An Effective Offloading Trade-Off to Economize Energy and Performance in Edge Computing

Yuting Cao, Haopeng Chen(✉), and Zihao Zhao

Shanghai Jiao Tong University, Dongchuan Rd., Shanghai, China
{seniyuting,chen-hp,1059198449}@sjtu.edu.cn

Abstract. Mobile Edge Computing is a new technology which aims to reduce latency, to ensure highly efficient network operation and to offer an improved user experience. Considering offloading will introduce additional wireless transmission overhead, the key technical challenge of mobile edge computing is tradeoff between computation cost and wireless transmission cost, reducing energy consumption of mobile edge devices and response time of computation task at the same time. A Mobile Edge Computing System composed of Mobile Edge Device and Edge Cloud, connecting with Wireless Stations, comes out. To protect user privacy, Data Preprocessing is proposed which includes irrelevant property clean and data segmentation. Aimed at reducing total energy consumption and response time, an energy consumption priority offloading (ECPO) algorithm and a response time priority offloading (RTPO) algorithm are put forward, based on Energy Consumption Model and Response Time Model. Combining both ECPO and RTPO, a dynamic computing offloading algorithm is raised which is more universal. Finally, simulations in four scenarios, including network normal scenario, network congested scenario, device low battery scenario and task time limited scenario, demonstrate that our algorithms can effectively reduce energy consumption of mobile edge device and response time of computation task.

Keywords: Mobile edge computing · ECPO · RTPO · Offloading

1 Introduction

With the rapid development of Internet of Things [1], mobile edge devices generate a lot of data, performing as not only data consumers but also data producers [2]. Under this circumstance, traditional cloud computing is no longer a wise choice since it has some obvious drawbacks. Firstly, it is not acceptable for real time computation task on mobile edge devices because bandwidth of network would be the bottleneck. Secondly, user privacy is another problem to be solved for cloud computing.

M. Luo and L.-J. Zhang (Eds.): CLOUD 2018, LNCS 10967, pp. 172–186, 2018.
https://doi.org/10.1007/978-3-319-94295-7_12

Mobile Edge Computing is a new technology which provides an IT service environment and cloud computing capabilities at the edge of mobile network, within the Radio Access Network and in close proximity to mobile subscribers. The aim is to reduce latency, to ensure highly efficient network operation and service delivery, and to offer an improved user experience [3]. In mobile edge computing scenario, network latency can be reduced by enabling computation and storage capacity at the edge network. And mobile edge devices can perform computation offloading for computing intensive applications to leverage the context-aware mobile edge computing service by using real time radio access network information [4]. Since offloading introduces additional communication overhead, a key technical challenge is how to balance between computation cost and communication cost to support applications with enhanced user experience, such as lower response time and energy consumption [5].

In this paper, we design a Mobile Edge Computing System composed of Mobile Edge Device and Edge Cloud, connecting with Wireless Station. To protect user privacy, Data Preprocessing is proposed including irrelevant property clean and data segmentation. Assuming that horizontal segmentation of input data will not affect computing result and execution time of computing is proportional to data size, input data can be divided into multiple data blocks and data block can be divided into multiple data slices with fixed data size. Only one data slice can be executed each time. In our research, we focus on reducing energy consumption of mobile edge device and response time of computation task, proposing an Energy Consumption Model and a Response Time Model.

In terms of mobile edge computing offloading problem, considering the change of network environment and the power of mobile edge device to make a dynamic decision will work better than making an unchanging decision at the beginning. Therefore, an energy consumption priority offloading (ECPO) algorithm and a response time priority offloading (RTPO) algorithm are raised in order to reduce energy consumption and response time. Furthermore, combining both ECPO algorithm and RTPO algorithm, we propose a new dynamic computing offloading algorithm as the greatest contribution of our paper. Finally, simulations in four scenarios: network normal scenario, network congested scenario, device low battery scenario and task time limited scenario, demonstrate that our proposed algorithms can effectively reduce energy consumption and response time.

