Multi-objective Cross-layer Resource Scheduling for Internet of Things in Edge-Cloud Computing

Ruichao Mo†, Fei Dai‡, Qi Liu†, Wanchun Dou‖ and Xiaolong Xu*†,
†*School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China*
Email: ruichaomo@gmail.com; qi.liu@nuist.edu.cn
‡*School of Big Data and Intelligence Engineering, Southwest Forestry University, Kunming, China*
Email: flydai.cn@gmail.com
‖*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China*
Email: douwc@nju.edu.cn
**Corresponding author: X. Xu (Email: njuxlxu@gmail.com)*

***Abstract*—Nowadays, Edge computing is being introduced as a powerful paradigm to collaborate with the cloud to provide sufficient computing resources for IoT to implement intelligent analysis and data mining. Generally, due to the edge nodes are closer to mobile users, the access latency and the cost of using cloud services are effectively reduced. However, the implementation of cross-layer resource scheduling between edge nodes and servers deployed in the cloud to meet service requirements (i.e., shortest completion time, maximum resource utilization, lower energy consumption, etc.) still faces great challenges. To address this challenge, a cross-layer resource scheduling method, named CRSM, for the IoT applications is proposed in this paper. Technically, the Pareto archived evolution strategy (PAES) is employed to optimize the time cost of IoT applications, resource utilization and energy consumption of edge node. Then, the technique for order preference by similarity to ideal solution (TOPSIS) and the multiple criteria decision making (MCDM) are leveraged to acquired the optimal cross-layer resource scheduling strategy. Finally, the comprehensive analysis of CRSM is introduced in detail.**

***Keywords*-Edge-cloud computing; Cross-layer; IoT ; Resource scheduling; PAES**

I. Introduction

With the rapid development of mobile communication and Internet of things (IoT), numerous IoT devices (e.g., smartphones, laptops, sensors, etc.) are connected to the network through wireless network, and the network connection reliability of IoT devices is effectively guaranteed [1]. The intervention of wireless communication technology has greatly enhanced the storage capacity of the IoT, and greatly improved the storage and processing capabilities of data, thereby realizing the demands of intelligent analysis and data mining. Furthermore, the platform provided by the mobile network enables emerging IoT applications (e.g., smart cities, Internet of vehicles, etc.) to collect data generated by IoT devices in a timely manner through the wireless network and analyze the data for providing the end-users with high-quality services [2].

Due to the limited size of IoT devices, most IoT devices have limited computing power and battery life [3]. Therefore, the computing tasks generated on IoT applications need to be sent from the distributed IoT devices to the distant cloud. With the powerful computing and storage capabilities deployed on the cloud, data generated by IoT applications is effectively processed, thereby ensuring that IoT applications provide effective services to end-users [4]. However, as the number of IoT devices connected to the network continues to increase, the data generated by the devices will also increase dramatically, which puts great pressure on the bandwidth of cloud. In addition, the cloud is often deployed in a central location in the core network, resulting in a longer physical distance between the cloud and end-users, which brings transmission delays and additional energy consumption[5].

Edge computing (EC) is an emerging computing paradigm that provides better services by decentralizing infrastructure-based cloud resources (e.g., computing, storage, bandwidth, etc.) to the edge of the network [6]. Differ from the traditional cloud, the physical distance between IoT applications and edge nodes is greatly shortened by the introduction of EC, which effectively reduces the data transmission delay. Moreover, the reduction of physical distance also enables EC to provide users with ultra-high energy efficiency and ultra-high reliability services. At the same time, EC focuses on the analysis of real-time, short-cycle IoT data to better support the real-time intelligent processing and execution of various IoT applications [7].

In general, due to the limited computing and storage resources of each edge node and the large scale of data generated by IoT devices, even the implementation of resource scheduling among edge nodes at the edge of the network cannot meet the computing demands of IoT [8]. Therefore, it is necessary to implement cross-layer resource scheduling to leverage the resources in the cloud to supplement the demands of the edge node, which will effectively solve the problem of insufficient computing power at the edge. However, the computing resource structure of the cloud and edge nodes is heterogeneous and the geographical locations of the edge nodes are scattered, it is difficult to achieve

2159-6190/20/$31.00 ©2020 IEEE
DOI 10.1109/CLOUD49709.2020.00055

cross-layer resource scheduling between the cloud and the edge nodes while meeting the minimum completion time requirements of IoT [9].

On the other hand, as important evaluation criteria of the edge nodes, the resource utilization rate and energy consumption of the edge nodes also need to be considered in the process of resource scheduling. When the resource utilization of the edge nodes is unbalanced, the performance of the edge nodes will be affected. Futhermore, the energy consumption of a large number of deployed edge nodes needs to be optimized to reduce operating costs [10]. In general, while realizing cross-layer resource scheduling while ensuring the shortest completion time of IoT applications, maximum resource utilization and minimum energy consumption remain serious challenges. Therefore, a cross-layer resource scheduling method, named CRSM, is proposed in this paper to achieve joint optimization of completion time of IoT applications, resource utilization, and energy consumption.

Specifically, the main contributions of this paper are the following:

- The time cost model for IoT applications, the resource utilization model and the energy consumption model for edge nodes are designed.
- The pareto archived evolution strategy (PAES) [11] is leveraged to find the solution set of the cross-layer resource scheduling strategies with the shortest completion time for IoT applications, maximum resource utilization and minimum energy consumption for edge nodes.
- The technique for order preference by similarity to ideal solution (TOPSIS) and the multiple criteria decision making (MCDM) are utilized to obtain the optimal resource scheduling strategy.
- A large number of experiments are conducted to verify the effectiveness of CRSM.

The rest of this paper is organized as follows. In Section II, the related work is listed. Then, system model of cross-layer resource scheduling is defined in Section III. Futhermore, a cross-layer resource scheduling method is proposed in section IV. Experimental evaluation is presented in Section V. Finally, conclusion is drawn in Section VI.

II. Related Work

Nowadays, as numerous IoT devices are connected to the Internet, people's life becomes more intelligent [12]. For example, the maturity of technologies of Internet of Vehicles (IoV) makes people's travel more convenient and faster. However, as the number of IoT devices connected to the Internet continues to increase, it also puts a lot of bandwidth pressure on cloud that handle IoT devices [13]. As a computing example where computing resources are placed near the end-user, the introduction of EC enables a large amount of data generated by the IoT devices to be sent to the edge node for processing, thus reducing the load of the cloud while effectively processing the data.

However, due to the deployed the edge nodes have a wide variety of computing structures, it poses a significant challenge for edge devices to implement cross-layers resource scheduling. Therefore, according to the computing demands of IoT devices, cross-layer resource scheduling at the edge nodes will make full use of computing resources at the edge nodes.

In [14], She et al. establish a cross-layer architecture to optimize user affinity, packet offload rates, and bandwidth scheduling for mission-critical Internet of things (Mc-IoT) services with short packets. Moreover, an optimization algorithm is proposed to shorten the communication delay when the throughput of eMBB service is much higher than that of Mc-IoT service. In [15], Tang et al. describe the cross-layer resource scheduling problem as a mixed-integer nonlinear programming (MINLP) and propose a low Shaping-and-Pruning (SP) algorithm to obtain the sparse solution active RRH set, so that the cross-layer resource scheduling strategy reduce energy consumption effectively. In [16], lyu et al. propose a new integration architecture for the cloud, MEC, and Internet of things, as well as a lightweight request and allow framework to address network scalability issues.

Meanwhile, as the main evaluation criteria for moving edge computing, tasks completion time, energy consumption of edge servers and resource utilization of the edge nodes also need to be considered.

In [17], Zhang et al. propose a motion aware hierarchical MEC framework for green and low latency IoT. Moreover, a game theory-based computational offloading algorithm is proposed to optimize the utility of service providers while reducing the energy consumption and task execution time of smart devices. In [18], Chunlin et al. propose an energy-aware method for moving cloud resource allocation across layers. By taking advantage of the system context and mobile user preferences, the energy-aware mobile cloud resource allocation method is used across layers to optimize cloud resource consumption and system performance. In [19], to solve the problem of computation offloading and content cache strategy, Wang et al. considers computation offloading, resource allocation and content cache as an optimization problem. Furthermore, an alternate direction algorithm based on distributed convex optimization is proposed to solve the optimization problem. In [20], Tran et al. breaks down computational resource allocation on the edge nodes into resource allocation (RA) problems with fixed task offload decisions. In addition, a new heuristic algorithm using convex and quasi-convex optimization techniques is proposed to solve the problem of suboptimal solutions in polynomial time.

To the best of our knowledge, there are few studies on the cross-layer resource scheduling for IoT in Edge-Cloud computing. In addition, due to the importance of task

completion time, energy consumption of the edge nodes and resource utilization of the edge nodes, it is necessary to take them all into consideration. Therefore, a cross-layer resource scheduling algorithm is proposed in this paper.

III. System Model and Problem Formulation

In this section, the system model of cross-layer resource scheduling in edge-cloud is proposed. Meanwhile, the cross-layer resource scheduling problem is described as a multi-objective optimization problem. The key notations are given in Table.I.

Table I: Notations

Notation	Description
M	The amount of IoT applications
N	The amount of edge node
NA	The IoT application set $NA = \{na_1, na_2, ..., na_M\}$
TN	The computing task set $TN = \{tn_1, tn_2, ..., tn_M\}$
EN	The edge node set $EN = \{en_1, en_2, ..., en_N\}$
TC_m	The time cost of the task tn_m
ES	The energy consumption of all edge nodes
AU	The resource utilization of edge node
K	The number of the resource scheduling strategy
RS	The cross-layer resource scheduling strategy set $RS = \{rs_1, rs_2, ..., rs_K\}$

A. Cross-layer Resource Scheduling Architecture and Resource Model

In EC, the computing demands of the end-user are addressed by means of an edge nodes deployed close to the side of the user. However, due to the weak computing power of the edge nodes, when a computing task with a large amount of computation is encountered, a reasonable resource scheduling is required to ensure the execution of the computing task.

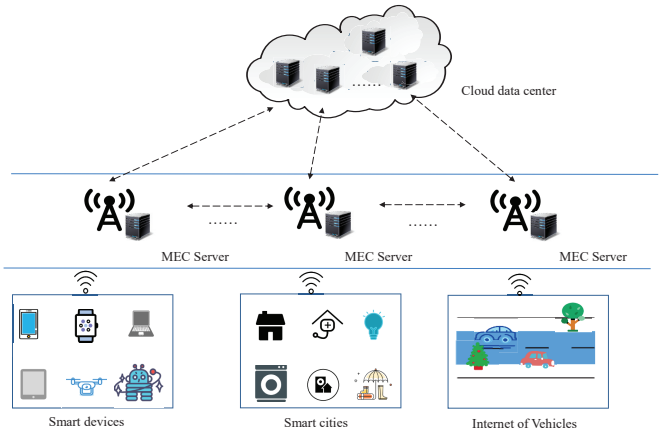

Figure 1: A cross-layer resource scheduling architecture for IoT in edge-cloud computing.

As shown in Fig.1, a variety of computing tasks through the wireless network is transmitted to the nearest the edge nodes to perform, when the computing resources of the edge node computing are not able to meet the task, it is necessary through the adjacent edge node for the resource scheduling to ensure effective execution of the tasks on the the edge nodes. Technically, assumed that the set of various applications is $NA = \{na_1, na_2, ..., na_M\}$, where $na_i (1 \leq i \leq M)$ is represented as an IoT application. Besides, assumed that each application generates only one computing task in the same time period, the set of computing tasks is denoted as $TN = \{tn_1, tn_2, ..., tn_M\}$, where tn_i is the computing task of is the computing task of na_i. Furthermore, assumed that the set of the edge nodes is $EN = \{en_1, en_2, ..., en_N\}$. In addition, the number of virtual machines contained in $en_n (1 \leq n \leq N)$ is defined as env_n.

B. Time Cost Model

The computing tasks generated in the IoT applications need to be transferred over the wireless network to the nearest edge node for processing. Therefore, the completion time of is divided into task transfer time DT_m , task completion time TE_m and result return time DF_m , which is calculated by

$$TC_m = DT_m + TE_m + DF_m. \tag{1}$$

Assume that the wireless network bandwidth between the IoT device and the edge node is BW_p, the network bandwidth among edge nodes is BW_q and the network bandwidth between cloud and the edge nodes is BW_r. Moreover, the variable i is introduced to represent three kinds of resource scheduling of computing task at the edge node: 1) i = 0 represents the computing resources that the edge node satisfy the task; 2) i = 1 means that the computing resources of the edge node cannot meet the task, and resources are allocated from other nodes to meet the task; 3) i = 2 means that the computing resources of the edge node cannot meet the task, and resources are allocated from the cloud to meet the task.

Thus, assume that the data size of task tn_m is D_m , the required transmission time of the task is measured by

$$DT_m = \begin{cases} \frac{D_m}{BW_p}, i = 0, \\ \frac{D_m}{BW_p} + \frac{D_m}{BW_q}, i = 1, \\ \frac{D_m}{BW_p} + \frac{D_m}{BW_q} + \frac{D_m}{BW_r}, i = 2. \end{cases} \tag{2}$$

Based on the above analysis, assume that the data size of the returned result is R_m the return time is calculated by

$$DF_m = \begin{cases} \frac{R_m}{BW_p}, i = 0, \\ \frac{R_m}{BW_p} + \frac{R_m}{BW_q}, i = 1, \\ \frac{R_m}{BW_p} + \frac{R_m}{BW_q} + \frac{R_m}{BW_r}, i = 2. \end{cases} \tag{3}$$

Furthermore, due to the computing performance of virtual machines in edge nodes is consistent, assuming that the

data processing capacity of en_n is χ_n, the execution time required by task tn_m is measured by

$$TE_m = \sum_{n=1}^{N} \frac{D_m \cdot f(en_n)}{\chi_n}. \tag{4}$$

where $f(en_n)$ is denoted as a binary variable to judge the en_n is occupied, which is measured by

$$f(en_n) = \begin{cases} 1, \text{if } en_n \text{ is occupied,} \\ 0, \text{otherwise.} \end{cases} \tag{5}$$

C. Resource Utilization Model

As an important evaluation criterion, the resource utilization of each edge node needs to be considered in resource scheduling. In this paper, the virtual machine usage contained in each edge node is used to evaluate the resource utilization of the edge node. Technically, supposed that the virtual machines size needed to execute $tn_m(1 \leq m \leq M)$ is nv_m. Thus, the utilization au_n of $en_n(1 \leq n \leq N)$ is calculated by

$$au_n = \frac{1}{env_m} \sum_{i=0}^{env_m - 1} \theta(tn_m) \cdot nv_m. \tag{6}$$

Denoted $\theta(tn_m)$ as a binary variable aiming to estimate whether the tn_m is deployed on en_n which is measured by

$$\theta(tn_m) = \begin{cases} 1, \text{if } tn_m \text{ runs on } en_n, \\ 0, \text{otherwise.} \end{cases} \tag{7}$$

Furthermore, ZR is represented as the total amount of occupied virtual machines, which is measured by

$$ZR = \sum_{i=0}^{N} env_m \cdot f(en_n). \tag{8}$$

Finally, the resource utilization AU of the edge nodes is deduced by

$$AU = \frac{1}{ZR} \sum_{n=1}^{N} au_n. \tag{9}$$

D. Energy Consumption Model

In this paper, the energy consumption of the edge nodes is mainly generated in three aspects: the basic operating energy consumption of the edge nodes, the energy consumption of full load VMs and the energy consumption of no-load VMs. Among them, the energy consumption of full load virtual machine depends on the complexity of computing task, that is, the time required to complete the task. The maximum task completion time of the edge node is recorded as the working time, which is measured by

$$wt_n = \max_{m=1}^{M} (\theta(tn_m) \cdot et_m). \tag{10}$$

Therefore, the basic operating energy consumption of the edge nodes is calculated by

$$BCS = \sum_{n=1}^{N} wt_n \cdot \alpha, \tag{11}$$

where α is utilized to represent the power rate of edge node.

Furthermore, the energy consumption of full load VMs is measured by

$$FCS = \sum_{n=1}^{N} \sum_{m=1}^{M} \theta(tn_m) \cdot wt_n \cdot \gamma, \tag{12}$$

where γ represents the power rate of full load virtual machine.

The energy consumption of no-load VMs is acquired by

$$NCS = \sum_{n=1}^{N} (ZR - \sum_{m=1}^{M} \theta(tn_m)) \cdot wt_n \cdot \varepsilon, \tag{13}$$

where ε is used to represent the power rate of no-load VMs.

Finally, the energy consumption of all edge nodes is calculated by

$$ES = BCS + FCS + NCS. \tag{14}$$

E. Problem Formulation

The mutil-objective cross-layer resource scheduling problem for IoT applications in edge node is focused on optimizing the time cost, resource utilization and the energy consumption of all edge nodes. Thus, the problem to be solved in this paper is defined by

$$min\ TC_m, ES. \tag{15}$$

$$max\ AU. \tag{16}$$

$$s.t. \forall \theta(tn_m) = 1, en_n \in EN \tag{17}$$

$$\sum_{i=1}^{M} \theta(tn_i) \cdot nv_i \leq en_n. \tag{18}$$

IV. Cross-layer Resources Scheduling Method Using PAES

In this paper, the cross-layer resource scheduling method (CRSM) is defined as a multi-objective optimization problem, which ensure the shortest completion time of tasks from the IoT devices, the lowest energy consumption of edge nodes and the maximum resource utilization of edge nodes. To solve the multi-objective resource scheduling problem, PAES, as a multi-objective optimization algorithm, is applied in the cross-layer resource scheduling. Compared with the traditional multi-objective optimization algorithm, the Pareto Front is acquired faster by PAES. Moreover, the algorithm complexity of PAES is lower.

A. Chromosome Representation

In multi-objective genetic algorithms, the goals of a problem are usually defined as genes and these genes are made up of chromosomes to find solutions that satisfy multiple targets. In this paper, while solving the problem of cross-layer resource scheduling, the goals of shortest task completion time, maximum resource utilization of the edge nodes and minimum energy consumption of edge nodes are satisfied. Therefore, the three goals of the shortest completion time of the computing task, the highest resource utilization and the lowest energy consumption of the edge node are defined as genes and constitute a chromosome. Furthermore, gene operation works on chromosomes to quickly find cross-layer resource scheduling strategies. On this basis, the cross-layer resource scheduling strategy set is represent as $RS = \{rs_1, rs_2, ..., rs_K\}$.

B. Fitness Function and Constraints

In PAES, fitness functions are used to determine the fitness of chromosomes in a population. Therefore, the chromosomes in the population were preliminarily screened by fitness function. In addition, various constraints have been used to limit the evolution of chromosomes in populations. In this paper, task completion time, resource utilization and energy consumption of edge nodes are taken as fitness functions and expressed by (15) and (16). Moreover, in the process of resource scheduling, the constraint that the number of resources required by the task should not exceed the resources owned on the edge node is expressed by (17) and (18).

C. Selection

During the selection process, PAES generates the evolutionary population and the elite population through the initialization operator, while the appropriate elite strength parameters are also obtained. Then evolutionary and elite populations compete to produce the next generation. Moreover, in the process of evolution, the scale and degree of elite individuals' participation will be limited by elite intensity, the sampling process is controlled by sampling operators, and vary operators generate the next generation of individuals through crossover and mutation.

By using the elite retention strategy, the dominant solution with good quality produced by PAES algorithm is preserved, effectively avoiding the loss of excellent individuals, and maintaining the diversity of the population through crossover and mutation. In addition, in the search process of PAES, the external solution set approximation is taken as the current non-dominant frontier, providing auxiliary information for the selection process of parent and child individuals, thus guiding the process to produce more excellent individuals.

D. Iteration

As an important part of PAES, iteration is utilized to make a choice between the current solution and the mutated solution. In addition, external files are updated according to the situation of the candidate solution. If the candidate solution of the resource scheduling is dominated by any solution in the file, the solution is deleted to ensure that the solution in the file is not dominated by the candidate solution. In PAES, the number of solutions in the archive will be in a dynamic state during the iteration process, so the size of the grid will be adjusted automatically during the iteration, and the location will be repositioned according to the distribution of solutions, so as to ensure that each solution is in a certain grid.

E. Optimal Cross-layer Resource Scheduling Strategy Acquired

In fact, there will be multiple cross-layer resource scheduling strategies generated during the CRSM iteration. Therefore, the MCDM and TOPSIS will be utilized to obtain the optimal cross-layer resource scheduling strategy after the iteration. In TOPSIS, the order is made by discriminating the distance between the optimal solution and the worst solution of the evaluation individual. If the evaluation individual is closest to the optimal solution and farthest from the worst solution, the evaluation individual is regarded as the best one; otherwise, the evaluation individual is regarded as the worst one.

As the cross-layer resource scheduling strategy set obtained by the algorithm, there are K solutions in RS, each solution contains the values of task completion time, resource utilization and energy consumption, and the corresponding values of the strategy are expressed as $TC = \{tc_1, tc_2, ..., tc_K\}$, $AU = \{au_1, au_2, ..., au_K\}$ and $DT = \{dt_1, dt_2, ..., dt_K\}$. The optimal cross-layer resource scheduling strategy is obtained through the following operations in TOPSIS.

Firstly, the task completion time for all solutions is processed as follows to complete the normalization

$$WN_i^T = \frac{tc_i}{\sqrt{\sum_{i=1}^{K} tc_i}}. \tag{19}$$

Besides, the resource utilization of the edge nodes and energy consumption of all solutions are regularized as above to obtain WN_i^U and WN_i^E.

Supposed that the weights of task completion time, resource utilization and energy consumption are expressed as ρ_t, ρ_u and ρ_e. Then the weighted normalization value of task completion time is calculated by

$$WNE_i^T = \rho_t \cdot WN_i^T. \tag{20}$$

Furthermore, the weighted regularization value of the resource utilization and energy consumption are acquired as WNE_i^U and WNE_i^E respectively.

Then, the distance from each target to the ideal solution is calculated, and the range scale is calculated by the Euclidean distance. The distance from the target to the ideal solution of DSD_k^T is calculated as

$$\begin{aligned} DSD_k = &\sqrt{(WNE_k^T - \max_{K=1}^{K}(WNE_k^T))} \\ &+ \sqrt{(WNE_k^U - \max_{K=1}^{K}(WNE_k^U))} \\ &+ \sqrt{(WNE_k^E - \max_{K=1}^{K}(WNE_k^E))}. \end{aligned} \tag{21}$$

Moreover, the distance of the anti-ideal solution is measured by

$$\begin{aligned} DSF_k = &\sqrt{(WNE_k^T - \min_{K=1}^{K}(WNE_k^T))} \\ &+ \sqrt{(WNE_k^U - \min_{K=1}^{K}(WNE_k^U))} \\ &+ \sqrt{(WNE_k^E - \min_{K=1}^{K}(WNE_k^E))}. \end{aligned} \tag{22}$$

Finally, the closeness degree of the ideal solution is calculated as

$$CD_k = \frac{DSF_k}{DSD_k + DSF_k}. \tag{23}$$

According to the degree of closeness of the ideal solution, the ranking result with the largest degree of closeness is regarded as the optimal resource scheduling strategy.

F. Method Overview of CRSM

In this paper, CRSM is divided into four steps, the generation of candidate solutions for cross-layer resource scheduling, the selection of candidate solutions, the utilization of the non-dominated solutions and the acquisition of the optimal solution. Initially, CRSM produces a solution by simply randomly mutation and the goal value of the initial solution is calculated and placed into the file that has been set. Then, a new solution is generated by mutating the single parent solution, and the target value of the new solution is also calculated. There will be two scenarios: 1) If the new solution is dominated by the parent solution, the new solution will be generated again by mutating the parent solution; 2) If it is not dominated, the new solution is compared with other solutions in the file, the file is updated through iteration, and a solution is randomly selected in the file as the new parent solution. Furthermore, CRSM iteration process ends at the end of the set number of iterations. Finally, the optimal cross-layer resource scheduling strategy is obtained by MCDM and TOPSIS. The specific flow of the CRSM is shown in Algorithm.1.

Algorithm 1 Cross-layer resource scheduling method using PAES.

Ensure: The resource allocation strategies set RS for the IoT applications.

1: Randomly generate the set of parent solutions.
2: Calculates the parent solution and puts the target value into the file.
3: **while** Iteration not finished. **do**
4: The parent solution performs the mutation to produce the offspring.
5: Calculate offspring solution.
6: **if** The descendant solution is not dominated by the parent solution. **then**
7: Compare the dominance of subsolutions with other solutions in the file.
8: Iteration to continue.
9: Select a solution in the file as the parent solution.
10: **else** Continue.
11: **end if**
12: **end while**
13: Optimal resource allocation strategy obtain by MCDM and TOPSIS.
14: **return** RS

V. Comparison and Analysis of Experimental Results

In this section, the performance of CRSM is evaluated. Firstly, the simulation settings of this experiment are given. Then, three resource scheduling algorithms are leveraged for comparison. Finally, the results of the experiment are analyzed.

A. Simulation Setup

In this paper, simulation experiments are conducted to verify the performance of CRSM. Specifically, the simulation experiment is performed on a laptop with Intel Core i7-5500U, 2.40GHZ CPU, 16 GB RAM and Window 10. In addition, the specific setup in this simulation is given in Table II.

Table II: Parameter Settings

Parameter	Value
The amount of IoT applications	50,100,150,200,250
The amount of VMs on each edge node	9
The amount of VMs required for IoT applications	[1,10]
The power rate of the edge node	330W
The power rate of employed VM instances	65W
The power rate of unemployed VM instances	25W

B. Compared Algorithms

To illustrate the performance of the CRSM, three resource scheduling algorithms are selected for comparison, i.e.,

Benchmark, FFD-C, and BFD-C. And the basic principles of these three methods are listed.

- Benchmark: The scheduling strategy for IoT application is randomly generated by Benchmark, but when the capacity of the edge node is not enough to meet the capacity required for the IoT application, the resource scheduling strategies will be randomly generated until the optimal strategy is acquired.
- First fit decreasing-based cross-layer resource scheduling method (FFD-C): All edge node are first sorted in ascending order according to their remaining capacity. Then, the IoT applications are allocated to edge node that meet their required capacity in ascending order.
- Best fit decreasing-based cross-layer resource scheduling method (BFD-C): In BFD-C, all edge nodes are sorted by their capacity. Then, when BFD-C finds the resource scheduling strategy for the IoT application, BFD-C will select the edge node with the smallest capacity but satisfy the IoT application. The method ends after all IoT applications have found a edge node for execution.

C. Performance Evaluation

1) Comparison of completion time for IoT application: In this paper, the completion time of the IoT applications is the key evaluation criterion for evaluating the resource scheduling method. The experimental results of the Benchmark, BFD-C, FFD-C, and CRSM are shown in Fig.2. Compared with the Benchmark, BFD-C, FFD-C, the resource scheduling strategy found by CRSM makes the IoT application completion time significantly shorter than other methods in the case of IoT applications of different scales.

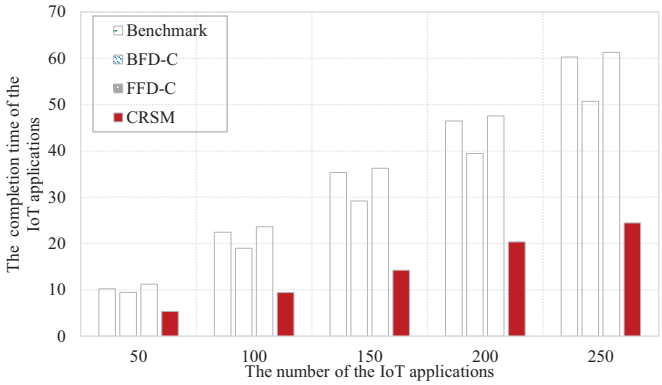

Figure 2: Comparison of the IoT application completion time by Benchmark, BFD-C, FFD-C, and CRSM.

2) Comparison of resource utilization: Resource utilization is an important indicator of the edge nodes by calculating the number of VMs occupied in each node. Fig.3 compares the resource utilization of Benchmark, FFD-C, BFD-C, and CRSM in different number of IoT applications. As is shown in Fig.3, CRSM maintains a higher resource utilization than the other three methods. In other words, compared with Benchmark, FFD-C, and BFD-C, CRSM reduces the number of free VMs, thereby reducing the energy consumption of edge node to a certain extent.

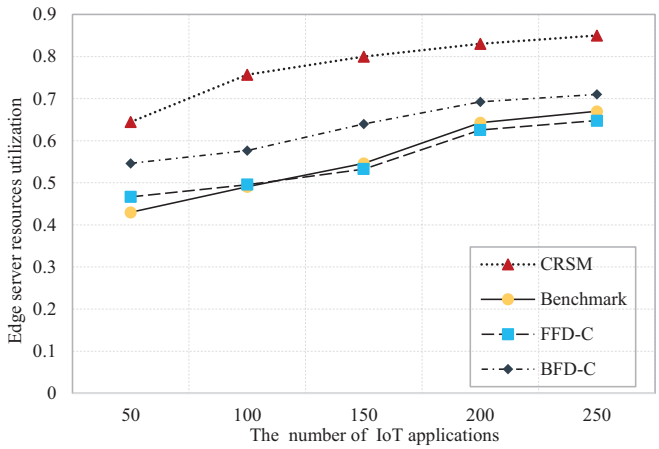

Figure 3: Comparison of the resources utilization by Benchmark, BFD-C, FFD-C, and CRSM.

3) Comparison of the total energy consumption: The total edge node energy consumption is analyzed and compared. As shown in Fig.4, with the increasing number of IoT applications, the total energy consumption of edge nodes is on the rise due to resource scheduling by Benchmark, BFD-C, FFD-C, and CRSM for IoT applications. However, compared with Benchmark, BFD-C and FFD-C, the resource scheduling strategy obtained by CRSM ensures the lowest energy consumption.

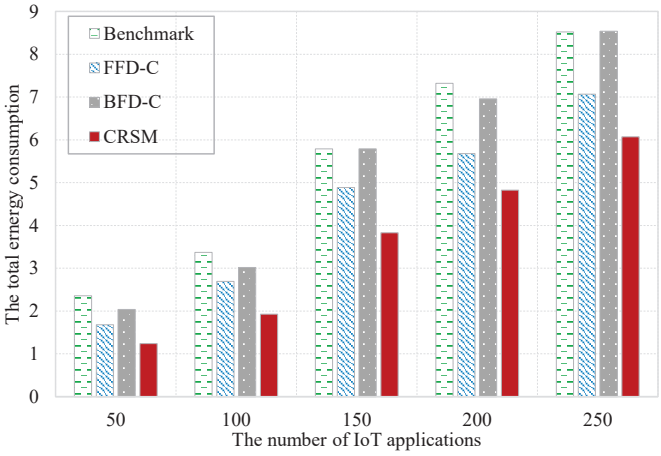

Figure 4: Comparison of the total energy consumption of the VMs by Benchmark, BFD-C, FFD-C, and CRSM.

VI. Conclusion

In this paper, to achieve effective cross-layer resource scheduling for IoT applications in EC, a multi-objective cross-layer resource scheduling method (CRSM) based on PAES was proposed in this paper. Technically, PAES was leveraged to identify cross-layer resource scheduling strategies that meet the minimum completion time, the maximum resource utilization, and the minimum energy consumption. Then, The TOPSIS and MCDM were utilized to acquire the optimal resource scheduling strategy. Finally, a large number of experiments were conducted to verify the effectiveness of CRSM.

Acknowledgment

This research is supported by the National Natural Science Foundation of China under grant no. 61672276, no. 61702442, no. 61702277, no. 61872219, the National Key Research and Development Program of China (No. 2017YFB1400600 and No. 2019YFE0190500), and the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps under grant no. 2020DB005.

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[2] X. Xu, C. He, Z. Xu, L. Qi, S. Wan, and M. Z. A. Bhuiyan, "Joint optimization of offloading utility and privacy for edge computing enabled iot," *IEEE Internet of Things Journal*, 2019.

[3] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2017.

[4] H.-C. Hsieh, J.-L. Chen, and A. Benslimane, "5g virtualized multi-access edge computing platform for iot applications," *Journal of Network and Computer Applications*, vol. 115, pp. 94–102, 2018.

[5] X. Xu, Y. Xue, L. Qi, Y. Yuan, X. Zhang, T. Umer, and S. Wan, "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.

[6] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for uav-assisted mobile edge computing in iot," *IEEE Transactions on Industrial Informatics*, 2019.

[7] X. Li, J. Wan, H.-N. Dai, M. Imran, M. Xia, and A. Celesti, "A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4225–4234, 2019.

[8] X. Xu, Y. Xue, L. Qi, X. Zhang, S. Wan, W. Dou, and V. Chang, "Load-aware edge server placement for mobile edge computing in 5g networks," in *International Conference on Service-Oriented Computing*. Springer, 2019, pp. 494–507.

[9] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[10] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6774–6785, 2019.

[11] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.

[12] Q. Wang, S. Guo, J. Liu, and Y. Yang, "Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing," *Sustainable Computing: Informatics and Systems*, vol. 21, pp. 154–164, 2019.

[13] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for iot-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.

[14] C. She, Y. Duan, G. Zhao, T. Q. Quek, Y. Li, and B. Vucetic, "Cross-layer design for mission-critical iot in mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9360–9374, 2019.

[15] J. Tang, W. P. Tay, and T. Q. Quek, "Cross-layer resource allocation with elastic service scaling in cloud radio access network," *IEEE Transactions on Wireless Communications*, vol. 14, no. 9, pp. 5068–5081, 2015.

[16] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, "Selective offloading in mobile edge computing for the green internet of things," *IEEE Network*, vol. 32, no. 1, pp. 54–60, 2018.

[17] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Mobile edge computing and networking for green and low-latency internet of things," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 39–45, 2018.

[18] L. Chunlin, L. Yanpei, and L. Youlong, "Energy-aware cross-layer resource allocation in mobile cloud," *International Journal of Communication Systems*, vol. 30, no. 12, p. e3258, 2017.

[19] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[20] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2018.