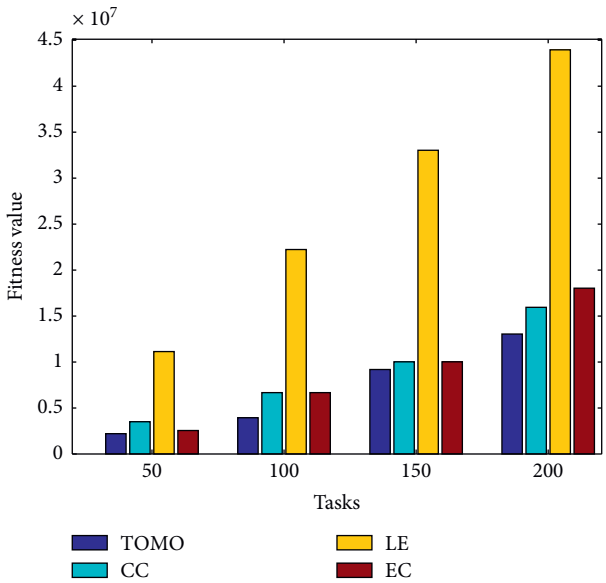

Figure 11: Fitness value comparison.

11, TOMO can also has better performance. It is indicated that the network robustness has been improved significantly.

The fitness values of the four offloading models are shown in Figure 11. When the number of published tasks ranged from 50 to 200, the fitness value obtained by TOMO is the lowest. When the number of published tasks is 50, the fitness value of TOMO is 39.2% lower than CC and 17.6% lower than EC. When the number of published tasks is 200, the fitness value of TOMO was 7.4% lower than CC and 14.3% lower than EC.

So, the task offloading model proposed in this paper can execute the task more efficiently. Terminal devices' energy consumption, task completion time, system latency, and network robustness have been optimized.

5. Conclusion

In this paper, the fitness calculation method is improved based on the HetMECC mode considering the requirement of low-latency and low energy consumption in future 6G heterogeneous network. Taking the energy consumption and safety factor of terminal devices as the evaluation index, system latency computation equation and TOMO algorithm were proposed to optimize the model. The energy consumption, task completion time, system latency, and network robustness are optimized according to the experiment result.

Data Availability

The data used to support the study are available within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Key Research Development Plan Funds under Grant 2018YFB1702804 and the Applied Basic Research Programs of Sichuan Province's Science and Technology Department under Grant 2019YJ0666.

References

[1] P. F. Wang, B. Y. Di, L. Y. Song, and Z. Han, "6G heterogeneous edge computing," *Chinese Journal on Internet of Things*, vol. 4, no. 1, pp. 121–130, 2020.

[2] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNS-based server selection," in *Proceedings of the 20th Annual Joint Conference of the IEEE-Computer-Society/IEEE-Communication-Society*, pp. 1801–1810, Anchorage, American, April 2001.

[3] D. Georgakopoulos, P. P. Jayaraman, M. Fazia, M. Villari, and R. Ranjan, "Internet of things and edge cloud computing roadmap for manufacturing," *IEEE Cloud Computing*, vol. 3, no. 4, pp. 66–73, 2016.

[4] T. Subramanya, L. Goratti, S. N. Khan, E. Kafetzakis, I. Giannoulakis, and R. Riggio, "A practical architecture for mobile edge computing," in *Proceedings of the IEEE Conference Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 174–177, Berlin, Germany, November 2017.

[5] R. Yuan and H. Li, "A multidisciplinary coupling relationship coordination algorithm using the hierarchical control methods of complex systems and its application in multidisciplinary design optimization," *Advances in Mechanical Engineering*, vol. 9, no. 1, pp. 1–11, 2017.

[6] R. Yuan, H. Li, Z. Gong, M. Tang, and W. Li, "An enhanced Monte Carlo simulation–based design and optimization method and its application in the speed reducer design," *Advances in Mechanical Engineering*, vol. 9, no. 9, pp. 1–7, 2017.

[7] R. Yuan, H. Li, and Q. Wang, "An enhanced genetic algorithm-based multi-objective design optimization strategy,"

Advances in Mechanical Engineering, vol. 10, no. 7, pp. 1–6, 2018.

[8] H. Lu, C. Gu, F. Luo, W. Ding, and X. Liu, "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] F. Yang, P. Cai, H. Qian, and X. Luo, "Task offloading strategy and pricing scheme in fog-enabled networks," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Waikoloa, HI, USA, December 2019.

