Multi-objective Computation Offloading in MEC-Empowered Smart Warehousing

Liangfei Yang[1,2], Kai Peng[1,2,3(✉)], 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
[3] Sincetech (Fujian) Technology Co., Ltd., Quanzhou, China
kai.peng@hqu.edu.cn

Abstract. As a typical application of the Industrial Internet of Things (IIoT), smart warehousing has attracted widespread attention. In addition, smart warehousing is regarded as a key part of logistics and supply chain management. The main idea of smart wareshousing is to deploy a large number of smart devices (SDs) to collect large amounts of data for improving the efficiency of digital management. However, it is difficult for SDs to process large amounts of data due to their limited computing capacity, meanwhile, traditional cloud-based smart warehousing paradigm often suffers from high latency disadvantage. Fortunately, mobile edge computing (MEC) can make up for the above shortcoming. Nevertheless, it is challenge to effectively integrate edge computing and smart warehousing. In view of this, we investigate the computation offloading problem in MEC-empowered smart warehousing, and propose an intelligent computation offloading algorithm to optimize time consumption and energy consumption of SDs as well as the resource utilization of the edge server cluster in this paper. Finally, we conduct several group of experiments to prove the effectiveness of our proposed method, and the results indicate that our method outperforms the other comparison methods in the given optimization objectives.

Keywords: MEC · Computation Offloading · Smart Warehousing

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, China Postdoctoral Science Foundation under Grant No. 2022M710700.

Y. Cao and X. Shao (Eds.): MONAMI 2022, LNICST 474, pp. 230–244, 2023.
https://doi.org/10.1007/978-3-031-32443-7_16

1 Introduction

The implementation of the Industrial Internet of Things (IIoT), which combines intelligence, data sharing, and more extensive computerization [1], has profoundly changed the way of living and working [2,3]. As a symbol of the IIoT, smart warehousing is proposed. Smart warehousing is an integrated hyper-connected platform which highly relies on the combination of intelligent devices and traditional industrial facilities.

Additionally, smart warehousing is a space to store goods for different fields, which aims to increase the overall intelligence, automation and integration of the warehouse. In addition, smart warehousing is a traceability and unmanned place when performing the operation of purchase, inbound, and outbound [4]. Smart devices (SDs) such as Automated Guided Vehicle (AGVs) and sensors are deployed in corresponding areas to control and manage the entire process of different segments in the warehouse. The maximum efficiency of the warehouse is guaranteed due to the participation of SDs [5]. Nevertheless, SDs have to process huge amount of digital data during the whole process [6,7]. However, due to the limited computing capacity of SDs, especially for the limited battery capacity, their efficiency will be greatly reduced when the amount of data becomes very large [8,9].

Fortunately, mobile edge computing (MEC) is regarded as one of the promising solutions to address the above shortcoming. MEC provides service by deploying ESs around SDs in smart warehousing. As an effective method to extend the computing capacity of SD, computation offloading has been extensively studied [10,11]. A large amount of data generated by SDs are offloaded to the ES, which can effectively provide fast and high-quality services for warehousing.

On the other hand, real-time information processing is very important for smart warehousing system. In another word, untimely cargo information adversely affects the entire warehouse management system [12]. And therefore, the optimization of time consumption in MEC-enabled smart warehousing is a critical issue. In addition, the computing resources of ES is usually characterized as limited, and how to use the limited resources to process more cargo information for high-efficiency warehouse is becoming critical. In view of this, we investigate the computation offloading in MEC-empowered smart warehousing. The main contributions of this paper can be concluded as the following points.

- We take into account the time consumption and energy consumption of SDs and the resource utilization of the ES cluster as the objectives in such a system. And we establish corresponding mode for the above objectives.
- A computation offloading algorithm called multi-objective optimization in smart warehousing (MOSW) is proposed based upon the traditional genetic algorithm, aiming at obtaining an appropriate offloading strategy.
- Finally, we conduct comprehensive experimental to verify the validity of our proposed method. The experimental results and analysis show that MOSW outperforms comparison methods under different situations.

The remainder of this paper is laid out as follows. Section 2 illustrates related work. Section 3 presents the model for multi-objective optimization in MEC-empowered smart warehousing scenario. Then, the proposed algorithm is illustrated in Sect. 4. Section 5 introduces experiments under different scenarios and the analysis of the results. Finally, we describe conclusion and our future work.

2 Related Work

In this part, we review the existing research from the perspectives of computation offloading in IIoT and computation offloading for MEC.

Computation offloading in IIoT. There has been extensive research on computation offloading in IIoT. Aiming at solving the problem of resource allocation and computation offloading, a task offloading method was proposed in [13]. This method introduced reinforcement learning to achieve an optimal binary computation offloading decision, which effectively reduced the computational cost and delay in IIoT scenarios. The multi-hop computation offloading problem was studied in [14], they proposed two distributed algorithms to provide stable performance gains for IIoT. Considering the high requirements of IIoT in terms of time and energy consumption, Chen et al. [15] focused on an energy-optimized non-static computation offloading program in fog computing scenarios with the goal of reducing energy cost. Similarly, Ren et al. [16] investigated a deep learning approach to minimize system energy consumption across multiple IIoT devices and multiple fog access points.

Computation Offloading for MEC. The concept of computation offloading is to migrate computing tasks generated by SDs to edge nodes. Computation offloading in MEC has been studied, either aims to optimize the delay [17,18], or energy consumption [19,20] or optimize the above two objectives jointly [21,22] . The time minimization problem was studied in a MEC system consisting of mobile users and heterogeneous ESs [17]. The optimal offloading node was selected by Markov decision process and the minimum offloading time was finally obtained. Wu et al. [18] studied an online method based on quality of service and real-time prediction of user trajectory aiming at minimizing response time of ES while considering user mobility. Wang et al. [19] researched the selective migration strategy problem by ARIMA-BP model with the aim of minimizing energy consumption of mobile devices, while meeting latency constraints. The ARIMA-BP model was used to estimate the computing power of the edge cloud. Zhang et al. [20] investigated the energy-saving computation offloading problem of MEC in 5G heterogeneous networks, aiming at minimizing the energy consumption of the MEC system. Tao et al. [21] studied a low-complexity sorting method to solve the multi-objective resource assignment issue. Finally, the joint optimization of task delay and device energy consumption is obtained. Wang et al. [22] studied the energy minimization optimization problem in MEC. They applied the Karush-Kuhn-Tucker to solve this problem and proposed a task offloading scheme to tradeoff time and energy consumption.

Different from previous studies, we study the computation offloading problem in MEC-empowered smart warehousing. And we consider both SDs and the server cluster with the aims of improving the overall service quality of smart warehousing. For one thing, time and energy consumption are taking into consideration to meet real-time applications requirements. For another, the resource utilization is considered to improve the system management efficiency. Meanwhile, an intelligent offloading method is leveraged.

3 System Model and Problem Formulation

In this part, the system model of MEC-empowered smart warehousing is illustrated, where the system model consists of time consumption, energy consumption model and resource utilization model, followed by the multi-objective optimization problem.

3.1 System Model

As illustrated in Fig. 1, our proposed system model of computation offloading in MEC-empowered smart warehousing is presented. In this system, AGVs and sensors are regarded as SDs to deliver and identify cargos. SDs in the warehouse are labeled as $ITD = \{i_1, i_2, i_3, \ldots, i_n\}$. The collection of ESs are deployed around SDs which is represented as $ES = \{es_1, es_2, es_3, \ldots, es_m\}$. The data center has more computing resources than ES but which is far from SDs.

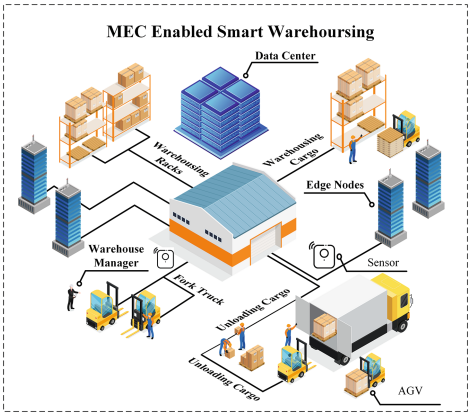

Fig. 1. An MEC-Enabled Smart Warehousing Architecture.

There are different offloading strategies for SDs. And choosing an appropriate offloading strategy is an important step to achieve multi-objective optimization. It is supposed that the offloading strategy of the $j-th$ task of the $i-th$ SD is

denoted as S_i^j. The tasks of SDs are assigned to the local, ES and data center according to the offloading strategy. These different offloading strategies are denoted as L, S, R, respectively.

$$\begin{cases} S_i^j = L & \text{The task is executed locally,} \\ S_i^j = S & \text{if } \mathrm{S_i^j} \text{ is offloaded to the ES,} \\ S_i^j = R & \text{if } \mathrm{S_i^j} \text{ is offloaded to the data center.} \end{cases} \tag{1}$$

3.2 Time Consumption Model

The total time consumption of SDs consists of three parts, including execution time TD_e, waiting time TD_w and transmission time TD_t.

Executing Time. Tasks are executed on the virtual machines (VMs) and the execution time is generated. Execution time is related to task workload and frequency of different SDs which is expressed as

$$TD_e(S_i^j) = \begin{cases} \frac{tw_{i,j}}{f_m}, S_i^j = L, \\ \frac{tw_{i,j}}{f_e}, S_i^j = S, \\ \frac{tw_{i,j}}{f_c}, S_i^j = R, \end{cases} \tag{2}$$

where $tw_{i,j}$ represents the task workload. The f_m represents the computing frequency of SDs, f_e represents the computing frequency of ES, and f_c represents the computing frequency of data center.

Waiting Time. The ES usually uses a limited number of VMs to provide resources, so the resources of ES are usually considered to be limited. When the number of tasks offloaded to ES exceeds the number of VMs, tasks $tw_{i,j}$ will join the waiting queue. Task join the execution sequence until ES has computed the previous task. The $a-th$ VM of the $b-th$ ES is represented by a two-tuple $v_{b,a} = (workload, tc)$. The workload represents a group of tasks workload of the $a-th$ VM and tc means the number of tasks in the VM. Corresponding to the migration strategy, task is considered to be offloaded to the VM with the smallest occupancy rate in the $p-th$ ES. As a result, the workload increases $tw_{i,j}$ on its basis. Similarly, the value of tc is also updated to tc+1. Loop the above steps until all tasks are allocated to the corresponding VM. And TD_w is expressed as

$$TD_w(S_i^j) = \sum_{i=1}^{N} \sum_{j=1}^{|tw_i|} TD_e(p(S_i^j)) \cdot \varepsilon_i^j, \tag{3}$$

where ε_i^j is to ensure whether $tw_{i,j}$ is assigned to the waiting queue, and the $TD_e(p(S_i^j))$ means the execution time of previous task.

Transmission Time. The transmission time between the SDs and ESs or data center is relevant to destination of task offload. If a task is assigned to execute locally, its transmission time is zero. And transmission time TD_t is expressed as

$$TD_t(S_i^j) = \begin{cases} 0, & S_i^j = L, \\ \frac{tw_{i,j}}{B_e}, & S_i^j = S, \\ \frac{tw_{i,j}}{B_c}, & S_i^j = R, \end{cases} \tag{4}$$

where B_e and B_c represent the bandwidth of the ESs and the data center, respectively.

Total Time Consumption. In summary, $TD^{(total)}(S)$ is used to represent the total time of the SDs, which can be expressed by

$$TD^{(total)}(S) = \sum_{i=1}^{N} \sum_{j=1}^{|tw_i|} (TD_e(S_i^j) + TD_w(S_i^j) + TD_t(S_i^j)). \tag{5}$$

3.3 Energy Consumption Model

Energy consumption is positively correlated with time consumption, corresponds to time consumption, energy consumption of SDs consist of execution energy consumption EC_e, waiting energy consumption EC_w and transmission energy consumption EC_t. The total energy consumption of SDs is shown below.

$$EC^{(total)}(S) = \sum_{i=1}^{N} \sum_{j=1}^{|tk_i|} (EC_e(S_i^j) + EC_w(S_i^j) + EC_t(S_i^j)). \tag{6}$$

3.4 Resource Utilization Model

In this paper, resource utilization is a key index for judging the performance of the ES cluster. When ESs process tasks from SDs, the resources of VMs in ESs are consumed. The value of resource utilization is obtained by the ratio of the used resources to the total resources. Utilized resources and total resources are represented by active VMs instances and VMs instances, respectively. The total number of VM instances of the $k-th$ ES is expressed as V_k. The resource utilization of ESs is as follows.

$$S_k = \frac{1}{V_k} \cdot \sum_{w=1}^{W} \sum_{m=1}^{M} p_{m,n} \cdot q_{m,n}^k, \tag{7}$$

where $p_{m,n}$ represents the number of VMs instances corresponding to s_i^j and $q_{m,n}^k$ is to judge whether the computing task is run on the kth ES. Therefore, the number of ESs performing tasks in this system be expressed as

$$PE = \sum_{s=1}^{m} O_k, \tag{8}$$

and the resource utilization of the ES cluster can be expressed as

$$AV(S) = \frac{1}{PE} \sum_{s=1}^{m} S_k. \tag{9}$$

3.5 Problem Formulation

We concentrate on minimizing the time consumption and energy consumption of SDs and the resource utilization of the ES cluster in this paper. subsequently, optimization goals are established. It is expressed as

$$\text{Min}\left\{\text{TD(S)}\right\}, \text{Min}\left\{\text{EC(S)}\right\}, \text{Max}\left\{\text{AV(S)}\right\} \tag{10}$$

$$TD^{(total)}(S) < deadline \tag{11}$$

Equation (11) indicates that the total time consumption of SD cannot exceed the set deadline.

4 Algorithm Design

In this part, we propose a multi-objective optimization genetic algorithm called multi-objective optimization in smart warehousing. MOSW has the advantages of traditional multi-objective genetic algorithms. On the one hand, MOSW divides individuals into several frontiers and MOSW does not rely on Pareto dominance but uses a scalarizing function. On the other hand, MOSW adopts the weight vector to maintain the diversity of the Pareto front (PF). Moreover, some traditional algorithms only consider ideal points as reference points in the fitness function, MOSW considers both the utopia point p^i and nadir point p^u at the same time, so that can be closer to PF.

The following introduces the main steps of MOSW. First of all, coding computation offloading strategies. Secondly, initialize the first generation population pop^f. Thirdly, the new generation population pop^n is obtained through crossover and mutation steps. Afterwards, populations are sorted by achievement scalarizing function (ASF) and use tournament selection to select next populations $pop^{(f+1)}$. Finally, SAW and MCDM are used for selecting optimal individual in the PF.

4.1 Encoding

In this part, the chromosomes of individuals in the population represent different computation offloading strategies for running applications in SDs. As shown in Fig. 2, the gene sequence in the chromosome is composed of $(0, 1, ..., S + 1)$. Different genes represent different ways of task offloading. And 0 means that the computing task is performed locally and S+1 represents that the task is migrated to the data center.

Fig. 2. Gene encoding.

4.2 Initialization

The first thing that a genetic algorithm does is perform initialization work. Initialization including set the size of the population, the probability of crossing genes p_c, the probability of gene mutation p_m, total number of iterations I_{tot} and the generations of the current iteration I_{now} in MOSW. Finally, it is assumed that initial population is established randomly.

4.3 Crossover and Mutation

During the crossover of chromosomes, two selected parent chromosomes exchange partial genes to form a better offspring chromosome. The purpose of the crossover operation is to pass on the chromosome segments of the excellent individuals to the offspring. As shown in Fig. 3, we assume that an individual has five genes and the crossover operation of the two genes takes place. It is obvious that crossover operation occurs between 1 and 5, 3 and 6 at the same time. The exchange of gene fragments is equivalent to the relevant offloading strategy is altered.

The mutation operator acts on the entire population, and the mutation probability p_m ensures the number of chromosomes to be mutated. An instance of a mutation operation be seen in Fig. 4. The gene in the chromosome 2 is turned into 5 means that the tasks originally offloaded to the 2-ed ES will be offloaded to the 5-th ES now. The mutation operation is to maintain population diversity by modifying some genes in chromosomes. New offspring are formed after completing these two operations.

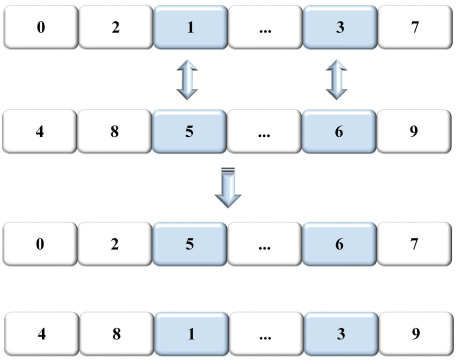

Fig. 3. Crossover operation.

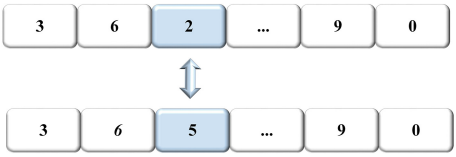

Fig. 4. Mutation operation.

4.4 ASF Sorting

In this subsection, we introduce the sorting process based on achievement scalarizing function. ASF is represented as the fitness function of this algorithm and is obtained by Chebyshev distance. The individuals in the population are divided into different frontiers by the value of ASF. This formula contains the objective function f and the reference point $q = \{q_1, q_2, ..., q_i\}$. The selection of reference points is generally considered to alternately select utopian points and nadir points. It is represented as

$$s = \max_{j \in 1,...,i} u_j(f - q_j) + p \sum_{j=1}^{i} (f - q_j) \tag{12}$$

where $u = \{u_1, u_2, ..., u_i\}$ represents vector of strictly positive weights. And p is regarded as augmentation coefficient. And augmentation term (the term $p \sum_{j=1}^{i} (f - q_j)$ in Eq. (12)) is set to ensure that minimizing the value of s can obtain a Pareto optimal solution. In Eq. (12), the reference points need to be updated when utopian points and nadir points are improved by an individual in offspring solutions.

Algorithm 1. ASF sorting

Input: initial population $\boldsymbol{pop^f}$ reference point $\boldsymbol{p^i}$ and $\boldsymbol{p^u}$
Output: next population $\boldsymbol{pop^{(f+1)}}$

while $x \in pop^f$ **do**
  The ASF value $x.n$ is obtained
  if $x.n < x.p^i$ **then**
$x.p^i \leftarrow x.n$
  end if
  if $x.n > x.p^u$ **then**
$x.p^u \leftarrow x.n$
  end if
  rank the population pop^f $sort \leftarrow 1$
  if Number of$(pop^{(f+1)}) < pop^f$ **then** $pop.sort \leftarrow sort$
  end if
$sort \leftarrow sort+1$
end while

4.5 Selection

Generally, each offspring have two parents. The parents are selected from the parent population through a tournament selection. In the parent population, all individuals are considered to have the same probability of being selected. Individuals with lower ASF values are regarded as the next parent population. Repeat the above operations until the next generation of population.

4.6 Optimal Selection by SAW and MCDM

When the number of iterations reaches the maximum iteration, the collection of Pareto frontiers is generated. It is necessary to choose the optimal individual among the population. And SAW [23] and MCDM [24] are regarded as an efficient method to select optimal solutions with higher speed and better convergence.

4.7 Method Overview

Algorithm 1 shows the detailed process of ASF sorting and Algorithm 2 demonstrates the steps of MOSW. The purpose of MOSW is to form an appropriate migration strategy to achieve the above-mentioned objective optimization. It is known that diverse chromosome represents different offloading strategies. Frist of all, the chromosomes of the population are initialized and called pop^f and total iterations called I_{tot}. Secondly, select two reference point, the nadir point and utopian points, respectively (Line 1). Then, perform ASF sorting to assign individuals to different fronts (Line 2). In Algorithm 1, we use ASF to divide the non-dominated solution set. The population is generated after the above steps are completed. Afterwards, select better individuals through the tournament selection, crossover and mutation operations to obtain the next generation (Lines 3-4). Calculate the three goals of the population separately by different formulas (Line 5-7). The population of 2N is selected as the population of N after the above operations (Line 8). Afterwards, evaluate population and update reference points including utopia points and nadir points, and perform the ASF sorting by Algorithm 1 (Lines 9-10). The population needs to be equal to the population size of P0. The population begins to iterate until it reaches the maximum number of iterations and obtain the collection of PF (Line 11).

5 Evaluation of Experimental Results

In this section, the superiority of MOSW is demonstrated by experiments under different conditions. Firstly, we introduce the experimental setting. Afterwards, experimental performance and analysis are shown.

Algorithm 2. MOSW

Input: The population $\boldsymbol{pop^f}$ and total number of iterations$\boldsymbol{I_{tot}}$
Output: The best solution $\boldsymbol{S}$

```
Select the reference point p^i and p^u
Perform ASF sorting algorithm by algorithm 1
while I_now < I_tot do
    Execute tournament selection,crossover and mutation procedures to pop^f and structure a population pop^n with N
    Time consumption is calculated by Equation (5)
    Energy consumption is calculated by Equation (6)
    Resource utilization is calculated by Equation (9)
    pop^t ← pop^n ⋃ pop^f
    Update reference points p^i, p^u
    Perform ASF sorting algorithm by algorithm 1
    Next population pop^(f+1)
end while
SAW and MCDM to select an optimal offloading strategy
return S
```

5.1 Experimental Setting

In our experiments, we set up several sets of experiments under different conditions to verify the superiority of MOSW. In multiple applications experiments, it assumes the number of applications is from 10 to 50 or 100 to 300, and the fixed number of SDs is 2. In multiple SDs experiments, the number of SDs is from 5 to 20 and each SD has 10 applications and the number of ESs is set to 5. More specific information about the experimental parameters is shown in Table 1.

Table 1. Parameters Setting

Parameter	Value
The active power of the smart device L_{v2d}	0.6 W
The idle power of the smart device L_{v2c}	0.01 W
The clock frequency of ESs	1800-2500 MHz
The clock frequency of data center	5000 MHz
The bandwidth between smart device and ES	300 kps
The bandwidth between ES and data center	200 kps
The maximum number of ESs	20
The population size	100
The maximum number of iterations	300

In order to highlight the superiority of MOSW in MEC-empowered smart warehousing. We introduce two comparison methods, namely, Benchmark and FCFS, which are shown as follows.

- **Benchmark** All applications are randomly migrated to ES cluster or data center. If VMs are not enough to handle applications, the new coming applications enter the waiting queue until the free VMs are released.
- **First Come First Service (FCFS)** All applications are offloaded in an orderly manner. More specifically, the first edge server handles the first application, the following applications are executed on the corresponding ES in order. Finally, the last one is executed by cloud.

The above experiments are all run based on JAVA language on a PC with 8 Intel Core i5-8250U 1.60 GHz processor and 8 GB RAM.

5.2 Experimental Evaluation and Discussion

We conduct rigorous and diverse comparative experiments to evaluate the performance of the three methods in terms of time and energy consumption of SDs and resource utilization of the ES cluster. Figure 5, 6 and 7 illustrate the experimental results.

5.3 Comparison of Time Consumption

The total time consumption is obtained by Eq. (5). Figure 5(a)–Fig. 5(c) shows the time consumption result of Benchmark, FCFS and MOSW under different conditions. Figure 5(a) reveals the comparison of time consumption with different numbers of SDs. In the case of different application numbers, the performance results of the three above-mentioned methods are indicated in Fig. 5(b) and Fig. 5(c). It can be observed that the time consumption increases with the number of SDs or applications from the experimental results. Therein, when the number of tasks becomes larger, the superiority of MOSW is more obvious under different conditions. Compared with the other algorithm, MOSW is superior in terms of time consumption because it has a more reasonable offloading strategy. Finally, we can conclude that MOSW is more suitable for time consumption optimization.

5.4 Comparison of Energy Consumption

The total energy consumption is obtained by Eq. (6). The positive correlation between energy consumption and time consumption has been shown in the model section. Their correlation also be reflected in the experimental results. Figure 6(a) reveals the comparison of energy consumption under different scales of SDs. And the performance results under different scales of applications of the three above-mentioned methods are shown in Fig. 6(b) and Fig. 6(c). It can be observed that the energy consumption increases with the number of SDs or applications.

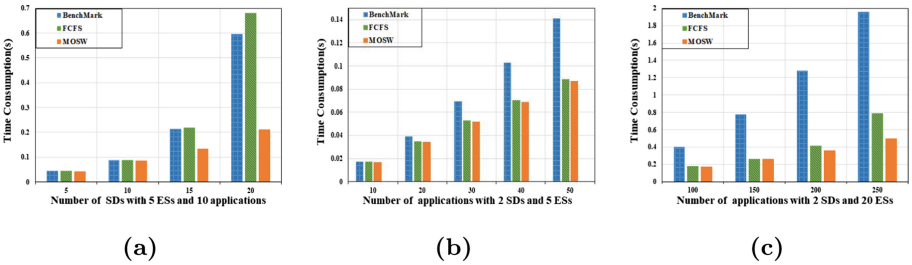

Fig. 5. Comparison of time consumption

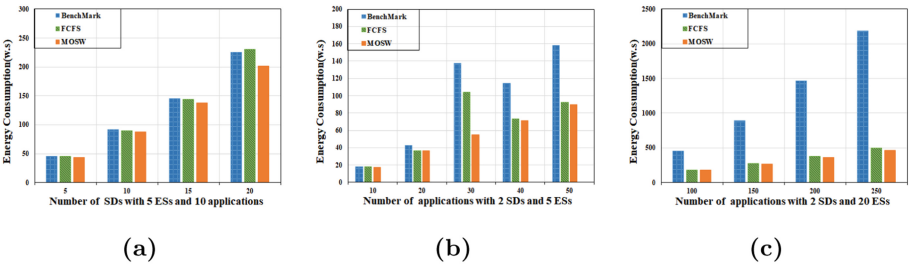

Fig. 6. Comparison of energy consumption

Therein, when the number of tasks becomes larger, the superiority of MOSW is more obvious. In summary, we can conclude that MOSW has advantages in energy optimization in various scenarios.

5.5 Comparison of Resource Utilization

The computing capacity of ESs is limited, and thus the resources should be fully utilized. The resource utilization is obtained by Eq. (9). Figure 7(a) reveals the comparison of resource utilization under different scales of SDs. In the case of different numbers of applications, the performance results of the three methods

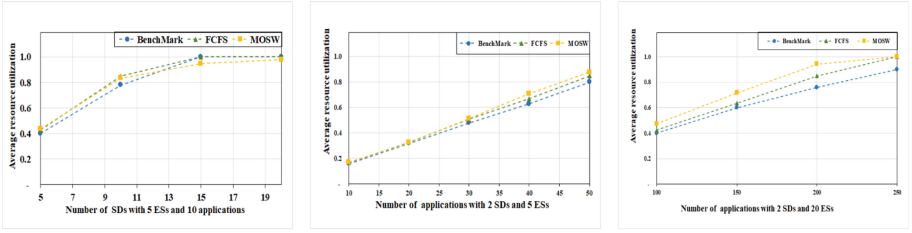

Fig. 7. Comparison of resource utilization

are indicated in Fig. 7(b) and Fig. 7(c). In the comparison results, MOSW has different degrees of optimization effect compared with the other two methods under different scenarios. Especially when the workload is heavy, the optimization effect is better. In conclusion, MOSW is more effective compared with other comparison methods.

6 Conclusion

The combination of MEC and warehouse provides a solution for realizing smart warehousing. In this paper, we have studied computation offloading for MEC-empowered smart warehousing, the purpose is to meet the demands of latency-sensitive applications with limited resources, we consider multiple optimization objectives, namely, time consumption and energy consumption of SD and resource utilization of ESs. Subsequently, a multi-objective optimization model is established and a computation offloading algorithm is proposed. Our proposed algorithm trades off three optimization objectives to get the optimization effect. Adequate results indicate that the proposed method has superior performance in terms of pre-optimization goals compared to the comparative methods. In future work, we will utilize deep learning to address the multi-objective optimization problem.

References

1. Wu, Y., Dai, H., Wang, H.: Convergence of blockchain and edge computing for secure and scalable IIoT critical infrastructures in industry 4.0. IEEE Internet Things J. **8**, 2300–2317 (2021)
2. 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**, 1206–1219 (2022)
3. Peng, K., Huang, H., Zhao, B., Jolfaei, A., Xu, X., Bilal, M.: Intelligent computation offloading and resource allocation in IIoT with end-edge-cloud computing using nsga-iii. IEEE Trans. Netw. Sci. Eng. 1–15 (2022). https://doi.org/10.1109/TNSE.2022.3155490
4. Liu, Y., Su, Z., Wang, Y.: Energy-efficient and physical layer secure computation offloading in blockchain-empowered internet of things. IEEE Internet Things J. (2022)
5. 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**, 13723–13736 (2021)
6. Qu, G., Wu, H., Li, R., Jiao, P.: Dmro: a deep meta reinforcement learning-based task offloading framework for edge-cloud computing. IEEE Trans. Netw. Serv. Manage. **18**(3), 3448–3459 (2021)
7. Kekana, P., Bakama, E.M., Mukwakungu, S.C., Sukdeo, N.: The impact of smart-warehousing on a local foodservice equipment-company's external customers. In: 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 771–775 (2020)

8. Wang, K., Ding, Z., So, D.K.C., Karagiannidis, G.K.: Stackelberg game of energy consumption and latency in MEC systems with NOMA. IEEE Trans. Commun. **69**, 2191–2206 (2021)
9. 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**, 2163–2176 (2021)
10. Xu, X., et al.: Edge content caching with deep spatiotemporal residual network for IoV in smart city. ACM Trans. Sens. Networks **17**, 29:1–29:33 (2021)
11. Peng, K., Huang, H., Bilal, M., Xu, X.: Distributed incentives for intelligent offloading and resource allocation in digital twin driven smart industry. IEEE Trans. Ind. Inf. (2022)
12. Wu, H., Sun, Y., Wolter, K.: Energy-efficient decision making for mobile cloud offloading. IEEE Trans. Cloud Comput. **8**(2), 570–584 (2018)
13. Hossain, M.S., Nwakanma, C.I., Lee, J.M., Kim, D.S.: Edge computational task offloading scheme using reinforcement learning for IIoT scenario. ICT Express **6**, 291–299 (2020)
14. Hong, Z., Chen, W., Huang, H., Guo, S., Zheng, Z.: Multi-hop cooperative computation offloading for industrial IoT-edge-cloud computing environments. IEEE Trans. Parallel Distrib. Syst. **30**, 2759–2774 (2019)
15. Chen, S., Zheng, Y., feng Lu, W., Varadarajan, V., Wang, K.: Energy-optimal dynamic computation offloading for industrial IoT in fog computing. IEEE Trans. Green Commun. Netw. **4**, 566–576 (2020)
16. Ren, Y., Sun, Y., Peng, M.: Deep reinforcement learning based computation offloading in fog enabled industrial internet of things. IEEE Trans. Industr. Inf. **17**, 4978–4987 (2021)
17. Yang, G., Hou, L., He, X., He, D., Chan, S., Guizani, M.: Offloading time optimization via markov decision process in mobile-edge computing. IEEE Internet Things J. **8**, 2483–2493 (2021)
18. Wu, C., Peng, Q., Xia, Y., Lee, J.: Mobility-aware tasks offloading in mobile edge computing environment. In: 2019 7th International Symposium on Computing and Networking (CANDAR), pp. 204–210 (2019)
19. Zhao, M., Zhou, K.: Selective offloading by exploiting ARIMA-BP for energy optimization in mobile edge computing networks. Algorithms **12**, 48 (2019)
20. Zhang, K., et al.: Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks. IEEE Access **4**, 5896–5907 (2016)
21. Tao, X., Ota, K., Dong, M., Qi, H., Li, K.: Performance guaranteed computation offloading for mobile-edge cloud computing. IEEE Wireless Commun. Lett. **6**, 774–777 (2017)
22. Xu, X., et al.: Game theory for distributed IoV task offloading with fuzzy neural network in edge computing. IEEE Trans. Fuzzy Syst. (2022)
23. Afshari, A., Mojahed, M., Yusuff, R.M.: Simple additive weighting approach to personnel selection problem (2010)
24. Aruldoss, M., Lakshmi, T.M., Venkatesan, V.P.: A survey on multi criteria decision making methods and its applications (2013)