The structure of this paper is as follows. Section 2 presents the related work. Section 3 describes system design and system model. In Sect. 4, we propose ECPO alogorithm, RTPO algorithm and a dynamic computing offloading algorithm combining previous two. Section 5 shows the simulation results and analysis. Section 6 is about conclusion and future work.

2 Related Work

Recently, computing offloading from mobile devices into cloud [6], as a key technical challenge of Mobile Edge Computing [7], has been extensively studied in many area including power management [8], cloud computing task migration [9,10] and virtual machine migration [11].

In order for better understanding of offloading, Orsini et al. provide a design guideline for the selection of suitable concepts for different classes of common cloud-augmented mobile applications and present open issues that developers and researchers should be aware of when designing their mobile cloud computing approach [12].

There are some existing offloading algorithms, like energy-optimal partial computation offloading (EPCO) algorithm [5], Lyapunov optimization-based dynamic computation offloading (LODCO) algorithm [13], distributed computation offloading algorithm [14] and the actor-model programming paradigm [15]. Sardellitti et al. consider an MIMO multicell system where multiple mobile users ask for computation offloading to a common cloud server and propose an iterative algorithm, based on a novel successive convex approximation technique, converging to a local optimal solution of original nonconvex problem [16].

In order to save energy, Zhao et al. design a threshold-based policy to improve the QoS of Mobile Cloud Computing by cooperation of local cloud and Internet cloud resources, which takes advantages of low latency of local cloud and abundant computational resources of Internet cloud simultaneously [17]. Ge et al. propose a game-theoretic approach to optimize the overall energy in a mobile cloud computing system and formulate the energy minimization problem as a congestion game, where each mobile device is a player to select one server to offload computation to minimize the overall energy consumption [18]. Wang and Giannakis investigate resource allocation policies for time-division multiple access over fading channels in the power-limited regime [19].

What's more, Chen et al. study the multi-user computation offloading problem for mobile-edge cloud computing in a multi-channel wireless interference environment and design a distributed computation offloading algorithm that can achieve a Nash equilibrium, deriving the upper bound of the convergence time and quantifying its efficiency ratio over the centralized optimal solution in terms of two important performance metrics [14]. You et al. study resource allocation for a multiuser MECO system based on time-division multiple access (TDMA) and orthogonal frequency-division multiple access (OFDMA) to solve the problem and characterize its policy structure, proposing a low-complexity sub-optimal algorithm by transforming the OFDMA problem to its TDMA counterpart [20].

3 System Design and System Model

In this section, we introduce system architecture design and data preprocessing method. The energy consumption model and response time model are also presented. Partial parameters used in the following are listed in Table 1.

3.1 System Architecture Design

The system is composed of two requisite parts: Mobile Edge Device and Edge Cloud, while data transmission of them is Wireless Station. The system architecture is shown in Fig. 1.

Table 1. Partial parameters

	Description
C_e	Number of CPU cycle required for computing 1-bit data at edge cloud
C_m	Number of CPU cycle required for computing 1-bit data at mobile edge device
D_l	Size of data slice in local computing
D_t	Size of data slice for offloading
D	Size of the whole data slice
E_c	Energy consumption per CPU cycle for a mobile edge device
E_l	Energy consumption for local computing of one data slice
E_t	Energy consumption for wireless transmission of one data slice
E	Energy consumption to compute the whole data slice
E_{total}	Total energy consumption of one computing task
f_e	CPU frequency for edge cloud
f_m	CPU frequency for mobile edge device
g	Channel gain
N	Variance of complex white Gaussian channel noise
P_t	Transmission power
R_t	Channel transmission rate
T_e	Response time for computing one data slice in edge cloud
T_l	Response time for local computing of one data slice
T_t	Transmission time of one data slice
T	Response time to compute the whole data slice
T_{total}	Total response time of one computation task
W	Channel bandwidth

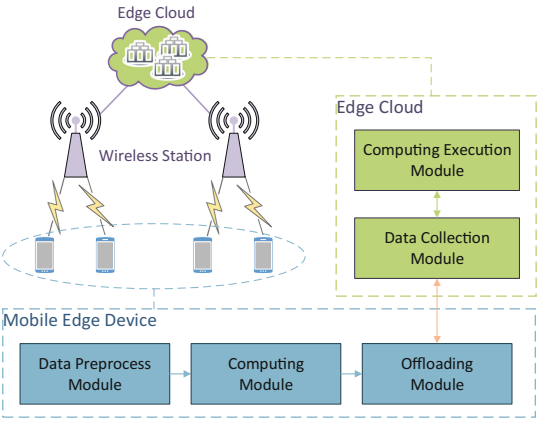


Fig. 1. System architecture

Mobile Edge Device. Mobile Edge Device is both data consumer and data producer in edge computing, including three modules: Data Preprocess Module, Computing Module and Offloading Module. Data Preprocess Module is responsible for irrelevant property clean and data segmentation. Computing Module is used for local computing. Offloading Module is used to develop calculation migration strategy.

Edge Cloud. There are two modules in Edge Cloud: Computing Execution Module and Data Collection Module. Computing Execution Module is a module that performs computational tasks, while Data Collection Module is applied for collecting data from Mobile Edge Device.

3.2 Data Preprocessing

There are two kinds of data preprocessing: Irrelevant Property Clean and Data Segmentation. On the one hand, the original data may contain some properties which are related to user private and computing irrelevant. On the other hand, in the case of horizontal segmentation of input data not affecting computing result, we propose to segment from multiple granularities on the basic definition of Data Unit, smallest data processing unit for performing computational tasks. Data Unit can be divided into multiple Data Blocks while Data Block can be divided into multiple Data Slices as shown in Fig. 2. The purpose of multi-granularity data segmentation is to dynamically adjust the computing migration mechanism for different granularity data which will be analyzed in Sect. 4.

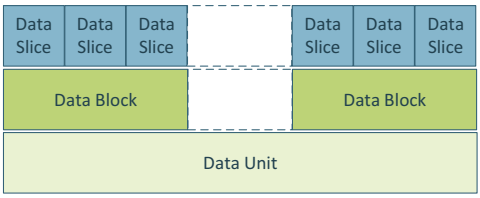


Fig. 2. Data segmentation

3.3 Energy Consumption Model

One of the main purposes of mobile edge computing is to reduce energy consumption of mobile edge devices as much as possible, including energy consumption for local computing and for wireless transmission.

Energy Consumption for Local Computing. For a mobile edge device, let E_c denote the energy consumption per CPU cycle for local computing, C_m denote the number of CPU cycles required for computing 1-bit of input data and D_l denote the size of data slice. Then the total energy consumption for local computing of a data slice is E_l, given by Formula 1.

$$E_l = C_m \cdot D_l \cdot E_c \tag{1}$$

Energy Consumption for Wireless Transmission. Let W denote the channel bandwidth, g denote the channel gain, P_t denote the transmission power and N denote the variance of complex white Gaussian channel noise. Then the channel transmission rate denoted by R_t, is Formula 2, according to Shannon formula.

$$R_t = W \cdot \log_2(1 + \frac{g \cdot P_t}{W \cdot N}) \tag{2}$$

Provided that W, P_t and N are constant, R_t will have a positive correlation with changeable g. However, if time slot is short enough, P_t and R_t can be considered constant, resulting to the total energy consumption for wireless transmission of a data slice as E_t, given by Formula 3, where T_t and D_t are transmission time and size of offloading data slice respectively.

$$E_t = P_t \cdot T_t = \frac{P_t \cdot D_t}{R_t} \tag{3}$$

By the way, energy consumption for wireless transmission of calculation result is negligible in this case due to size of result is much smaller than that of original data normally.

Total Energy Consumption. Considering the size of data slice is small enough, the local computing time and transmission time of data slice is short. Under this circumstances, analysis of energy consumption is acceptable.

Let D denote size of the whole data slice, D_l denote the size of data slice in local computing with $0 \leq D_l \leq D$, then the size of offloading data slice is $D_t = D - D_l$. Total energy consumption denoted as E is given by Formula 4.

$$E = E_l + E_t = \frac{P_t}{R_t} \cdot D + (C_m \cdot E_c - \frac{P_t}{R_t}) \cdot D_l \tag{4}$$

For the case of $C_m \cdot E_c - \frac{P_t}{R_t} > 0$, E will be minimum when $D_l = 0$. Oppositely, supposing $C_m \cdot E_c - \frac{P_t}{R_t} \leq 0$, E will be minimum when $D_l = D$. It shows that for a data slice small enough, to make energy consumption minimum, the whole data slice is either executed in local computing or offloaded to edge cloud.

The total energy consumption of one computation task can be expressed as Formula 5.

$$E_{total} = \sum E_{slice}, \ \textit{for each data slice} \tag{5}$$

3.4 Response Time Model

Another purpose of mobile edge computing is to reduce computing response time, which is mainly affected by three factors: local computing, wireless transmission and cloud computing.

Response Time for Local Computing. For a mobile edge device, let f_m denote CPU frequency for local computing which means the number of CPU cycles per second, C_m denote the number of CPU cycles required for computing 1-bit data and D_l denote the size of data slice in local computing. Then the total response time for local computing of a data slice is T_l, given by the following Formula 6.

$$T_l = \frac{C_m \cdot D_l}{f_m} \tag{6}$$

Response Time for Wireless Transmission. If the time slot is short enough, the channel transmission rate R_t can be considered to be constant. In this case, the total response time for wireless transmission of a data slice of input data is T_t, given by Formula 7.

$$T_t = \frac{D_t}{R_t} = \frac{D - D_l}{R_t} \tag{7}$$

where D_t is the size of offloading data slice. Response time for wireless transmission of computing results is ignored due to the relative smaller sizes.

Response Time for Cloud Computing. Obviously in edge cloud, the CPU frequency denoted as f_e and the number of CPU cycles required for computing 1-bit data denoted as C_e can be considered constant within a time slot which is short enough. Then the total response time for cloud computing of a data slice is T_e, given by Formula 8.

$$T_e = \frac{C_e \cdot D_t}{f_e} = \frac{C_e \cdot (D - D_l)}{f_e} \tag{8}$$

Total Response Time. For a data slice, local execution and offloading to edge cloud are simultaneous, which leads to the total response time be the maximum of them, given by Formula 9.

$$T = max(T_l, T_t + T_e) \tag{9}$$

Total response time of a computation task can be expressed as Formula 10.

$$T_{total} = \sum T_{slice}, \ for\ each\ data\ slice \tag{10}$$

4 Computing Offloading Mechanism

Reducing energy consumption and response time is two main purpose for computing offloading mechanism in edge computing. In this section, we proposed an energy consumption priority offloading algorithm named ECPO and a response time priority offloading algorithm named RTPO. Ultimately, we put forword a dynamic computing offloading algorithm based on ECPO and RTPO.

4.1 Energy Consumption Priority Offloading Algorithm

To reduce the total energy consumption of mobile edge device, an energy consumption priority offloading (ECPO) algorithm is proposed in Algorithm 1.

Algorithm 1: ECPO Algorithm

```
Input: data: input data or data block
Divide data into data slices
for each DataSlice do
    if DataSlice is non-offloadable then
        Execute DataSlice in local computing
    else
        Get connection information of wireless station
        Calculate C_m, E_c, P_t and R_t in Formula 4
        Set Param = C_m · E_c − P_t/R_t
        if Param > 0 then
            Offloading DataSlice to Edge Cloud
        else
            Execute DataSlice in local computing
```

As is shown in Fig. 3, in ECPO algorithm, data will be divided into multiple data slices with fixed data size under the assumption that horizontal segmentation will not affect computing result. For each data slice, it will be either executed in local computing or offloaded to edge cloud. Only one data slice can be executed each time. What's more, total energy consumption of one data slice E in ECPO algorithm is shown in Formula 4. As is proved by Energy Consumption Model, E will be minimum when using ECPO algorithm.

4.2 Response Time Priority Offloading Algorithm

Considering that ECPO algorithm has space for improving response time of computation task, an response time priority offloading (RTPO) algorithm is proposed in Algorithm 2.

As is shown in Fig. 4, in RTPO algorithm, the offloading weight of a data slice is calculated according to the power percentage of mobile edge device and

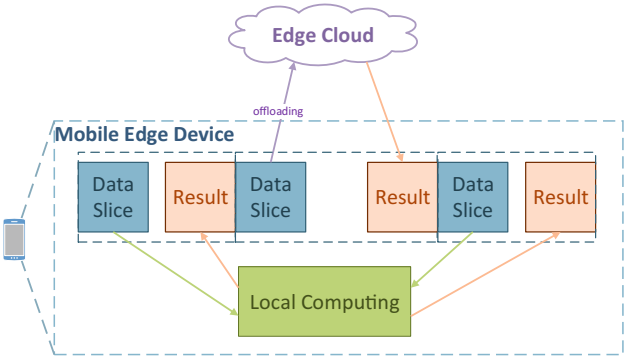


Fig. 3. Energy consumption priority offloading

Algorithm 2: RTPO Algorithm

Input: *data*: input data or data block
Divide *data* into data slices
for *each DataSlice* **do**
  if *DataSlice is non-offloadable* **then**
    Execute *DataSlice* in local computing
  else
    Get connection information of wireless station
    Calculate current channel transmission rate R_t
    $D_l = (f_m \cdot f_e + C_e \cdot f_m \cdot R_t) \cdot D/(C_m \cdot f_e \cdot R_t + f_m \cdot f_e + C_e \cdot f_m \cdot R_t)$
    Execute local computing and offload to Edge Cloud at the same time

network status. In other words, RTPO algorithm allows one data slice to execute both in local and edge cloud to earn less response time.

For each data slice, the response time for executing local computing to process a data unit is $\frac{C_m}{f_m}$, while time for offloading to edge cloud is $\frac{1}{R_t} + \frac{C_e}{f_e}$. So for a data slice, in order to equalize this two latencies, we get the weight for local computing D_l in Formula 11. As is analyzed previously in Formula 4, total energy consumption of one data slice E in RTPO algorithm will not be the worst case as D_l always satisfy the condition $0 < D_l < D$.

$$D_l = \frac{(f_m \cdot f_e + C_e \cdot f_m \cdot R_t) \cdot D}{C_m \cdot f_e \cdot R_t + f_m \cdot f_e + C_e \cdot f_m \cdot R_t} \tag{11}$$

4.3 Dynamic Computing Offloading Algorithm

In order to deal with more complex scenarios, we propose a dynamic computing offloading algorithm in Algorithm 3, combining ECPO and RTPO algorithm.

The dynamic computing offloading algorithm takes both power of mobile edge device and user requirements into consideration. On the one hand, if mobile edge

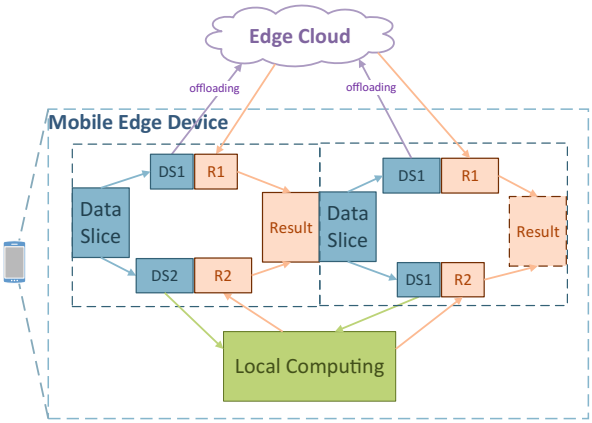


Fig. 4. Response time priority offloading

Algorithm 3: Dynamic Computing Offloading Algorithm

Input: w_e: weight of energy consumption, w_t: weight of response time

Get current power of Mobile Edge Device: *power*
Get maximum power of Mobile Edge Device: *power_max*
Get expected minimum energy consumption: E_{min}
if $power < E_{min}$ **then**
    return
else
    Divide input data into data blocks
    for *each DataBlock* **do**
        Divide *DataBlock* into data slices
        Get current power of Mobile Edge Device: *power*
        if $power > power\_max \cdot threshold\_power$ **then**
            Calculate $E_{ECPO}, E_{RTPO}, T_{ECPO}, T_{RTPO}$
            Set $Overhead_{ECPO} = E_{ECPO} \cdot w_e + T_{ECPO} \cdot w_t$
            Set $Overhead_{RTPO} = E_{RTPO} \cdot w_e + T_{RTPO} \cdot w_t$
            if $Overhead_{RTPO} < Overhead_{ECPO}$ **then**
                Use RTPO algorithm
            else
                Use ECPO algorithm
        else
            Use ECPO algorithm

device does not have enough power to support, task will be abandoned. Besides, let *threshold_power* denote the threshold of percentage of battery charge and compare power before processing each data block. On the other hand, let w_e and w_t denote the weight of reducing energy consumption and shortening response time respectively according to user requirements to calculate overhead as

Formula 12. Finally, we select algorithm with lower overhead to deal with current data block.

$$Overhead = E \cdot w_e + T \cdot w_t \tag{12}$$

The energy consumption model and time model of one computaion task are the sum of cost in each data block, which can be expressed as Formula 13.

$$\begin{aligned} E_{total} &= \sum E_{block},\ for\ each\ data\ block \\ T_{total} &= \sum T_{block},\ for\ each\ data\ block \end{aligned} \tag{13}$$

5 Evaluation

As 5G technology has not fully applied currently, it is scarcely possible to carry out experiments in real environment. In this section, we analyze our simulation result in four scenarios to evaluate performance of the proposed dynamic computing offloading mechanism, including network normal scenario, network congested scenario, device low battery scenario and task time limited scenario.

5.1 Experimental Parameter

Settings of our simulation are as follows. The size of data block and data slice are 100 MB and 1 MB separately. Energy consumption per CPU cycle for mobile edge device $E_c = 1.0 \times 10^{-10}$ J/cycle. The number of CPU cycles required for computing 1-bit data at mobile edge device and edge cloud is $C_m = C_e = 1000$ cycles/bits. CPU frequency for a mobile edge device is $f_m = 2.0 \times 10^9$ cycles/s while it for edge cloud is $f_e = 1.0 \times 10^{10}$ cycles/s. Transmission power is $P_t \in [0, 0.2]$ J/s. Channel transmission rate is $R_t \in [0, 2000]$ KB/s. Energy of mobile edge device at full charge is 20000 J and *threshold_power* is set to 30%.

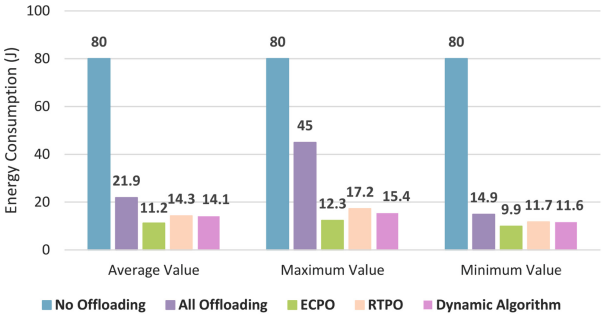
Moreover, as Formula 14 shows, $WirelessStation$ consists of n channels and m devices are maintained for each channel, of which m has negative correlation of the wireless transmission rate.

$$\begin{aligned} &WirelessStation(Channel_1, Channel_2, ..., Channel_{n-1}, Channel_n) \\ &Channel(Device_1, Device_2, ..., Device_{m-1}, Device_m) \end{aligned} \tag{14}$$

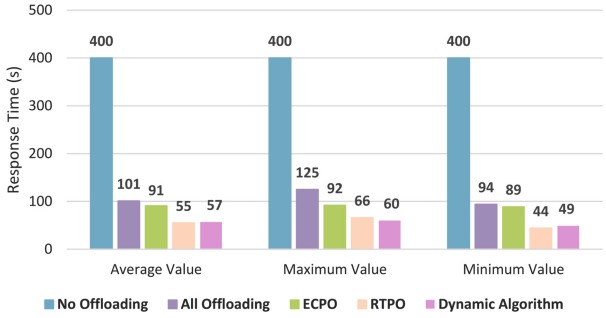
Assuming that $Number_{Channel}$ and $Number_{Device}$ represent the number of channels and mobile edge devices severally, we define the ratio of $Number_{Channel}$ and $Number_{Device}$ equalizing 20 as congested network, while equalizing 2 as normal network in our experiment.

5.2 Simulation Results Analysis

Network Normal Scenario. Processing computation task with size of 100 MB in normal network scenario with $Number_{Device}/Number_{Channel} = 2$, total energy consumption and response time of edge device are shown in Fig. 5. Apparently, ECPO, RTPO and Dynamic Algorithm have a great advantage over executing locally and all offloading to edge cloud. However, in terms of energy consumption, compared with ECPO and RTPO, advantage of Dynamic Algorithm is not obvious, as even slightly worse than ECPO. While in terms of response time, combining the superiority of RTPO, it performs better for Dynamic Algorithm than ECPO while a little worse than RTPO. Therefore, Dynamic Algorithm plays a role as a compromise between ECPO and RTPO.



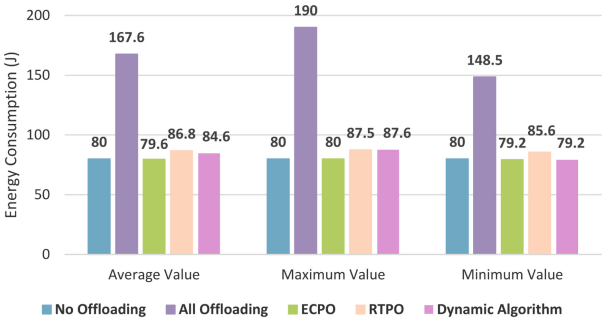
(a) Energy Consumption Analysis



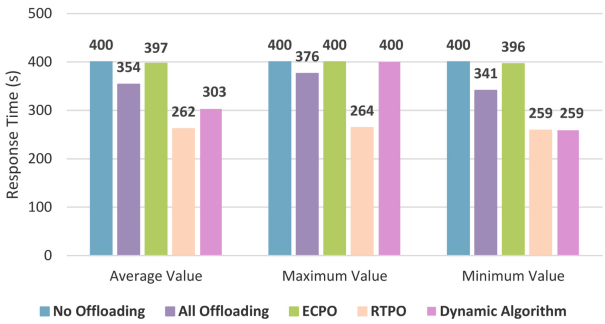
(b) Response Time Analysis

Fig. 5. Analysis in normal network scenario

Network Congested Scenario. Dealing with task with size of 100 MB in congested network scenario with $Number_{Device}/Number_{Channel} = 20$, results are shown in Fig. 6. In the respect of energy consumption, cost of Dynamic Algorithm is slightly higher than local execution due to high transmission cost in congested network, while is half as low as it of all offloading. From another perspective, response time of Dynamic Algorithm has distinct improvement over no offloading and all offloading, but still in the midst of ECPO and RTPO.



(a) Energy Consumption Analysis



(b) Response Time Analysis

Fig. 6. Analysis in Congested Network Scenario

Device Low Battery Scenario. The initial power of mobile edge device is set lower than 30% randomly in low battery scenario and the original data size of each task is 5 GB. Number of initial tasks is 100. Results of computing offloading is shown in Table 2. The failure of computing tasks is due to exhaustion of mobile edge devices. Success rates of ECPO and RTPO are 91% and 75% each. Dynamic Algorithm will estimate energy consumption before executing tasks and abandon calculation task when do not have enough power, resulting to 100% success rate.

Table 2. Result of computing offloading in device low battery scenario

	Initialization	Abandon	Failure	Success	SuccessRate
ECPO	100	0	9	91	91%
RTPO	100	0	25	75	75%
Dynamic	100	10	0	90	100%

Task Time Limited Scenario The initial power of mobile edge device is set randomly and the original data size of each task is 5 GB. Number of initial tasks is 100. In addition, each computing task is limited to 1 h. Results of computing offloading is shown in Table 3. For ECPO, there are 2 tasks failed due to depletion of mobile edge devices, while the remaining 98% violate the time limit due to timeout. For RTPO, although its success rate is 95%, 5 tasks failed due to exhaustion of mobile edge device. For Dynamic Algorithm, success rate is 83% without failure which is optimal entirely.

Table 3. Result of computing offloading in task time limited scenario

	Initialization	Abandon	Failure	Timeout	Punctuality	SuccessRate
ECPO	100	0	2	98	0	0%
RTPO	100	0	5	0	95	95%
Dynamic	100	4	0	16	80	83%

5.3 Brief Summary

The simulation results conducted in 4 scenarios show that our computing offloading machanism can comprehensively consider energy consumption, response time of computing tasks and load of edge cloud, by dynamically adjusting and calculating offloading strategy. The machanism is far better than local execution and all offloading to edge cloud.

6 Conclusion

We study computing offloading mechanism for a mobile edge computing system composed of mobile edge device and edge cloud, connecting with wireless station, to reduce energy consumption of mobile edge devices and response time of computation tasks. We propose ECPO and RTPO algorithm according to Energy Consumption Model and Response Time Model. Ultimately, we put forward a dynamic offloading algorithm combining the previous two, which is proved to be an effective way to achieve the original goals entirely. In the future, after full application of 5G technology, we will make further effort to build a real mobile edge computing scenarios to verify the effectiveness of our mechanism.

References

1. Xia, F., Yang, L.T., Wang, L., Vinel, A.: Internet of Things. Int. J. Commun. Syst. **25**(9), 1101 (2012)
2. Shi, W., Dustdar, S.: The promise of edge computing. Computer **49**(5), 78–81 (2016)

3. Hu, Y.C., Patel, M., Sabella, D., Sprecher, N., Young, V.: Mobile edge computing a key technology towards 5G. ETSI White Pap. **11**(11), 1–16 (2015)
4. Ahmed, A., Ahmed, E.: A survey on mobile edge computing. In: 2016 10th International Conference on Intelligent Systems and Control (ISCO), pp. 1–8 (2016)
5. Wang, Y., Sheng, M., Wang, X., Wang, L., Li, J.: Mobile-edge computing: partial computation offloading using dynamic voltage scaling. IEEE Trans. Commun. **64**(10), 4268–4282 (2016)
6. Kumar, K., Lu, Y.H.: Cloud computing for mobile users: can offloading computation save energy? Computer **43**(4), 51–56 (2010)
7. Dinh, H.T., Lee, C., Niyato, D., Wang, P.: A survey of mobile cloud computing: architecture, applications, and approaches. Wirel. Commun. Mob. Comput. **13**(18), 1587–1611 (2013)
8. Van, H.N., Tran, F.D., Menaud, J.M.: Performance and power management for cloud infrastructures. In: 2010 IEEE 3rd International Conference Cloud Computing (CLOUD), pp. 329–336 (2010)
9. Chun, B.G., Ihm, S., Maniatis, P., Naik, M., Patti, A.: CloneCloud: elastic execution between mobile device and cloud. In: Proceedings of the Sixth Conference on Computer Systems, pp. 301–314 (2011)
10. Kosta, S., Aucinas, A., Hui, P., Mortier, R., Zhang, X.: ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: INFOCOM, 2012 Proceedings IEEE, pp. 945–953 (2012)
11. Xiao, Z., Song, W., Chen, Q.: Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1107–1117 (2013)
12. Orsini, G., Bade, D., Lamersdorf, W.: Context-aware computation offloading for mobile cloud computing: requirements analysis, survey and design guideline. Procedia Comput. Sci. **56**, 10–17 (2015)
13. Mao, Y., Zhang, J., Letaief, K.B.: Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J. Sel. Areas Commun. **34**(12), 3590–3605 (2016)
14. Chen, X., Jiao, L., