AI-enabled IoT-Edge Data Analytics for Connected Living

ZHIHAN LV and LIANG QIAO, School of Data Science and Software Engineering, Qingdao University, China
SAHIL VERMA and KAVITA, School of Computer Science and Engineering, Lovely Professional University, India

As deep learning, virtual reality, and other technologies become mature, real-time data processing applications running on intelligent terminals are emerging endlessly; meanwhile, edge computing has developed rapidly and has become a popular research direction in the field of distributed computing. Edge computing network is a network computing environment composed of multi-edge computing nodes and data centers. First, the edge computing framework and key technologies are analyzed to improve the performance of real-time data processing applications. In the system scenario where the collaborative deployment tasks of multi-edge nodes and data centers are considered, the stream processing task deployment process is formally described, and an efficient multi-edge node-computing center collaborative task deployment algorithm is proposed, which solves the problem of copy-free task deployment in the task deployment problem. Furthermore, a heterogeneous edge collaborative storage mechanism with tight coupling of computing and data is proposed, which solves the contradiction between the limited computing and storage capabilities of data and intelligent terminals, thereby improving the performance of data processing applications. Here, a Feasible Solution (FS) algorithm is designed to solve the problem of placing copy-free data processing tasks in the system. The FS algorithm has excellent results once considering the overall coordination. Under light load, the V value is reduced by 73% compared to the Only Data Center-available (ODC) algorithm and 41% compared to the Hash algorithm. Under heavy load, the V value is reduced by 66% compared to the ODC algorithm and 35% compared to the Hash algorithm. The algorithm has achieved good results after considering the overall coordination and cooperation and can more effectively use the bandwidth of edge nodes to transmit and process data stream, so that more tasks can be deployed in edge computing nodes, thereby saving time for data transmission to the data centers. The end-to-end collaborative real-time data processing task scheduling mechanism proposed here can effectively avoid the disadvantages of long waiting times and unable to obtain the required data, which significantly improves the success rate of the task and thus ensures the performance of real-time data processing.

CCS Concepts: • **Computer systems organization → Embedded systems**; **Redundancy**; **Robotics**; • **Networks → Network reliability**;

Additional Key Words and Phrases: Internet of Things, smart device, edge data, end-to-end collaboration

This work was supported by National Natural Science Foundation of China (Grant No. 61902203) and Key Research and Development Plan—Major Scientific and Technological Innovation Projects of ShanDong Province (Grant No. 2019JZZY020101).
Authors' addresses: Z. Lv (corresponding author) and L. Qiao, College of Computer Science & Technology, Qingdao University, Qingdao 266071, China; emails: lvzhihan@gmail.com, leonqiaoove@gmail.com; S. Verma and Kavita, School of Computer Science and Engineering, Lovely Professional University, India; email: sahilverma@ieee.org, kavita.21914@lpu.co.in.

1533-5399/2021/06-ART104 $15.00
https://doi.org/10.1145/3421510

ACM Reference format:
Zhihan Lv, Liang Qiao, Sahil Verma, and Kavita. 2021. AI-enabled IoT-Edge Data Analytics for Connected Living. *ACM Trans. Internet Technol.* 21, 4, Article 104 (June 2021), 20 pages.
https://doi.org/10.1145/3421510

1 INTRODUCTION

With the rapid development of communication networks and information technology, the **Internet of Things (IoT)** applications have increasingly penetrated people's daily lives [1]. Different types of IoT devices use multiple communication networks to build a complex network system of interconnection in the world, serving scenarios such as smart cities, health care, and smart homes. It is expected that by 2025, as big data reaches an unprecedented level, and IoT will become the basis of this reality, companies that can effectively use IoT technology will not only survive but also flourish [2, 3]. A wide range of smart devices will continuously generate a large amount of event data in real-time. IoT applications need to process continuous data streams generated by batch devices, filter and pre-process these data, perform logical analysis to obtain information from the data, and react according to the obtained information. From home or business applications to connected vehicles, edge computing runs through the entire IoT. As the amount of data increases, this type of computing requires a powerful interconnected edge computing platform with optimal network security functions and the highest functional security level [4].

Matters data acquisition in edge networks

IoT-**Edge Intelligent Data Process (EIDP)** is for IoT operating companies, providing edge **artificial intelligence (AI)** computing platforms including edge AI boxes or gateways, and cloud management platform services, allowing customers to achieve edge AI data processing capabilities simply, quickly, and at low cost [5]. IoT-EIDP allows customers to easily create edge nodes, purchase various AI algorithm models, publish their AI models or secondary development applications, and deploy these products to the edge side through the cloud console with one click, making the IoT data processing response faster, lowering the bandwidth requirements, and further saving customer operation and maintenance costs [6]. The heterogeneous nature of edge devices in the IoT world faces a series of challenges. The remote deployment of models and the edge of performance monitoring is another huge potential area. There must be a powerful mechanism to remotely deploy and fine-tune the AI model. It is also important to pay close attention to the operating condition of the hardware. Academia and industry have proposed a new computing model: edge computing. In the edge computing model, computing and storage resources are scattered from the centralized cloud data center to the edge of the network, such as base stations, routers, and streetlamps. These devices are called edge servers. Edge servers are close to the users, with specific computing, storage, and communication capabilities. Users interact with the edge servers to obtain faster responses and higher service quality.

While rich IoT smart devices provide humans with smart technology experience, they also require huge computing power to support the normal operation of various devices [7]. Currently, many edge devices have built-in computing power. Many IoT-edge devices have a **Graphic Processing Unit (GPU)**, a **Vector Processing Unit (VPU)**, or a **Tensor Processing Unit (TPU)** [8]. For example, some high-end security cameras now have GPU cards, which enables them to run AI-based image recognition models on the edge itself without having to send all high-definition video back to the cloud for processing. Moving processing to the edge ensures better response time and reduces bandwidth usage. As one of the emerging technologies to solve the data processing problem in the IoT, edge computing has attracted more attention [9, 10]. AI edge processing in the IoT world will help provide intelligent real-time decision-making for businesses in a cost-effective and low-latency manner.

Table 1. Data Classification Processing of IoT

Layer	Content	Function
1	Cooperative sensing of local regions	Multiple sensors of the same or different kind perceive the target to be measured
2	Data processing during transmission	Perceptual information fusion processing, application layer coding, and data security transmission
3	Common support, service decision, and coordination control	Data filtering, data aggregation, and data processing

Edge computing network is a network computing environment composed of multi-edge computing nodes and data centers. First, the edge computing framework and key technologies are analyzed. In the system scenario where the collaborative deployment tasks of multi-edge nodes and data centers are considered, the stream processing task deployment process is formally described, and an efficient multi-edge node-computing center collaborative task deployment algorithm is proposed, which solves the problem of copy-free task deployment in the task deployment problem. Furthermore, a heterogeneous edge collaborative storage mechanism with tight coupling of computing and data is proposed, which solves the contradiction between the limited computing and storage capabilities of data and intelligent terminals, thereby improving the performance of data processing applications. The innovation lies in designing an algorithm to solve the problem of copy-less data processing task placement in the system, thereby accurately schedule real-time data processing tasks between the terminal and the edge, as well as balancing the collaborative storage mechanism between edge server load and edge layer model.

2 METHOD

2.1 Real-time Processing of IoT Data Based on Artificial Intelligence

The advantages provided by artificial intelligence are quite attractive, which can help promote the digital transformation of enterprises. As the IoT devices deployed increases, the demand for solutions with edge computing capabilities and artificial intelligence features has grown exponentially. The massive data search technology based on artificial intelligence fuses and matches feature information to obtain data features based on artificial intelligence. According to these data features, the massive data are accurately processed, which reduces the query time and improves the search efficiency, thereby optimizing the processing.

The IoT represents a disruptive creation that has begun to overturn existing processes and technologies, which has brought a brand-new way of working. If the IoT is properly utilized, then it can bring better products and services, customer experience, security, and health care. One of the best ways to exert all the functions of the IoT is real-time analysis. IoT and real-time analysis constitute an optimal combination [11]. Without real-time analysis, it is impossible to utilize all the advantages offered by the IoT. The IoT is also a supplement to real-time analysis and vice versa. From the perspective of information processing, IoT can be divided into three levels. The main contents of each level are shown in Table 1. The bottom layer is the collaborative perception of local areas; the second layer is the data processing during transmission; the third layer is the common support, service decision-making, and coordinated control based on various types of IoT applications on the application support layer [12].

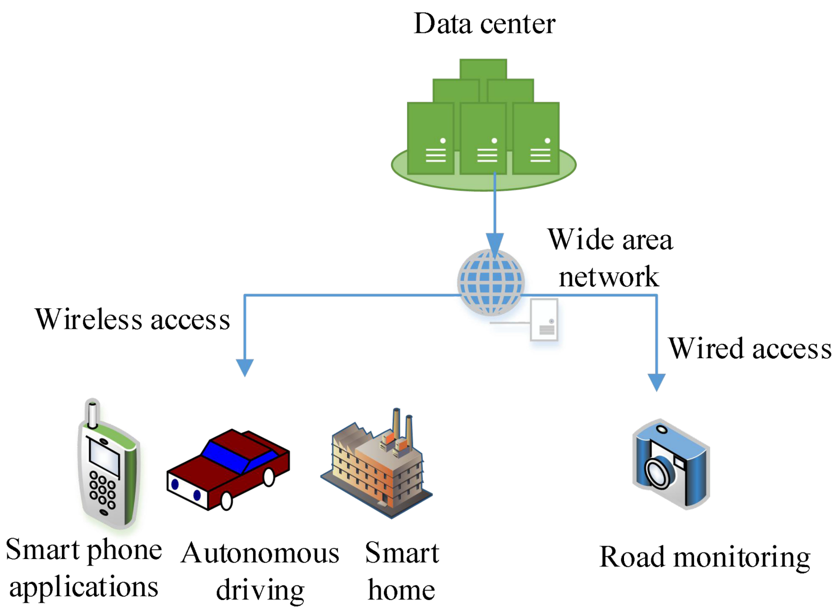

Fig. 1. Schematic diagram of a traditional data center and users.

A wide range of smart devices will continuously generate a large amount of event data in real-time. IoT applications need to process continuous data streams generated by batch devices, filter and pre-process these data, perform logical analysis to obtain information from the data, and react according to the obtained information. These applications usually require low latency to ensure timely statistics and analysis of real-time data, thereby quickly responding to real-time actions of the device. Since IoT applications serve distributed devices in multiple geographic areas, these applications will continue to generate data stream processing requests to the background. IoT data processing applications need to respond to service requests from multiple data sources simultaneously; due to the diversification of background service program programming modes, each application task may have complex and difficult-to-decompose computing connotations and processes [13]. To make it easier to deploy smart devices that are closely related to people's lives, and to allow users to have a good experience, the relevant applications need to perform a lot of data calculation and access operations. The current data center-based task offloading solution can minimize the amount of computing and reduce the computing pressure on the device by transferring computation-intensive tasks from the device to the data center. The schematic diagram of the traditional data center and users is shown in Figure 1.

A data real-time processing method based on the IoT is shown in Figure 2. The method mainly includes: connecting the line between the IoT sensor and the IoT data processing system; collecting sensor-side data in real-time at the sensing layer and the application layer, and transmitting it through the above-mentioned lines; configuring the channel during the transmission process to improve the usage rate of the channel; processing the transmitted data by the data processing system of the IoT; after processing the data, sending the response back to the sensor by the control component through the line; storing the data processed by the data processing system [14–16]. This method can ensure real-time data processing, increase the resource utilization rate of the data processing system, obtain the best anti-interference ability, and maintain the reliability of the

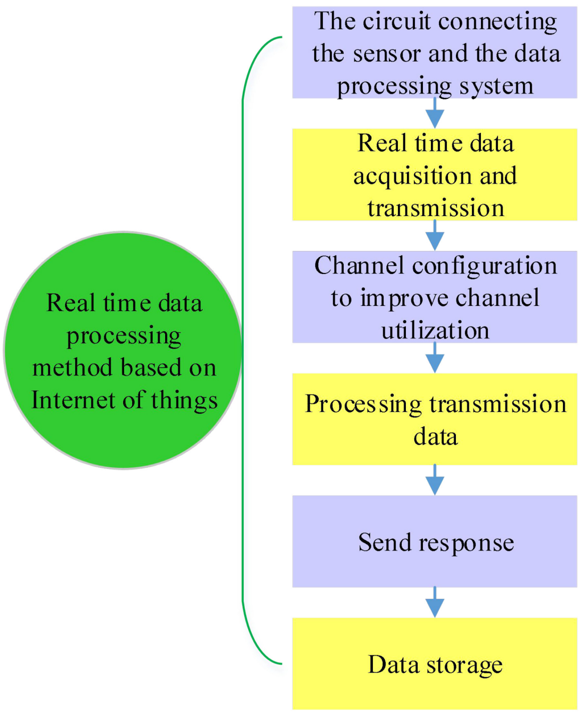
Real time data processing method based on Internet of things
The circuit connecting the sensor and the data processing system
Real time data acquisition and transmission
Channel configuration to improve channel utilization
Processing transmission data
Send response
Data storage

Fig. 2. Real-time data processing method based on IoT.

data, thereby ensuring that the IoT is not affected by changes in external physical factors, which improves the data file access speed and ensures the disaster tolerance of the IoT.

2.2 Edge Computing Framework and key Technologies

2.2.1 Edge Computing. The IoT is an important means to realize the digital transformation of the industry, which will spawn a new industrial ecology and business model. With the help of edge computing, the intelligence of the IoT can be improved, and the IoT can take root in various vertical industries [17]. Edge computing is widely used in the IoT and is particularly suitable for application scenarios with special business requirements such as low latency, high bandwidth, high reliability, massive connections, heterogeneous aggregation, local security, and privacy protection. When deploying edge services, there are three types of edge scenarios that need to be considered, i.e., personal edge, business edge, and cloud edge, which are shown in Figure 3. Personal edge computing revolves around people, such as smart phones, smart robots, medical sensors, wearable watches, and other home automation systems. The service edge is used to aggregate information from personal edge devices. Such devices can be deployed in office areas or home areas to support information interaction and processing within the area. A complex IoT application will involve the coordination of multiple cloud platforms. The cloud edge is equivalent to providing data analysis, data interaction, and data collaboration functions on different cloud platforms [18].

Edge computing is a service network that can provide resource-intensive network applications with powerful computing capabilities and low network service delays. The basic idea is to deploy edge service equipment close to user requests. The edge service equipment and the data center together constitute a two-tier task resource allocation structure, which, in turn, supports application service requests [19, 20]. Edge services will be distributed and deployed in different geographically

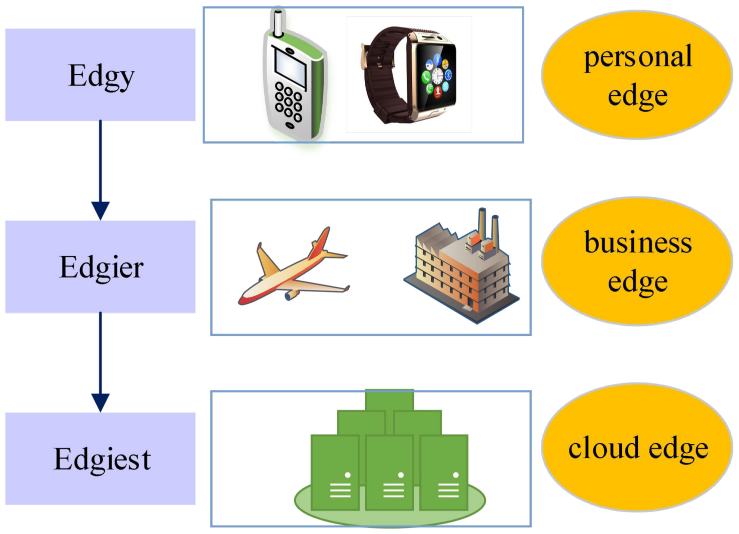

Fig. 3. Three types of edge scenarios.

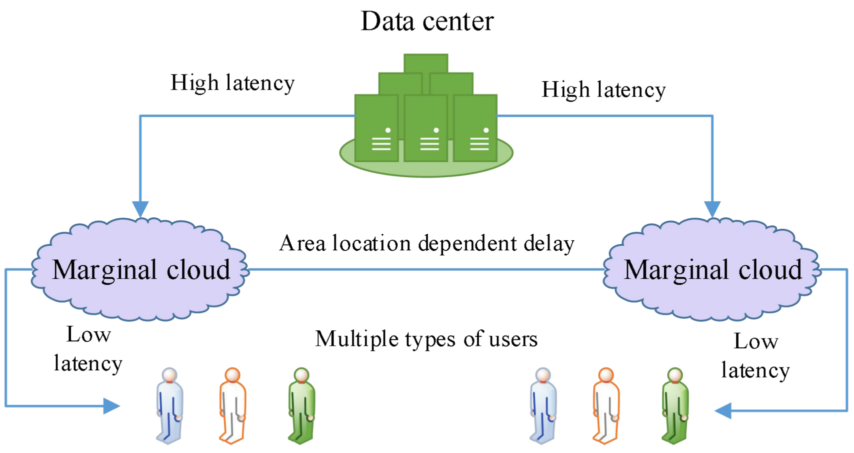

Fig. 4. Delay relation among users, edge cloud, and data center cloud.

distributed edge network areas, with lower latency than data centers; therefore, it can be seen as a cloudlet closer to the user's area. The intelligent monitoring equipment is only responsible for completing the task of data transmission, and the edge computing network mainly operates relevant calculations and data processing. If there is no edge service equipment nearby, then user road data can be transmitted to a remote edge service equipment or data center for calculation, to obtain a better service experience. The delay relation among the user, the edge cloud, and the data center cloud is shown in Figure 4.

2.2.2 Copy-free Streaming data Processing. In a system of data processing tasks, there are multiple application data processing tasks that need to be deployed. Different application data processing tasks are independent individuals, and each task has only one copy in the system [21]. Application data processing tasks accept data inflows from different computing nodes, and the system allocates network bandwidth for communication between computing nodes for

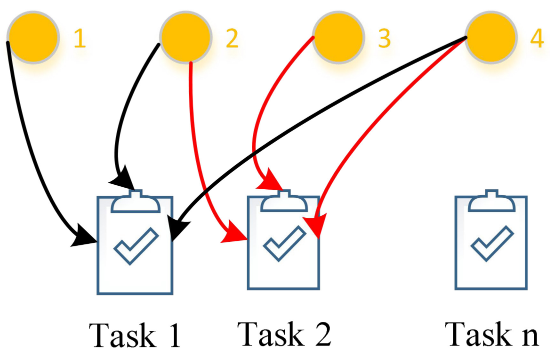

Fig. 5. Schematic diagram of a copy-free deployment decision.

application data processing tasks. There are delays in the bandwidth of different computing nodes, and the bandwidth capability of the data center is much stronger than that of edge computing nodes [22–24]. The data flow that needs to be processed by the same application in the system may need to be analyzed by the processing logic combined by multiple data sources. An application data stream starts from a computing node and is finally sent to the corresponding data processing task through the edge computing network. The schematic diagram of decision-making for the deployment of copy-free tasks is shown in Figure 5.

x_{td} is used to indicate whether data processing task t is deployed on computing node d. If task t is on node d, then the value is 1; otherwise, it is 0. The positional relationship function between the data processing task and the computing node of the system can be expressed as

$$\forall t \in T, x_{td} = \{0, 1\}, \tag{1}$$

$$\forall t \in T, \sum_{d \in D} x_{td} = 1. \tag{2}$$

When t and the data stream are in the same computing node, the bandwidth resources between the computing nodes will not be consumed, and the data stream can be processed; when t and the data stream are not in the same computing node, both computing nodes need to allocate some bandwidth resources for data streaming. After t is deployed in d, the bandwidth resources $P_{td}^{d'}$ consumed by other computing nodes d' in the system can be expressed as

$$\forall t \in T, \forall d' \in D, d' \neq d\, P_{td}^{d'} = \sum_{q \in \{Q_t \cap Q_{d'}\}} r_{qt}, \tag{3}$$

where r_{qt} means that one data stream consumes the bandwidth resources of two nodes, Q_t means the data stream set to be processed by data processing t, $Q_{d'}$ means the data stream set at the data source d', and $\{Q_t \cap Q_{d'}\}$ represents a collection of query requests on the same computing node in the data stream to be processed.

The bandwidth resource consumed by task t after being deployed on node d is represented as L_{td}, and the bandwidth resource consumed by node d is

$$\forall t \in T, \forall d \in D\, L_{td} = \sum_{q \in \{Q_t - Q \cap Q_{d'}\}} r_{qt}, \tag{4}$$

where $\{Q_t - Q \cap Q_{d'}\}$ represents the query request set on different computing nodes in the data stream to be processed.

When a data processing task is deployed in different computing nodes, it will cause different delay effects on different data streams in the dataset to be processed. The delay between the computing node where the data processing task is located and the source node of the data flow is determined by the location of the computing node where the data processing task is located and the computing node where the streaming data source is located. M_{qd} denotes the delay of a data stream from its source node to a computing node d. The total delay cost of the final data stream to t H_{td} in the system is expressed as

$$\forall t \in T, \forall d \in D\, H_{td} = \sum_{q \in \{Q - Q_d\}} M_{qd}. \tag{5}$$

In the system, the deployment problem of copy-free streaming data processing tasks belongs to the non-deterministic polynomial (NP-hard) problem. A computing node is selected for the task entity as the current deployment location. The goal is to select $\{d-1\}$ edge from each of the r edge sets to minimize the sum of the weights of the currently selected edges.

In practical problems, assuming the bandwidth resources of a computing node can transmit all data streams of the node to the wide area network, the following equation can be obtained:

$$\forall d \in D \sum_{t \in T_d} r_{td} \leq U_d, \tag{6}$$

where U_d represents the bandwidth resource limit size of each node d, and r_{td} indicates that a data stream consumes the bandwidth resources of two nodes t and d.

When data processing tasks are deployed in a data center, the system delay cost is higher than the system cost deployed in the edge computing service area. Here, the **Feasible Solution (FS)** algorithm is designed to solve the problem of placing copy-free data processing tasks in the system. The first is to deploy all data processing tasks in the data center; then, the worst possible solution will be obtained. Second, after the task is deployed on any computing node in the system, the bandwidth resources consumed on each computing node are uniquely determined. In the last step, it needs to find another location to replace the original task. If the new solution meets the resource constraints after replacing the deployment location, then the replacement solution is feasible; otherwise, it is not feasible. After t is deployed to d, the increase in total resource consumption compared with the original deployment position d' can be expressed as

$$\Delta a_{td} = \frac{\sum_{j \in J} (w_{td'j} - w_{tdj}) \times Z_j}{|\overrightarrow{Z}|}, \tag{7}$$

where w represents the weight value between the edges, Z_j represents the sum of the bandwidth resources used by the jth area that has been consumed, and $\overrightarrow{Z}$ represents the vector of resources used on each node.

The key to the FS algorithm is to find a feasible task to deploy a replacement location. If a feasible replacement location is found, then the current solution is updated, and the search for another new alternative location is started. Finding a new task deployment location involves three steps: (1) calculating for all candidates; (2) if there is at least one feasible replacement location, then FS will maximize the overall bandwidth resource consumption and perform replacement upgrade; (3) if there is no feasible replacement position, then the new position of t is $\Delta H_{td}/\Delta a_{td}$. If no new feasible alternative deployment location is found during the above process, then the program stops and the current task deployment plan is output.

2.3 End-to-end Collaborative Real-time Data Processing Tasks

2.3.1 Processing Cost Modeling. The real-time data processing tasks generated by intelligent terminals include the following scheduling strategies: terminal processing, edge server processing, and neighboring server processing [25, 26]. Because it is necessary to complete a scheduling decision at a certain interval, the final task completion time will include the arrival time, waiting time, and service time.

If the terminal can meet the deadline requirement of the task, then the task can be processed by the terminal without offloading to the edge layer processing. At this time, the task arrival time is the start time t_{design} of this decision; the waiting time depends on the execution order of the task on the terminal q_i, the service time depends on terminal processing capacity f_i, and the completion time t_{ij}^{finish} of task t can be expressed as

$$t_{ij}^{finish} = wait_{ij} + \sum_{q_{ik}^{i} < q_{ij}^{i}} \frac{c_{ik}}{f_i} + \frac{c_{ij}}{f_i}, \tag{8}$$

where c_{ij} and c_{ik} represent the calculation amount of tasks t_{ij} and t_{ik}, respectively.

If the terminal cannot meet the task requirements, and the edge server connected to the terminal can provide the required resources and model data, then the edge server can process the task. The task start time is determined by the decision time t_{design} and the time for the task to be transmitted to the edge server $trans_i^{edge_i}$. The waiting time depends on the order $q_{ij}^{edge_i}$ in which the edge server processes the task. The service time depends on the processing capacity of the edge server f_{edge_i}. The completion time of task t t_{ij}^{finish} can be expressed for

$$t_{ij}^{finish} = wait_{ij} + \frac{input_{ij}}{B_i^{edge_i}} + \sum_{q_{ik}^{edge_i} < q_{ij}^{edge_i}} \frac{c_{tk}}{f_{edge_i}} + \frac{c_{ij}}{f_{edge_i}}, \tag{9}$$

where $input_{ij}$ represents the input size of t_{ij}, and $B_i^{edge_i}$ represents the bandwidth between the edge server z and the adjacent z'.

If the edge server connected to the terminal still cannot process the task effectively, then the task is allowed to migrate from the edge server to the neighboring server [27]. The arrival time of a task is determined by the time $trans_i^z$ for the task to be transmitted from the terminal to the adjacent server. The waiting time depends on the order q_{ij}^z in which the adjacent server processes the task. The service time depends on the processing capacity f_z of the adjacent server. The completion time of the task t t_{ij}^{finish} can be expressed as

$$t_{ij}^{finish} = wait_{ij} + \frac{input_{ij}}{B_i^{edge_i}} + \frac{input_{ij}}{B_{edge_i}^{z}} + \sum_{q_{tk}^{z} < q_{ij}^{z}} \frac{c_{tk}}{f_z} + \frac{c_{ij}}{f_z}, \tag{10}$$

where $B_{edge_i}^z$ represents the bandwidth between the adjacent server and another server adjacent to it.

2.3.2 Real-time Data Processing Task Scheduling Mechanism. To reduce the computational complexity, an end-to-end collaborative real-time data processing task scheduling mechanism is proposed. The mechanism includes two stages: the terminal task scheduling stage and the edge layer task scheduling stage. In the previous stage, the terminal only makes scheduling decisions based on its processing capacity and data storage conditions, which will determine whether to migrate the task to the edge server; in the second stage, the edge controller is responsible for selecting the appropriate edge server for each task of the terminal migration [28–30].

The terminal task scheduling decision only completes as many tasks as possible on the terminal according to the terminal's task load and data storage and moves the remaining tasks to the edge layer. The implementation is simple and the complexity is low. The mathematical modeling of task scheduling between the terminal and the edge layer can be expressed as

$$\max \sum_{i \in I} \sum_{j \in J_i} \alpha_{ij}\, success_{ij}, \tag{11}$$

$$t_{ij}^{finish} = t_{design} + wait_i + \sum_{q_{ik}^i < q_{ij}^i} \frac{c_{ik}}{f_i} + \frac{c_{ij}}{f_i}, \alpha_{ij} = 1, \tag{12}$$

where I represents the terminal set, i is the terminal number, J is the task set of terminal i; $success_{ij}$ indicates whether the task is successful; the completion before the deadline is 1; otherwise, it is 0.

In the task scheduling problem between the terminal and the edge, the task scheduling on each terminal is independent of each other. Since the start time of the task is not fixed, this strategy should consider not only the compatibility between tasks but also the influence of the terminal processing order on the task start time.

After the terminal decision stage is completed, the task set on the terminal is divided into the task set that has been decided to be processed by the terminal, and the task set to be uninstalled. The latter cannot be completed by the terminal before the deadline and needs to be migrated to the edge layer for processing [31]. For a task to be uninstalled on a terminal, the target migration location O_{ij} of task $task_{ij}$ includes only the edge server $edge_i$ connected to the terminal and the adjacent server E_{edge_i} of the server. The modeling structure of the task scheduling problem at the edge layer can be expressed as

$$\max \sum_{i \in I} \sum_{j \in J_i^{off}} success_{ij}, \tag{13}$$

$$t_{ij}^{finish} = \begin{cases} t_{design} + trans_i^{O_{ij}} + wait_{O_{ij}} + \sum_{q_{ik}^{O_{ij}} < q_{ij}^{O_{ij}}} \frac{c_{tk}}{f_{O_{ij}}} + \frac{c_{ij}}{f_{O_{ij}}} \\ trans_i^{O_{ij}}, otherwise \end{cases}, \tag{14}$$

where J_i^{off} represents the remaining set of tasks to be uninstalled in terminal i, and $wait_{O_{ij}}$ represents the remaining task processing time.

Only when the completion time is within the deadline, it is considered successful, and the transmission time and service time of the existing work of non-edge computing can be considered successful within the deadline. If neither the edge server connected to the terminal nor the adjacent server can meet the deadline of the task, then the task needs to be migrated to the cloud data center [32–34]. Because the computing and storage capabilities of the cloud data center are very powerful, the processing time of the task can be ignored. However, the distance between the cloud data center and the user is too far, the time of the migration task is the main time overhead. Therefore, the terminal migration time to the cloud data center $trans_i^{cloud}$ is used as the migration cost.

The edge layer task scheduling mechanism is mainly divided into three steps: proposal generation, proposal decision, and task migration. First, after receiving the task information sent by the terminal, the edge controller comprehensively considers the task requirements and edge server capabilities and generates a series of task proposals to all edge servers that meet the task requirements. In the proposal decision stage, the edge controller will decide which proposals are effective or invalid, as well as the processing order of the proposed corresponding tasks. After the proposed decision step is completed, the edge controller sends the target migration location and processing order corresponding to each task to the terminal and the edge server according to the decision result.

2.4 Simulation Experiment

2.4.1 Performance Test of the FS Algorithm Copy-free Data Processing. Simulation experiments are used to verify the effectiveness of the proposed FS algorithm. The simulation experiment platform uses MATLAB 2018b software on a server with 16 GB memory. In the edge network environment, the network delay of the general application transmission data to the data center is about 0.75 s. Since the delay between edge computing nodes is related to the regional distribution of edge service nodes in the edge network, the delay between edge computing nodes is generally between 0.1 and 0.4 s. The bandwidth capability of each edge computing node to transmit data to the WAN is set as 50 MB/s. To verify the algorithm performance at different edge computing node sizes, set the number of edge nodes in the system to 50~500, with an increase of every 50 nodes.

When generating its specific information, the data stream to be processed is first generated on a data gateway of the computing node. Based on a particular number of edge nodes, the processing requirements of the data stream under light and heavy loads are set. In the case of light load, the bandwidth occupancy of the data streams generated on all computing nodes is about 60% of the bandwidth of the computing nodes; that is, one node generates 16 data streams. In the case of heavy load, the bandwidth occupancy of the data streams generated on all computing nodes is about 80% of the bandwidth of the computing nodes; that is, one node generates 20 data streams.

Here, the **Only Data Center Available (ODC)** algorithm, the Hash algorithm, and the proposed FS algorithm are compared. ODC is a method of deploying all tasks to the data center; the hash function is used to deploy task t in the edge service node; if the node that meets the bandwidth resource cannot be found, then the computing task is deployed to the data center. If V denotes the sum of delay costs, then V will be expressed as

$$V = \sum_{t \in T, d \in D} x_{td} H_{td}, \tag{15}$$

where x_{td} indicates whether the data processing task t is deployed on the computing node d, and H_{td} indicates the total delay cost of the final data flow to t in the system. The goal here is to find a deployment solution that minimizes the V value in a multi-computing node edge-data center computing network.

2.4.2 End-side Scheduling Mechanism Based on Multiple Features. Here, we design a simulation program for an edge computing system for real-time processing tasks of intelligent terminals to test the effectiveness of the proposed mechanism. Here, the BRITE software developed by the School of Computer Science is used to generate the topological relationship of the edge layer, and the average adjacent number represents different topologies. The total number of edge servers is 20, and the computing power depends on the product of basicCapacity and the adjacent number. The more adjacent numbers are, the stronger the computing power is. The default setting of basicCapacity is 60, the edge storage capacity is 40,000~50,000, the average connected server is 4, and the task arrival rate is 50. The bandwidth between the edge servers is 1 Gbps, and each edge server is connected to 10~15 intelligent terminals.

DCSG (Delay-sensitive, Cooperation with Success Rate, Greedy) is a task scheduling mechanism proposed here. The edge server adopts a greedy strategy to store data. DCSG adopts the task scheduling mechanism of end-side collaboration, as well as a greedy strategy for data storage at the edge layer. In other words, each edge server saves model data from large to small according to its data access preferences. Here, the proposed end-side scheduling mechanism is compared with three other end-side scheduling strategies. The three scheduling strategies are: (1) **ENG (Edge only, No cooperation, Greedy)** strategy, the terminal migrates all tasks to the connected edge server, but each edge server processes tasks independently and stores data in a greedy strategy.

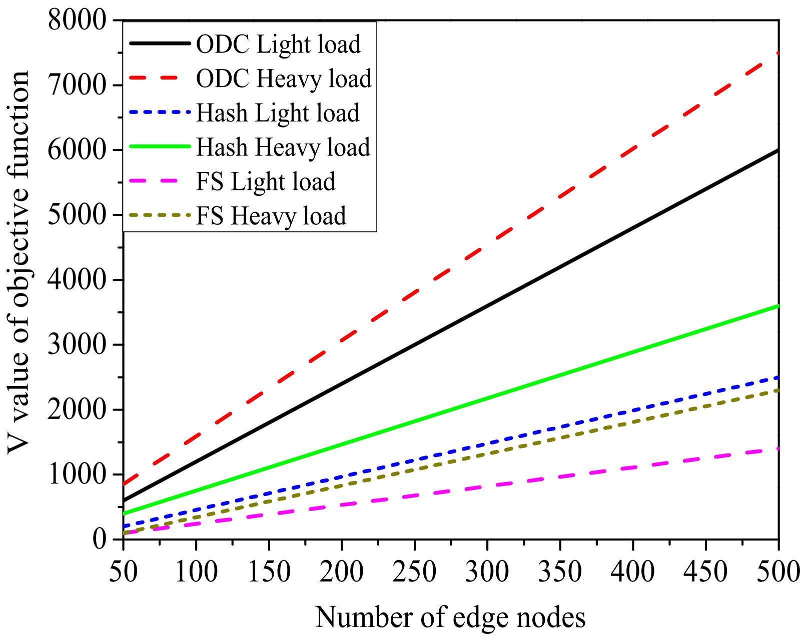

Fig. 6. Variation curve of V values of different task load for each algorithm.

(2) **OCEG (Offload, Cooperation with Execution time, Greedy)** strategy, the task offloading between the terminal and the edge server adopts the idea of terminal and edge processing cost, the edge server cooperates and considers how to minimize the task service time and uses greedy strategy to store data. (3) **OCSG (Offload, Cooperation with Success Rate, Greedy)** strategy, the task offloading between the terminal and the edge server adopts the idea of terminal and edge processing costs, the collaboration between the edge servers considers task queuing and service time and adopts a greedy strategy to store data. Also, through the control variable method, the influence of factors, such as edge computing power, edge storage space, and average arrival interval of tasks on the accuracy of task scheduling decision results and task success rate, is tested.

3 RESULTS AND DISCUSSION

3.1 Performance of Algorithms Under Different Loads

Given the different number of edge nodes, the V values of the objective function obtained by the ODC algorithm, Hash algorithm, and the proposed FS algorithm under light load and heavy load are tested separately. The variation curve of V values of different task load of each algorithm is shown in Figure 6. The ODC method does not consider the role of edge computing, while the Hash method only considers whether the bandwidth resources of edge nodes are satisfied. Figure 7 shows the average request delay of different algorithms under different loads and edge nodes. The average request delay value of other algorithms is used to minus the average request delay value of the FS algorithm, which is then divided by the average request delay value of other methods to obtain the final result.

The FS algorithm proposed here is superior to other algorithms under different loads. Among them, the V value of the ODC algorithm is the highest, because it deploys all tasks to the data center, thereby increasing the total delay cost. The Hash algorithm can partially utilize the resources of edge cloud nodes, which is superior to the ODC algorithm. The proposed FS algorithm has achieved

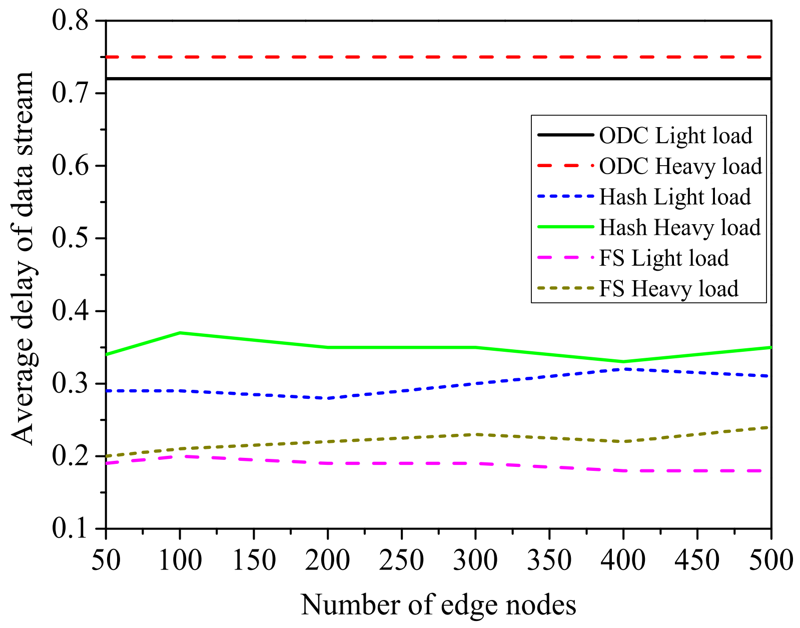

Fig. 7. Average delay variation curves of different load requests of various algorithms.

good results after considering the overall coordination. Under light load, the V value is reduced by 73% compared to the ODC algorithm and 41% compared to the Hash algorithm; under heavy load, the V value is 66% less than the ODC algorithm and 35% less than the Hash algorithm.

Next, the changes in the average utilization rate of bandwidth resources in the edge computing node system with different loads and edge node numbers are analyzed, as shown in Figure 8. Under light load, the proposed FS algorithm has advantages in improving the efficiency of system resource usage compared to the ODC algorithm and the Hash algorithm, which is 35% higher than the ODC algorithm and 20% higher than the Hash algorithm. The proposed FS algorithm can more effectively use the bandwidth of edge nodes to transmit and process data stream, so that more tasks can be deployed in edge computing nodes, thereby saving time for data transmission to the data centers. However, under heavy load, the FS algorithm is superior to the other two algorithms, but its effect in improving the utilization of bandwidth resources is not obvious. The main reason is that the bandwidth resources of each node are naturally occupied and cannot accommodate more computing tasks deployments.

By analyzing the problem of copy-free streaming tasks, the FS algorithm can analyze the global bandwidth resources of the edge computing nodes and the delay between the nodes well, and it can select as many computing tasks as possible to be deployed on the edge nodes, so that the consumption of bandwidth resources of edge computing nodes can obtain greater benefits. Through the above simulation experiments, it is verified that the FS algorithm places tasks on edge computing nodes as much as possible, which improves the efficiency of bandwidth resource usage of edge computing nodes.

On this basis, the task copies are added, and the performance of the next algorithm is compared in two and four task copies, respectively. The changes in V values, the average delay of data flow, and the usage rate of bandwidth resources of edge nodes under different task copies are illustrated in Figures 9~11.

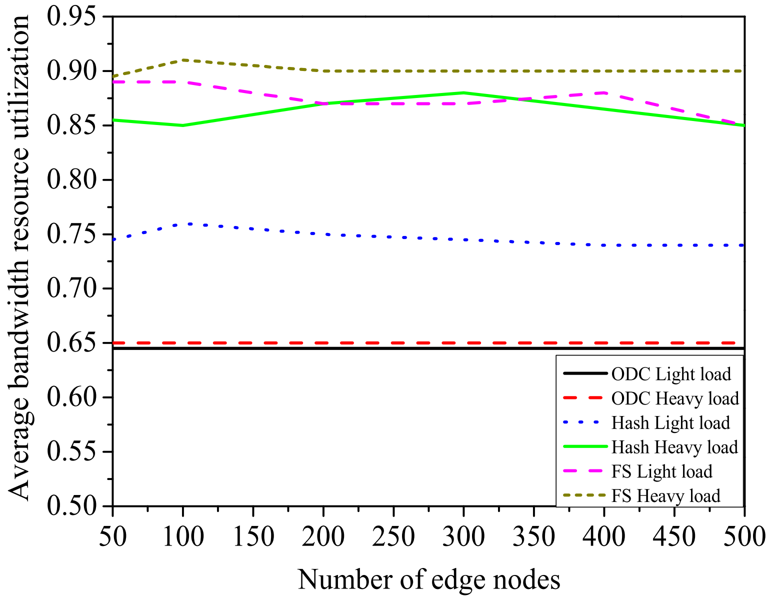

Fig. 8. The average bandwidth resource usage rate of the different load of each algorithm.

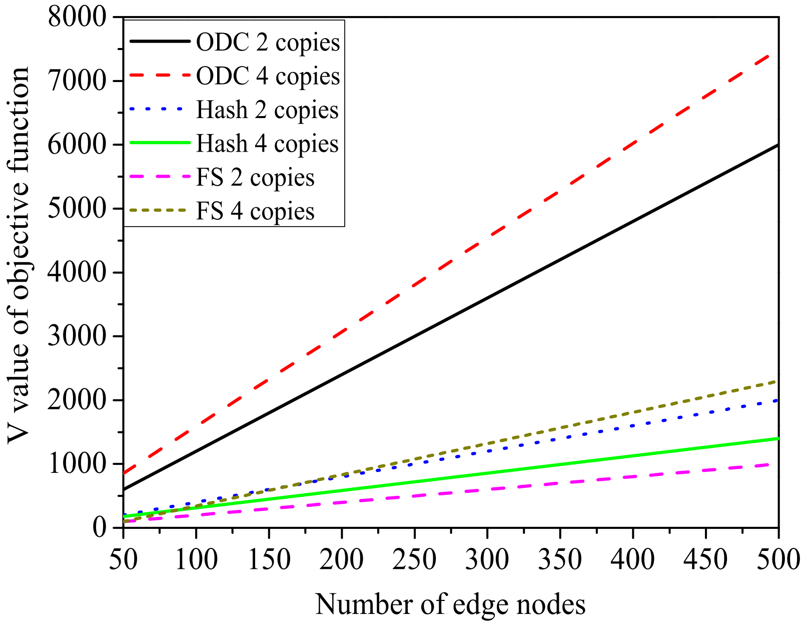

Fig. 9. V value changes under different task copies of each algorithm.

When there are two task copies, the V value of the FS algorithm is reduced by 80% compared to the ODC algorithm and 35% compared to the Hash algorithm; when there are four task copies, the V value of the FS algorithm is reduced by 85% compared to the ODC algorithm and 15% compared to the Hash algorithm. The average delay discovery of requests is analyzed by several algorithms under the different number of edge nodes. Whether it is two copies or four copies, the average

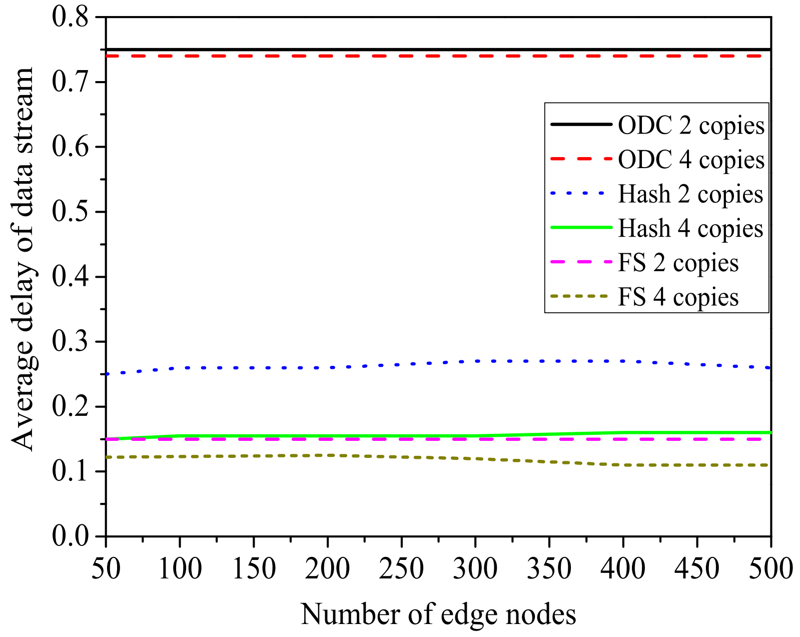

Fig. 10. Average delay of data flow under different task copies of each algorithm.

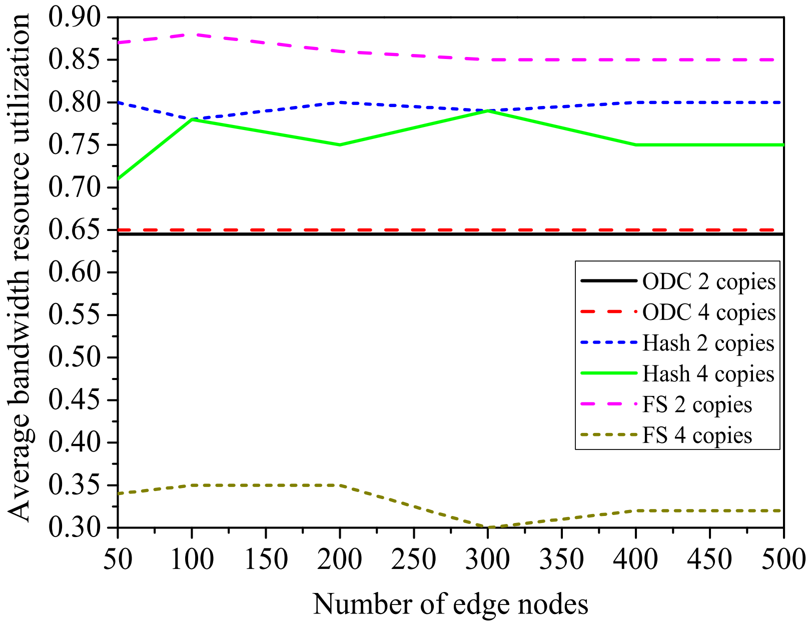

Fig. 11. Bandwidth resource usage rate of edge nodes under different task copies of each algorithm.

delay of the FS algorithm request is significantly lower than the other two algorithms. In the two copy tasks, the FS algorithm performs a second rerouting process, and it can find the computing nodes with free bandwidth resources around it so that the task is deployed. Compared with the ODC algorithm and Hash algorithm, the request delay is improved. However, the resource usage rate of the edge node of the FS algorithm is reduced when there are four copy tasks,

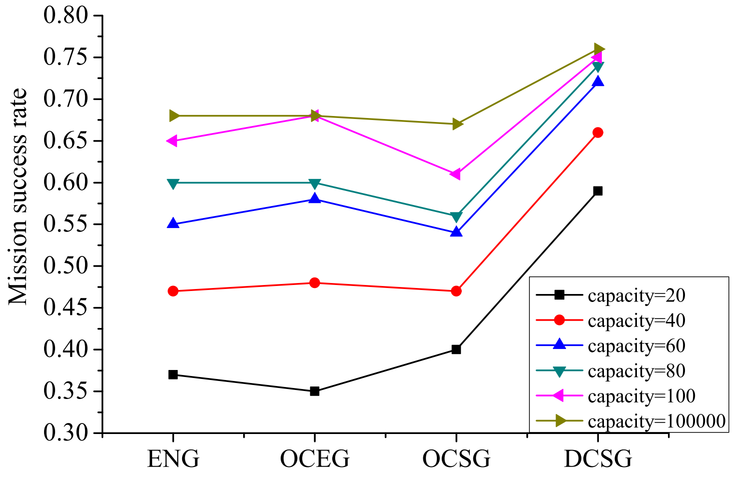

Fig. 12. The task success rate for different edge server processing capabilities.

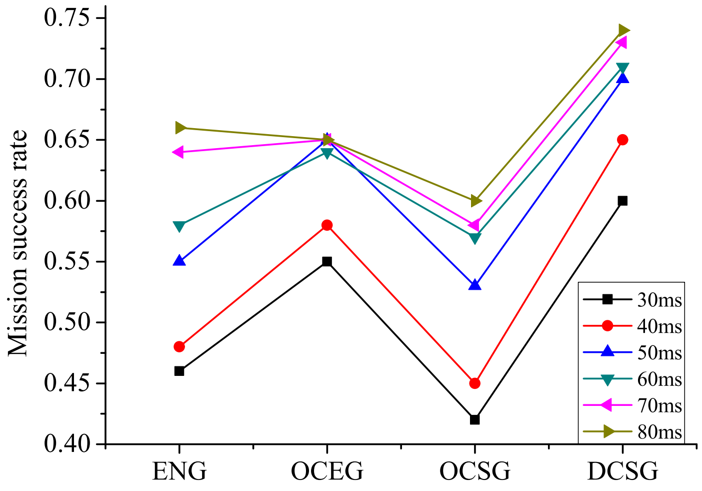

Fig. 13. The mission success rate for different mission arrival intervals.

mainly because the number of task copies increases and the FS has the characteristics of local data processing.

3.2 Task Completion Status of End-side Scheduling Strategy

By testing the task completion of the four scheduling mechanisms, it is found that when the processing capacity of the edge server is weak, the power of the four task scheduling machines is not high, but the proposed DCSG mechanism is significantly better than the other three mechanisms. The effect of edge server processing power on the task success rate is shown in Figure 12. As the edge processing power increases, the task success rate also increases, but even when the edge processing power is infinite, about 25% of the tasks will still fail to obtain the required data. This is mainly because the edge server adopts a greedy strategy to store model data so that some data cannot be stored in the edge server.

The impact of the task arrival rate on the task success rate is tested. The specific results are shown in Figure 13. No matter how the average task arrival interval changes, the DCSG

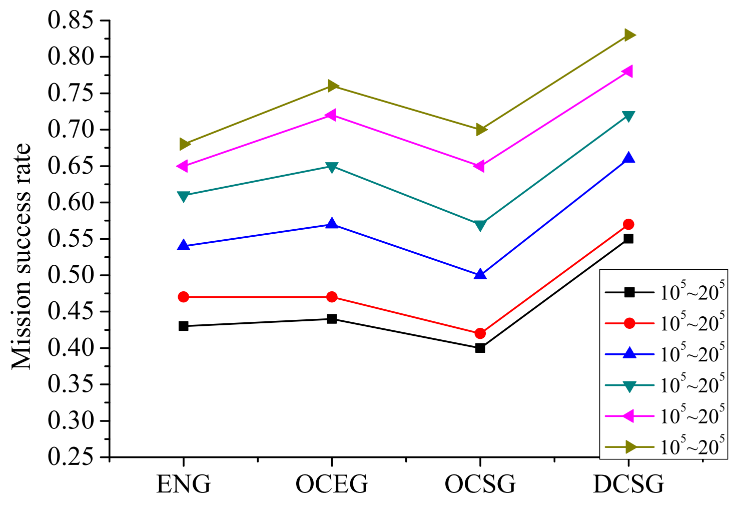

Fig. 14. The task success rate for different edge server storage capabilities.

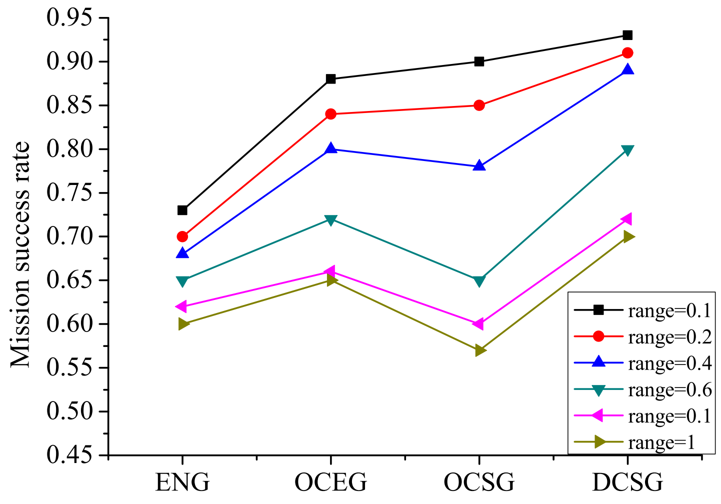

Fig. 15. The mission success rate for random ranges of different required models.

mechanism is obviously superior to other scheduling mechanisms. As the average task arrival interval increases, the task load of the smart terminal and the edge server is lower, so more tasks can be completed before the deadline. As the average task arrival interval increases, too long a waiting time will result in fewer tasks that cannot be completed before the deadline.

The influence of the storage capacity of the edge server and the random range of the model on the task success rate is illustrated in Figures 14 to 15. As the storage space of the edge server is larger, the task success rate is higher, it is more difficult to obtain the required data at this time, and the task failure rate is also lower. As the random range of the model required by the task increases, the failure rate of the task completion also increases. The major cause is the increased storage space of the edge server; even if the edge server adopts a greedy storage strategy, it can store more data, the success rate of the task will also increase. This shows that whether the edge server and its neighboring servers can store as much different data as possible has a great impact on the success rate of the task.

4 CONCLUSIONS

The mechanism of collaborative computing and storage between the terminal and the edge in the edge environment for the characteristics of real-time data processing applications is researched, such as delay sensitivity and a large amount of calculation. When data processing tasks are deployed in a data center, the system delay cost is higher than the system cost deployed in the edge computing service area. Here, the FS algorithm is designed to solve the problem of system copy-free data processing task deployment. The objective function V values obtained under the light load and heavy load of the ODC algorithm, Hash algorithm, and the proposed FS algorithm are tested under different numbers of edge nodes. The FS algorithm proposed here is superior to other algorithms under different loads. The proposed FS algorithm has achieved good results after considering the overall coordination and cooperation and can more effectively use the bandwidth of edge nodes to transmit and process data stream, so that more tasks can be deployed in edge computing nodes, thereby saving time for data transmission to the data centers.

To reduce the computational complexity, an end-to-end collaborative real-time data processing task scheduling mechanism is proposed. The mechanism includes two stages: the terminal task scheduling stage and the edge layer task scheduling stage. Here, the impact of the task arrival rate on the task success rate is tested. As the average task arrival interval increases, the task load of the smart terminal and the edge server is lower, so more tasks can be completed before the deadline. As the random range of the model required by the task increases, the failure rate of the task completion also increases. This shows that whether the edge server and its neighboring servers can store as much different data as possible has a great impact on the success rate of the task. In summary, the proposed algorithm can effectively avoid the disadvantages of long waiting times and unable to obtain the required data, which significantly improves the success rate of the task and thus ensures the performance of real-time data processing. The results have suggested that for smart terminals with weak processing capabilities, the time consumed by various links, such as serialization, deserialization, and network transmission, is considerable, which has a more significant impact on the performance of real-time data processing applications. Therefore, in the future, the overall architecture, code implementation, and technical routes of the prototype system will be optimized, and more real-time data processing applications will be built to enrich the application scenarios of the prototype system. In future the proposed research may improve the previous researches in IoT, 6G, and AI [35–43].

REFERENCES

[1] M. Du, K. Wang, Y. Chen, et al. 2018. Big data privacy preserving in multi-access edge computing for heterogeneous Internet of Things. *IEEE Commun. Mag.* 56, 8 (2018), 62–67.

[2] T. Wang, L. Qiu, A. K. Sangaiah, et al. 2020. Edge-computing-based trustworthy data collection model in the internet of things. *IEEE Internet Things J.* 7, 5 (2020), 4218–4227.

[3] K. Kaur, S. Garg, G. S. Aujla, et al. 2018. Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay. *IEEE Commun. Mag.* 56, 2 (2018), 44–51.

[4] P. Porambage, J. Okwuibe, M. Liyanage, et al. 2018. Survey on multi-access edge computing for internet of things realization. *IEEE Commun. Surveys Tutor.* 20, 4 (2018), 2961–2991.

[5] X. Li, D. Li, J. Wan, et al. 2018. Adaptive transmission optimization in SDN-based industrial internet of things with edge computing. *IEEE Internet Things J.* 5, 3 (2018), 1351–1360.

[6] J. Pan and J. McElhannon. 2017. Future edge cloud and edge computing for internet of things applications. *IEEE Internet Things J.* 5, 1 (2017), 439–449.

[7] D. Puthal, S. Nepal, R. Ranjan, et al. 2016. Threats to networking cloud and edge datacenters in the Internet of Things. *IEEE Cloud Comput.* 3, 3 (2016), 64–71.

[8] M. B. Mollah, M. A. K. Azad, and A. Vasilakos. 2017. Secure data sharing and searching at the edge of cloud-assisted internet of things. *IEEE Cloud Comput.* 4, 1 (2017), 34–42.

[9] G. Mitsis, E. E. Tsiropoulou, and S. Papavassiliou. 2020. Data offloading in UAV-assisted multi-access edge computing systems: A resource-based pricing and user risk-awareness approach. *Sensors* 20, 8 (2020), 2434.

[10] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, et al. 2017. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments. *Softw.: Pract. Exper.* 47, 9 (2017), 1275–1296.

[11] R. Ranjan, O. Rana, S. Nepal, et al. 2018. The next grand challenges: Integrating the internet of things and data science. *IEEE Cloud Comput.* 5, 3 (2018), 12–26.

[12] H. Khelifi, S. Luo, B. Nour, et al. 2018. Bringing deep learning at the edge of information-centric internet of things. *IEEE Communications Letters* 23, 1 (2018), 52–55.

[13] D. Sabella, A. Vaillant, P. Kuure, et al. 2016. Mobile-edge computing architecture: The role of MEC in the Internet of Things. *IEEE Consum. Electron. Mag.* 5, 4 (2016), 84–91.

[14] G. C. Nobre and E. Tavares. 2017. Scientific literature analysis on big data and internet of things applications on circular economy: A bibliometric study. *Scientometrics* 111, 1 (2017), 463–492.

[15] R. Li, T. Song, B. Mei, et al. 2018. Blockchain for large-scale internet of things data storage and protection. *IEEE Trans. Services Comput.* 12, 5 (2018), 762–771.

[16] F. Lin, Y. Zhou, X. An, et al. 2018. Fair resource allocation in an intrusion-detection system for edge computing: Ensuring the security of Internet of Things devices. *IEEE Consum. Electron. Mag.* 7, 6 (2018), 45–50.

[17] C. H. Chen, M. Y. Lin, and C. C. Liu. 2018. Edge computing gateway of the industrial internet of things using multiple collaborative microcontrollers. *IEEE Netw.* 32, 1 (2018), 24–32.

[18] E. Fitzgerald, M. Pióro, and A. Tomaszewski. 2018. Energy-optimal data aggregation and dissemination for the Internet of Things. *IEEE Internet Things J.* 5, 2 (2018), 955–969.

[19] Z. Ning, P. Dong, X. Kong, et al. 2018. A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things. *IEEE Internet Things J.* 6, 3 (2018), 4804–4814.

[20] A. Munir, P. Kansakar, S. U. Khan. 2017. IFCIoT: Integrated fog cloud IoT: A novel architectural paradigm for the future Internet of Things. *IEEE Consum. Electr. Mag.* 6, 3 (2017), 74–82.

[21] T. Adegbija, A. Rogacs, C. Patel, et al. 2017. Microprocessor optimizations for the internet of things: A survey. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 37, 1 (2017), 7–20.

[22] Z. Xiong, Y. Zhang, N. C. Luong, et al. 2020. The best of both worlds: A general architecture for data management in blockchain-enabled Internet-of-Things. *IEEE Netw.* 34, 1 (2020), 166–173.

[23] M. Abbasi, E. M. Pasand, and M. R. Khosravi. 2020. Workload allocation in IoT-fog-cloud architecture using a multi-objective genetic algorithm. *J. Grid Comput.* 18 (2020), 43–56.

[24] J. Lin, W. Yu, N. Zhang, et al. 2017. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet Things J.* 4, 5 (2017), 1125–1142.

[25] Y. Zhang, H. Huang, L. X. Yang, et al. 2019. Serious challenges and potential solutions for the industrial Internet of Things with edge intelligence. *IEEE Netw.* 33, 5 (2019), 41–45.

[26] Q. Huang, Y. Yang, and L. Wang. 2017. Secure data access control with ciphertext update and computation outsourcing in fog computing for Internet of Things. *IEEE Access* 5 (2017), 12941–12950.

[27] Y. Chen, N. Zhang, Y. Zhang, et al. 2018. Dynamic computation offloading in edge computing for Internet of Things. *IEEE Internet Things J.* 6, 3 (2018), 4242–4251.

[28] G. Jia, G. Han, H. Xie, et al. 2018. Hybrid-LRU caching for optimizing data storage and retrieval in edge computing-based wearable sensors. *IEEE Internet Things J.* 6, 2 (2018), 1342–1351.

[29] O. Elijah, T. A. Rahman, I. Orikumhi, et al. 2018. An overview of Internet of Things (IoT) and data analytics in agriculture: Benefits and challenges. *IEEE Internet Things J.* 5, 5 (2018), 3758–3773.

[30] W. Chen, Z. Zhang, Z. Hong, et al. 2019. Cooperative and distributed computation offloading for blockchain-empowered industrial Internet of Things. *IEEE Internet Things J.* 6, 5 (2019), 8433–8446.

[31] W. Wang, Q. Wang, and K. Sohraby. 2016. Multimedia sensing as a service (MSaaS): Exploring resource saving potentials of at cloud-edge IoT and fogs. *IEEE Internet Things J.* 4, 2 (2016), 487–495.

[32] G. Li, J. He, S. Peng, et al. 2018. Energy efficient data collection in large-scale internet of things via computation offloading. *IEEE Internet Things J.* 6, 3 (2018), 4176–4187.

[33] G. Li, J. Wu, J. Li, et al. 2018. Service popularity-based smart resources partitioning for fog computing-enabled industrial Internet of Things. *IEEE Trans. Industr. Inform.* 14, 10 (2018), 4702–4711.

[34] B. Du, R. Huang, Z. Xie, et al. 2018. KID model-driven things-edge-cloud computing paradigm for traffic data as a service. *IEEE Netw.* 32, 1 (2018), 34–41.

[35] Z. Lv and H. Song. 2019. Mobile internet of things under data physical fusion technology. *IEEE Internet Things J.* 7, 5 (Nov. 2019), 4616–4624.

[36] R. Shi, Y. Gan, and Y. Wang. 2018. Evaluating scalability bottlenecks by workload extrapolation. In *Proceedings of the IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'18)*. IEEE, 333–347.

[37] Z. Lv and W. Xiu. 2019. Interaction of edge-cloud computing based on SDN and NFV for next generation IoT. *IEEE Internet Things J.* 7, 7 (Oct. 2019), 5706–5712.

[38] S. Dang, O. Amin, B. Shihada, and M. S. Alouini. 2020. What should 6G be? *Nature Electron.* 3, 1 (2020), 20–29.

[39] Z. Lv, X. Li, H. Lv, and W. Xiu. 2019. BIM big data storage in WebVRGIS. *IEEE Trans. Industr. Inform.* 16, 4 (May 2019), 2566–2573.

[40] A. Zhou, S. Wang, S. Wan, and L. Qi. 2020. LMM: Latency-aware micro-service mashup in mobile edge computing environment. *Neural Comput. Appl.* 1–15, 2020.

[41] S. Wan, Z. Gu, and Q. Ni. 2020. Cognitive computing and wireless communications on the edge for healthcare service robots. *Comput. Commun.* 149 (2020), 99–106.

[42] S. Yang, B. Deng, J. Wang, H. Li, M. Lu, Y. Che, X. Wei, and K. A. Loparo. 2019. Scalable digital neuromorphic architecture for large-scale biophysically meaningful neural network with multi-compartment neurons. *IEEE Trans. Neural Netw. Learn. Syst.* 31, 1 (Mar. 2019), 148–62.

[43] S. Wan, Y. Xia, L. Qi, Y. H. Yang, and M. Atiquzzaman. 2020. Automated colorization of a grayscale image with seed points propagation. *IEEE Trans. Multimedia* 22, 7 (2020), 1756–1768.

Received July 2020; revised August 2020; accepted August 2020