[10] T.-P. Pham, J. J. Durillo, and T. Fahringer, "Predicting workflow task execution time in the cloud using A two-stage machine learning approach," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 256–268, 2017.

[11] B. P. Rimal and M. Maier, "Workflow scheduling in multitenant cloud computing environments," *IEEE Transactions on Parallel & Distributed Systems*, vol. 28, no. 1, pp. 290–304, 2016.

[12] L. Tang and S. He, "Multi-user computation offloading in mobile edge computing: a behavioral perspective," *IEEE Network*, vol. 32, no. 1, pp. 48–53, 2018.

[13] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.

[14] X. Meng, W. Wang, Y. Wang, V. K. N. Lau, and Z. Zhang, "Closed-form delay-optimal computation offloading in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4653–4667, 2019.

[15] S.-P. Zhu, B. Keshtegar, S. Chakraborty, and N.-T. Trung, "Novel probabilistic model for searching most probable point in structural reliability analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 366, Article ID 113027, 2020.

[16] R. Liu, P. Chen, X. Zhang, and S. Zhu, "Non-shock ignition probability of octahydro-1,3,5,7-tetranitro-tetrazocine-based polymer bonded explosives based on microcrack stochastic distribution," *Propellants, Explosives, Pyrotechnics*, vol. 45, no. 4, pp. 568–580, 2020.

[17] D. Meng, Y. Li, S.-P. Zhu, Z. Hu, T. Xie, and Z. Fan, "Collaborative maritime design using sequential optimisation and reliability assessment," *Proceedings of the Institution of Civil Engineers—Maritime Engineering*, vol. 173, no. 1, pp. 3–12, 2020.

[18] D. Meng, Z. Hu, P. Wu, S.-P. Zhu, J. A. F. O. Correia, and A. M. P. De Jesus, "Reliability-based optimisation for offshore structures using saddlepoint approximation," *Proceedings of the Institution of Civil Engineers—Maritime Engineering*, vol. 173, no. 2, pp. 33–42, 2020.

[19] D. Meng, T. Xie, P. Wu, S.-P. Zhu, Z. Hu, and Y. Li, "Uncertainty-based design and optimization using first order saddlepoint approximation method for multidisciplinary engineering systems," *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, vol. 6, no. 3, Article ID 04020028, 2020.

[20] H. Li, R. Yuan, and J. Fu, "A reliability modeling for multicomponent systems considering random shocks and multistate degradation," *IEEE Access*, vol. 7, pp. 168805–168814, 2019.

[21] R. Yuan, M. Tang, H. Wang, and H. Li, "A reliability analysis method of accelerated performance degradation based on bayesian strategy," *IEEE Access*, vol. 7, pp. 169047–169054, 2019.

[22] R. Yuan, H. Li, and Q. Wang, "Simulation-based design and optimization and fatigue characteristics for high-speed backplane connector," *Advances in Mechanical Engineering*, vol. 11, no. 6, pp. 1–10, 2019.

[23] J. Xu, S. Wang, B. K. Bhargava, and F. Yang, "A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3538–3547, 2019.

[24] V. Vassilakis, I. P. Chochliouros, A. S. Spiliopoulou et al., "Security analysis of mobile edge computing in virtualized small cell networks," in *Proceedings of the 12th IFIP WG 12.5 International Conference on Artificial Intelligence Applications and Innovations (AIAI)*, pp. 653–665, Thessaloniki, Greece, September 2016.

[25] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: a survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1508–1532, 2019.

[26] P. P. Ray, D. Dash, and D. De, "Edge computing for Internet of Things: a survey, e-healthcare case study and future direction," *Journal of Network and Computer Applications*, vol. 140, pp. 1–22, 2019.

[27] S. Rinaldi, F. Bonafini, P. Ferrari, A. Flammini, E. Sisinni, and D. Bianchini, "Impact of data model on performance of time series database for internet of things applications," in *Proceedings of the 2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pp. 1–6, Auckland, New Zealand, May 2019.

[28] J. Xu, X. Li, R. Ding, and X. Li, "Energy efficient multi-resource computation offloading strategy in mobile edge computing," *Computer Integrated Manufacturing Systems*, vol. 25, no. 4, pp. 954–961, 2019.