A Multi-objective Optimization Method for Latency-Sensitive Applications in MEC-Enabled Smart Campus Using SMS-EMOA

Hanmeng Wang[1,2], Kai Peng[1,2(✉)], and Bohai Zhao[1,2]

[1] Huaqiao University, Quanzhou, China
[2] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, People's Republic of China
kai.peng@hqu.edu.cn

Abstract. The Internet of Things (IoT) enables all devices to sense, communicate, which has given rise to the evolution of traditional educational planning. Benefiting from this revolution, a new paradigm, named smart campus came into being. More specifically, teaching way has changed significantly with the help of quite a few emerging real-time applications. However, the computing capabilities of smart devices (SDs) are constrained. Fortunately, mobile edge computing (MEC) has emerged as a scalable computing model to augment capacity of SDs by placing computing resources closer to users. However, the resources of the ESs in MEC are not inexhaustible. In view of this, we investigate computation offloading for latency-sensitive application which is modeled as workflow application in MEC-enabled smart campuses. Specifically, time consumption, energy consumption, and resource utilization are regarded as our optimized goals. To this end, a corresponding method is proposed. Extensive experimental results show that our method can achieve effective optimization effect in terms of given objectives.

Keywords: Smart Campus · Mobile Edge Computing · Workflow Applications · Multi-Objective Optimization

This work is supported by the National Science Foundation of China under Grant No. 61902133, the Fundamental Research Funds for the Central Universities under Grant No. ZQN-817, Quanzhou Science and Technology Project under Grant No. 2020C050R, Huaqiao University 2021 Teacher Teaching Development Reform Project under Grant No. HQJG202120.

Y. Cao and X. Shao (Eds.): MONAMI 2022, LNICST 474, pp. 63–77, 2023.
https://doi.org/10.1007/978-3-031-32443-7_5

1 Introduction

Combining the collected data with sensor devices enables people to experience intelligent services, realize information exchange through different media, and thus connect everything, this is Internet of Things(IoT) [1,2]. IoT has spawned many new applications, among them, smart campus is one of the most popular representatives [3–5].

Generally, smart campus is regarded as an information-based instance, which can provide networked teaching, research, management, and life services for students and teachers by collecting, integrating, and utilizing digital information. Different from the traditional model, smart campus facilitates smart devices (SDs), sensors, and campus servers to provide more accurate and timely information services, thereby improving school operations' efficiency [6–8]. In addition, with the development of IoT technologies, such as virtualization, wireless communication and RFID, it is credible that smart campus has ushered in its historical moment [9,10].

Generally, most campus networks mainly use large-scale cloud-like centralized service, this kind of service is simple and effective for the traditional campus and has been widely used for many years [11–13]. However, with the popularization of education and the expansion of electronic devices, the disadvantages of centralized service are gradually reflected. On the one hand, more and more SDs are flooding into the campus network, which means that the threshold of the central server load must be constantly breached, and thus requires continuous and cumbersome maintenance [14,15]. On the other hand, the bandwidth allocated to each device will drop rapidly as the number of SDs increases. To sum up, it is challenging to address above shortcomings [16,17].

Fortunately, the advent of mobile edge computing (MEC) offers hope for solving this problem [18]. In MEC scenario, edge servers (ESs) are usually arranged at the edge of the network closer to SDs to provide low response latency and high-quality services. Applications generated by mobile users (MUs) are allowed to be offloaded to the nearest ES for processing. Nevertheless, ESs are characterized as heterogeneous and resource-constrained. In addition, the coverage of ES services is also limited. Therefore, how to select the most suitable ES is a critical challenge.

In view of aforementioned description and challenge, we study the computation offloading for latency-sensitive applications in MEC-enabled smart campus scenario. The main contributions of this paper are summarized as follows.

- Firstly, the latency-sensitive applications are represented as sequential constrained workflow applications. In addition, computation offloading for workflow application is formulated as a multi-objective optimization issue where time consumption, energy consumption, and resource utilization are considered as the optimization objectives.
- Secondly, a corresponding system model and mathematical model are established. In what follows, a method based on overcapacity non-dominated sorting, called MOWASC, is proposed to get the optimal offloading strategy.

– Finally, we evaluate the advantages of our method through experimental tests under different scenarios. Extensive experiments have verified that our method can effectively reduce time and energy consumption, as well as improve the resource utilization of ESs.

The forthcoming portions of this paper is described as follows. First, we discuss the related work. Next, the definition of the computational model and optimization problems are illustrated. And then, the method and the related experimental evaluation are described. Finally, the conclusion and our future work are summarized.

2 Related Work

In this section, the existing work of computation offloading for general applications as well as workflow applications are reviewed, respectively.

Computation Offloading for General Applications. Computation offloading is a potent technology that can boost computing capacity while lessening the demand for MUs. Chang et al. [19] dynamically migrated the task to the edge fog node for execution to decrease service delay and energy usage. On the basis of hierarchical MEC networks, Li et al. [20] expanded the auxiliary cloudlets collaborative computing and relieved the user's operating costs. The complexity of the network architecture may lead to resource contention and unequal allocation. [21] focused on the resource allocation in MEC network, and the resource utilization and load balancing are framed as optimization goals in a heuristic algorithm. An online dynamic task assignment scheme was employed in [22], which can allocate resources dynamically, resulting in high-efficiency and low-latency communication. Similar contemporary techniques were used with MEC to increase computing effectiveness. Huang et al. [23] utilized the benefits of deep reinforcement learning to serve multiple wireless devices, which improves the quality of service for the MEC network.

Computation Offloading for Workflow Applications. Computation offloading strategies for general tasks cannot be directly applied to latency-sensitive and task-dependent items. Huang et al. [24] put data compilation into service, and customized differentiated offloading schemes for data-dependent and latency-sensitive programs. For the scheduling problem of workflow, Sun et al. [25] formulated the problem as an integer question and combined two algorithms to shorten the manufacturing time of workflow. Xu et al. [26] studied the computation offloading of workflow applications in cloud environments, which migrates tasks to the cloud and cloudlets to save energy consumption of SDs. Due to the particularity of the items in workflow, the completion time is attentively considered in the research. Ma et al. [27] proposed a deadline-constrained workflow service scheme that minimizes the application execution cost according to a cost-aware scheduling algorithm.

Different from the existing studies, in this paper, we focus on the computation offloading for latency-sensitive applications in MEC-enabled smart campus. Additionally, this study considers the dependence between sub-task while the time consumption, energy consumption, and resource utilization are jointly optimized.

3 System Model and Problem Formulation

In this section, the MEC-enabled smart campus system model is introduced firstly, followed by the description of workflow application, and then mathematical models for each optimization objective are introduced.

3.1 System Model

Figure 1 depicts the structure of the MEC-enabled smart campus, which guarantees the integration of management, teaching, and scientific research. Some MUs and infrastructures are included in the smart campus for teaching activities and campus management. The general MUs are students and teachers, who usually use mobile phones and laptops for socializing and learning online. Same for administrators, who frequently employ intelligent network equipment to monitor campus governance and network security issues. Among them, not only can MUs use the local area network (LAN) for data transfer with surrounding ESs, but they can interact with the Cloud Center (CC) over the wide area network (WAN).

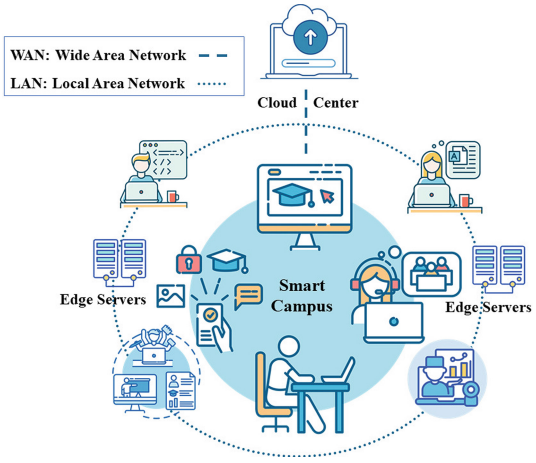


Fig. 1. An MEC-Enabled Smart Campus Architecture.

3.2 Workflow Application

One special feature, the multiple data dependencies that form a complicated application which can be reflected as a directed acyclic graph (DAG). Figure 2 shows a workflow consisting of eight items (I_1-I_8), they have different demands on resources. The weight of each edge represents the amount of data transferred between two tasks, while the order of task execution can be defined as a set $Ord = \{\ I_1,\ (I_2,\ I_3,\ I_4),\ I_5,\ (I_6,\ I_7),\ I_8\ \}$.

It can be seen from Fig. 2, except for the last item I_8, after each task is completed, some information will be transmitted to guarantee the next calculation, which is a special feature of workflow applications. After item I_1 is executed, 25, 15, and 20 pieces of data will be transferred to I_2, I_3, and I_4, respectively. After the calculation of I_2, I_3, and I_4 is finished, 30, 20, and 45 pieces of data will be transmitted to I_5, respectively. Similarly, after I_6, and I_7 are executed, the corresponding data will be transmitted to item I_8.

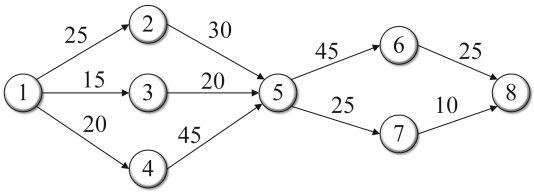


Fig. 2. An example of Workflow Application.

For those nodes that need to receive the data, they will work when all messages from their predecessor items are collected. When I_5 only gets one or two data among I_2, I_3, and I_4, it means that the preparation work for I_5 has not been completed. Likewise, I_8 only can start when I_8 receive complete data from I_6 and I_7.

P_i^w denotes a task to be processed, in which the value of w is the index of workflow, and i represents the index of the items in the workflow. The offloading strategies of tasks can be defined as S_i^w as follows.

$$S_i^w = \begin{cases} 0, & \text{if } \mathrm{P_i^w} \text{ is executed on SDs,} \\ 1, ..., C, & \text{if } \mathrm{P_i^w} \text{ is offloaded to ESs,} \\ C+1, & \text{if } \mathrm{P_i^w} \text{ is offloaded to CC.} \end{cases} \tag{1}$$

$S_i^w = 0$ means that the computation task is executed on the SDs while $\{\ 1, \ldots,$ C $\}$ represents that task is offloaded to the ESs and $C+1$ indicates that task is migrated to the CC for execution.

3.3 Time Consumption Model

The system time consumption T_{sum} is introduced in this subsection, which is made up of task transmission time T_{tra}, queue waiting time T_{que} and task execution time T_{exe}.

Transmission Time. The time for tasks to migrate to the executing side and send processed data to the successor node constitutes task transmission time, which can be calculated as

$$T_{tra} = T_{tra}(P_a^w) + \frac{D_{a,b}}{B_{a,b}}. \tag{2}$$

The offloading time $T_{tra}(P_a^w)$ is the delay in data transmission when the task arrives at the associated server, which varies depending on where the work is carried out.

$$T_{tra}(P_a^w) = \begin{cases} 0, & \mathrm{S_i^w} = 0, \\ L_{LAN}, & \mathrm{S_i^w} \subseteq \{1, ...\mathrm{C}\}, \\ L_{WAN}, & \mathrm{S_i^w} = \mathrm{C}+1, \end{cases} \tag{3}$$

where L_{LAN} represents the latency from SDs to ESs through the LAN and L_{WAN} indicates the latency from SDs to CC over the WAN.

The time for information transfer is related to the size of the data $D_{a,b}$ and the bandwidth of the transmission path $B_{a,b}$. The specific bandwidth will change with different offloading modes, which can be categorized into the following three groups.

$$B_{a,b} = \begin{cases} \infty, & \mathrm{S_a^w} = \mathrm{S_b^w}, \\ B_{ee}, & \mathrm{S_a^w} \subseteq \{1, ...\mathrm{C}\}, \ \mathrm{S_b^w} \subseteq \{1, ...\mathrm{C}\}, \\ B_{ce}, & \mathrm{S_a^w} \subseteq \{1, ...\mathrm{C}\}, \ \mathrm{S_b^w} = \mathrm{C}+1. \end{cases} \tag{4}$$

Among them, Bee is the bandwidth between ESs. When the task in ESs transfer data to CC, the packets will be transmitted through the WAN, and the bandwidth is Bce at this time.

Queue Waiting Time. ESs have an advantage in physical distance compared to the CC, which can reduce round-trip latency when tasks are offloaded. When many tasks are migrated to ESs, there is a need to queue for available resources if all the current resources are occupied, as resources are limited. The time consumed by queuing is called the queue waiting time.

Tasks typically arrive in queues at random. According to the queuing theory [28], the average queue time T_{que} is used to complete the model.

M parallel-running virtual machines (VMs), each with a service time that obeys negative exponential distribution with λ_s, are set up. Correspondingly, the time slots that tasks arrive successively obey the negative exponential distribution with λ_n. The average queue length L_a is composed of the average length of the line-up plus the average number of customers currently serving. The specific calculation of L_a is

$$L_a = \sum_{N=M+1}^{\infty} (N-M)P_N + \frac{\lambda_n}{\lambda_s}, \tag{5}$$

where P_N is the probability distribution state of the queue length N after the system reaches the equilibrium state. The average number of customers being served can be obtained by $\frac{\lambda_n}{\lambda_s}$. In summary, the average waiting time T_{que} can be calculated as

$$T_{que} = \frac{L_a}{\lambda_n} - \frac{1}{\lambda_s}. \tag{6}$$

Task Execution Time. The execution time T_{exe} is related to the amount of data and the computing rate of the device, which is calculated as

$$T_{exe} = \begin{cases} \frac{D_i^w}{H_e}, & \mathrm{S_i^w} \subseteq \{1, ...\mathrm{C}\}, \\ \frac{D_i^w}{H_c}, & \mathrm{S_i^w} = \mathrm{C}+1, \end{cases} \tag{7}$$

where D_i^w indicates the size of data while H_c and H_e represent the data processing capabilities of CC and ESs.

Total Time Consumption. After the above analysis and calculation of time consumption, the total time consumption T_{sum} can be expressed as

$$T_{sum} = \begin{cases} T_{tra} + T_{que} + T_{exe}, \mathrm{S_i^w} \subseteq \{1, ...\mathrm{C}\}, \\ T_{tra} + T_{exe}, \qquad \text{otherwise.} \end{cases} \tag{8}$$

3.4 Energy Consumption Model

As energy consumption usually has a positive correlation with time consumption, and thus the total energy consumption of the system E_{sum} can be expressed as

$$E_{sum} = \begin{cases} E_{tra} + E_{que} + E_{exe}, \mathrm{S_i^w} \subseteq \{1, ...\mathrm{C}\}, \\ E_{tra} + E_{exe}, \qquad \text{otherwise.} \end{cases} \tag{9}$$

Among them, E_{tra}, E_{que} and E_{exe} represent the energy consumption of task transmission, queuing, and execution, respectively.

3.5 Resource Utilization Model

Resource utilization, obtained by the number of VMs currently being occupied in the VM pool, reflects the usage of ES resources. The k-th VM VM_k in the VM pool has completed T tasks, and its resource utilization R_k is

$$R_k = \frac{1}{T} \cdot \sum_{w=1}^{W} \sum_{i=1}^{I} V_{w,i} \cdot F_{w,i}^k, \tag{10}$$

where $V_{w,i}$ is the VM workload occupied by P_i^w, and $F_{w,i}^k$ is a flag to determine whether the workload has been offloaded to the k-th ES.

$$F_{w,i}^k = \begin{cases} 1, \text{ if } \mathrm{V_{w,i}} \text{ offloaded to the } \mathrm{k-th} \text{ ES}, \\ 0, \qquad\qquad \text{otherwise.} \end{cases} \tag{11}$$

By calculating the resource utilization of each ES, the average resource utilization A_R of the ESs can be expressed as

$$A_R = \frac{1}{E_E} \cdot \sum_{k=1}^{K} R_k, \tag{12}$$

where E_E, the number of ESs occupied, can be defined as follows.

$$E_E = \sum_{k=1}^{K} FE, \tag{13}$$

where FE is a flag to determine the status of ES.

$$FE = \begin{cases} 1, \text{ if the } \mathrm{k-th} \text{ ES is employed}, \\ 0, \qquad\qquad \text{otherwise.} \end{cases} \tag{14}$$

4 Algorithm Design

In this section, the basic framework and core steps of the multi-objective optimization method for latency-sensitive applications in MEC-Enabled Smart Campus (MOWASC)are introduced. First, task migration strategies are generated using SMS-EMOA. Then, the normalized fitness value is used to determine the optimal placement strategy.

4.1 Initialization

Some fundamental parameters need to be set for the algorithm, such as the population settings, the maximum number of evaluations, the probability and distribution indices for mutation and crossover, the comparator dominance comparator, and the number of matches. In this paper, genes symbolize the task performed site and chromosomes represent the computational offloading scheme. Number 0 means the task is executed locally, 9 for CC, and number 1 to number 8 represents the task is executed in any of ESs.

4.2 Crossover and Mutation

Swapping the intersection genes in two coding strings, hoping to produce progeny of superior caliber, is the random crossover, which can be illustrated in Fig. 3. In the operation of Fig. 3, the number 3 is bartered with the number 5, which

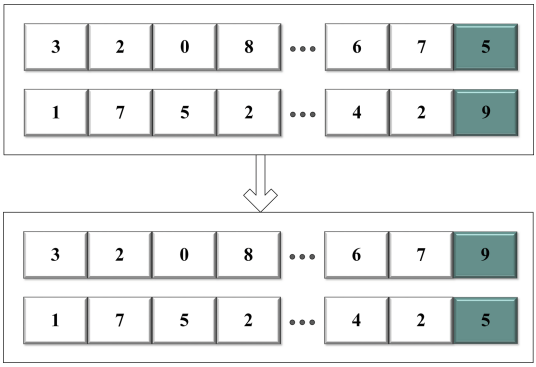


Fig. 3. Crossover operation.

means that the ES selected has shifted. The interchange of the numbers 5 and 9 indicates that the execution location is changed from ESs to CC.

The mutation is achieved by modifying chromosomes to ensure the diversity of the population and avoid early convergence of the solution set. An illustration of generating new individuals by mutation is shown in Fig. 4. After the mutation operation, the second task is executed locally, and the third task is executed by CC.

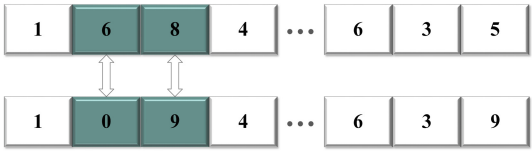


Fig. 4. Mutation operation.

4.3 Non-dominated Sorting

In multi-objective optimization, A dominates B signifying that the benefits of all objectives of solution A are superior to that of B, which can be defined as $A \prec B$. Non-dominance sorting is the process of sorting the solutions to distinct ranks $R_1, ...R_v$ by the dominance relationships.

4.4 Elimination Mechanism

$d(S, P)$ indicates the dominant points of the point S in the set P_t. $d(S, P)$ can highlight the sparse regions of the solution set and retain the diversity of the population, which can be measured as

$$d(S,P) = \mid \{y \in P \mid y \prec S\} \mid . \tag{15}$$

Hypervolume (HV) can convey the unique contribution of the current strategy. A benefit is HV also can get the Pareto front when the reference point is unknown. HV can be evaluated as

$$V(A, Y_{ref}) = \wedge \left(\bigcup_{y \in A} \{Y \mid y \prec Y \prec Y_{ref}\} \right), \tag{16}$$

where $\wedge$ symbolizes the Lebesgue measure, A signifies the Pareto optimal solution set, and Y_{ref} denotes a reference point.

$d(S,P)$ is employed as the selection criterion when the solution dominates. Otherwise, the size of HV is chosen as the requirement if all individuals of P_t are non-dominated [29,30].

4.5 Optimal Selection

As computation offloading is a discrete problem, adaptation values for time consumption $V(T_{sum})$, energy consumption $V(E_{sum})$, and resource utilization $V(A_R)$ is required. The normalized fitness value fit is used for the determination of the optimal solution, which can be expressed as

$$fit = \alpha \cdot V(T_{sum}) + \beta \cdot V(E_{sum}) + \delta \cdot V(A_R), \tag{17}$$

where α, β and δ is the weight of each objective

4.6 Method Overview

Algorithm 1 illustrates the overall process of MOWASC. Firstly, some basic parameters are initialized (Lines 1–2). Then, a new generation is generated by crossover and mutation (Line 4). Then, Eqs. (8), (9), and (13) are used to calculate the time and energy consumption and resource utilization (Lines 5–7). Then, the solution is classified as different ranks by non-dominated sorting (Line 8). Then, $d(S,P)$ is used to complete the initial screening in the elimination mechanism, then the further disuse through HV to perform, and P_{t+1} is updated finally by calculating the fitness value (Lines 9–17). The algorithm will end when the maximum number of iterations is met (Lines 3–19).

5 Comparison and Analysis of Experimental Results

In this section, we demonstrate the effectiveness of MOWASC through extensive experiments. Firstly, the experimental setup is introduced. Then, the comparative algorithms and experimental evaluation are given.

Algorithm 1. MOWASC

Input: The max iterations
Output: The offspring population $\boldsymbol{P_{t+1}}$

$P_0 \leftarrow$ Initial population
$t \leftarrow 0$
while max iterations fulfilled **do**
    Update P_t by crossover and mutation
    Calculate T_{sum} by Equation (8)
    Calculate E_{sum} by Equation (9)
    Calculate A_R by Equation (13)
    $\{ R_1, ...R_v \} \leftarrow$ non-dominated sort(P)
    if v>1 **then**
        Calculate $d(S, P)$ by Equation (15)
        r $\leftarrow R_v$ with highest $d(S, P)$
    else
        Calculate HV by Equation (16)
        r $\leftarrow R_1$ with lowest HV
    end if
    $P_{t+1} \leftarrow \{ P_t \setminus r \}$
    $t++$
end while

Optimal selection through fit **return** $\boldsymbol{P_{t+1}}$

5.1 Experimental Setting

All the simulation experiments are processed on a Win10 64-bit Operating System based on JAVA and the processor of physical machine is AMD Ryzen 7 PRO 4750U with Radeon Graphics 1.70 GHz.

In order to evaluate the applicability of these experiments in smart campus, two different groups of experiment are carried out. Firstly, to test the experimental utility when changing with the number of MUs, we study a test with 8 ESs serving, whose frequencies are distributed between 2000 and 2500.

Then, the second experiment is set up to test whether the effect would make adjustments for some challenging heterogeneous workflows. Two MUs by default in this experiment, each MU has an inconsistent workflow. The detailed parameter settings in experiments are listed in Table 1.

Next, the comparative methods, namely, Benchmark and FCFS are introduced in detail.

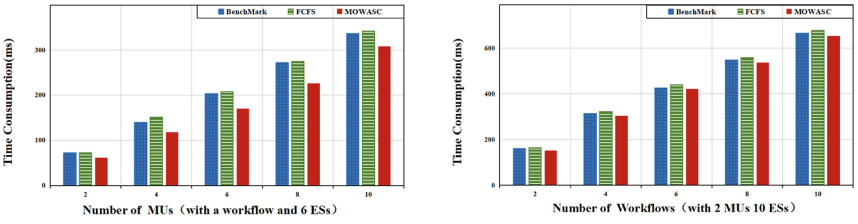
- **Benchmark** Tasks are distributed in a 2:3:15 ratio. Numbers 2, 3, and 15 indicate the probability that the task is executed locally, offloaded to CC, and migrated to ESs, respectively.
- **First Come First Service (FCFS)** Tasks are executed sequentially. Counting from 0, the 0-th arriving task is executed locally, and the tasks $\{ 1, \ldots, C \}$ are offloaded to the ESs for execution. Task $C + 1$ is offloaded to CC for execution.

Table 1. Parameters Setting

Parameter	Value
The computing frequency of SD	500 Mhz
The computing frequency of CC	3000 MHz
The computing frequency of ES	2000 MHz
The active power of SD	2 W
The idle power of SD	0.1 W
The transmission power of SD	0.8 W
The propagation time of LAN	0.5 ms
The propagation time of WAN	2 ms
The bandwidth of LAN	200 kps
The bandwidth of WAN	150 kps

5.2 Experimental Evaluation and Discussion

In this subsection, the optimized results of total time consumption, energy consumption, and resource utilization are comprehensively analyzed.

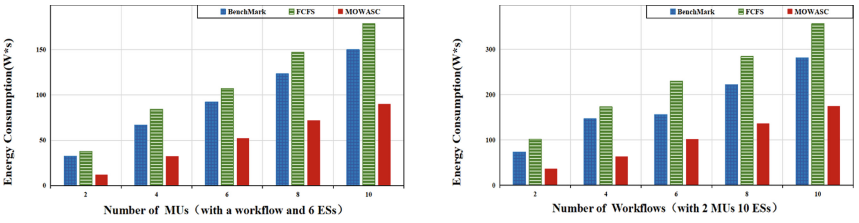


(a) (b)

Fig. 5. Comparison of time consumption.

Comparison of Time Consumption. The total time consumption is obtained by Eq. (8) and the comparative outcomes of FCFS, Benchmark, and MOWASC in two distinct circumstances are depicted in Fig. 5. It can be seen that the time overhead of the methods FCFS and Benchmark is similar, while MOWASC consumes the least time. Focusing on Fig. 5(a), MOWASC requires 16% and 14% less time than FCFS and Benchmark, respectively. In Fig. 5(b), as the quantity of workflows changes, the optimization effect and trend are similar to the situation in Fig. 5(a). In conclusion, with the increase of MUs or workflows, MOWASC is the best method in reducing time consumption.

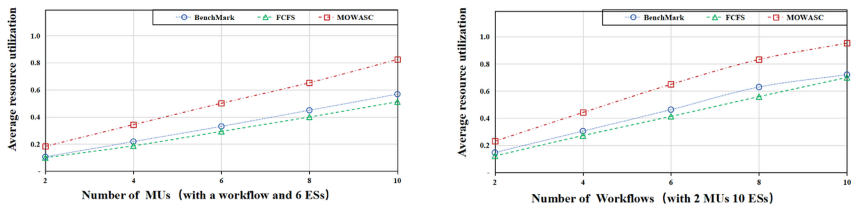
Comparison of Energy Consumption. The total energy consumption is calculated by Eq. (9) and Fig. 6(a) and Fig. 6(b) depict the experimental comparison of FCFS, Benchmark, and MOWASC. It can be seen that FCFS consumes the most energy, followed by Benchmark, while the energy depletion of MOWASC is always less than them under different situations. More specifically, compared with the other two methods, the overall energy optimization revenue of MOWASC reached 50%. In conclusion, MOWASC can obtain the best effect in reducing energy consumption.



(a) (b)

Fig. 6. Comparison of energy consumption.

Comparison of Resource Utilization. The resource utilization of ESs can be obtained by Eq. (13) and the related experiment effects of methods FCFS, Benchmark, and MOWASC can be found in Fig. 7(a) and Fig. 7(b). From Fig. 7(a) and Fig. 7(b), we can see that the resource utilization of method MOWASC is the highest under various conditions, with FCFS and Benchmark far behind. This means that, compared with other methods, method MOWASC adequately schedules the computing resources of ESs. When the number of workflows is 10, the magnitude of the rise in MOWASC slowed down. To sum up, MOWASC can obatin the best effect in optimizing resource utilization.



(a) (b)

Fig. 7. Comparison of resource utilization.

6 Conclusion

The integration of smart campus and MEC is an awesome way to improve the service quality of the campus. Latency-sensitive applications are representative applications, and the execution of such applications affects the overall performance of the smart campus. In this paper, this kind of application is modeled as workflows. Furthermore, energy consumption, time consumption, and resource utilization are seen as the optimization goals. To this end, we proposed an intelligent computation offloading method using SMS-EMOA. Extensive experiments have demonstrated that our proposed method MOWASC can achieve better optimization results than the other comparative methods. In another word, our proposed method can fully utilize the computing capacity of ESs provided by MEC to improve the performance of smart campus. In future work, we will concentrate on computation offloading for multi-workflow by combining of reinforcement learning.

References

1. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
2. Wu, H., Wolter, K., Jiao, P., Deng, Y., Zhao, Y., Xu, M.: EEDTO: an energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing. IEEE Internet Things J. **8**(4), 2163–2176 (2020)
3. Xu, X., et al.: Edge content caching with deep spatiotemporal residual network for IoV in smart city. ACM Trans. Sens. Netw. (TOSN) **17**(3), 1–33 (2021)
4. Bastari, A., Bandono, A., Suharyo, O.S.: The development strategy of smart campus for improving excellent navy human resources. Global J. Eng. Technol. Adv. **6**(2), 033–043 (2021)
5. Chen, T.C.: Smart campus and innovative education based on wireless sensor. Microprocess. Microsyst. **81**, 103678 (2021)
6. Peng, G., Wu, H., Wu, H., Wolter, K.: Constrained multiobjective optimization for IoT-enabled computation offloading in collaborative edge and cloud computing. IEEE Internet Things J. **8**(17), 13723–13736 (2021)
7. Li, W.: Design of smart campus management system based on internet of things technology. J. Intell. Fuzzy Syst. **40**, 3159–3168 (2021)
8. Liu, P., Peng, K., Tao, P.: Intelligent computation offloading for educational virtual reality applications in smart campus using MoCell. Comput. Intell. **39**, 82–103 (2022)
9. Dong, Z.Y., Zhang, Y., Yip, C., Swift, S., Beswick, K.: Smart campus: definition, framework, technologies, and services. IET Smart Cities **2**(1), 43–54 (2020)
10. Prandi, C., Monti, L., Ceccarini, C., Salomoni, P.: Smart campus: fostering the community awareness through an intelligent environment. Mob. Netw. Appl. **25**(3), 945–952 (2020)
11. Bello, S.A., et al.: Cloud computing in construction industry: use cases, benefits and challenges. Autom. Constr. **122**, 103441 (2021)
12. Schleier-Smith, J., et al.: What serverless computing is and should become: the next phase of cloud computing. Commun. ACM **64**(5), 76–84 (2021)

13. Wu, H., Sun, Y., Wolter, K.: Energy-efficient decision making for mobile cloud offloading. IEEE Trans. Cloud Comput. **8**, 570–584 (2020)
14. Abdulqadir, H.R., et al.: A study of moving from cloud computing to fog computing. Qubahan Acad. J. **1**(2), 60–70 (2021)
15. Kourgiozou, V., Commin, A.N., Dowson, M., Rovas, D.V., Mumovic, D.: Scalable pathways to net zero carbon in the UK higher education sector: a systematic review of smart energy systems in university campuses. Renew. Sustain. Energy Rev. **147**, 111234 (2021)
16. Peng, K., Zhao, B., Bilal, M., Xu, X.: Reliability-aware computation offloading for delay-sensitive applications in mec-enabled aerial computing. IEEE Trans. Green Commun. Netw. **6**(3), 1511–1519 (2022)
17. Qu, G., Wu, H.: DMRO: a deep meta reinforcement learning-based task offloading framework for edge-cloud computing. IEEE Trans. Netw. Serv. Manag. **18**, 3448–3459 (2021)
18. Xu, X., Tian, H., Zhang, X., Qi, L., He, Q., Dou, W.: DisCOV: distributed COVID-19 detection on X-ray images with edge-cloud collaboration. IEEE Trans. Serv. Comput. **15**(3), 1206–1219 (2022)
19. Chang, Z., Liu, L., Guo, X., Sheng, Q.: Dynamic resource allocation and computation offloading for IoT fog computing system. IEEE Trans. Ind. Inf. **17**, 3348–3357 (2021)
20. Li, N., Yang, S., Wang, Z., Hao, W., Zhu, Y.: Multi-tier MEC offloading strategy based on dynamic channel characteristics. IET Commun. **14**, 4029–4037 (2020)
21. Sun, J., Yin, L., Zou, M., Zhang, Y., Zhang, T., Zhou, J.: Task offloading, load balancing, and resource allocation in MEC networks. IET Commun. **14**, 1451–1458 (2020)
22. Zhang, G., Zhang, W., Cao, Y., Li, D., Wang, L.: Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. IEEE Trans. Ind. Inf. **14**(10), 4642–4655 (2018)
23. Huang, L., Feng, X., Zhang, L., Qian, L., Wu, Y.: Multi-server multi-user multi-task computation offloading for mobile edge computing networks. Sensors **19**(6), 1446 (2019)
24. Huang, M., Liu, W., Wang, T., Liu, A., Zhang, S.: A cloud-MEC collaborative task offloading scheme with service orchestration. IEEE Internet Things J. **7**, 5792–5805 (2020)
25. Sun, J., Yin, L., Zou, M., Zhang, Y., Zhang, T., Zhou, J.: Makespan-minimization workflow scheduling for complex networks with social groups in edge computing. J. Syst. Archit. **108**, 101799 (2020)
26. Xu, X., et al.: Multiobjective computation offloading for workflow management in cloudlet-based mobile cloud using NSGA-II. Comput. Intell. **35**(3), 476–495 (2019)
27. Ma, X., Gao, H., Xu, H., Bian, M.: An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing. EURASIP J. Wirel. Commun. Netw. **2019**(1), 1–19 (2019)
28. Borodin, A., Kleinberg, J., Raghavan, P., Sudan, M., Williamson, D.P.: Adversarial queuing theory. J. ACM (JACM) **48**(1), 13–38 (2001)
29. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. **181**(3), 1653–1669 (2007)
30. Koch, P., Kramer, O., Rudolph, G., Beume, N.: On the hybridization of SMS-EMOA and local search for continuous multiobjective optimization. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 603–610 (2009)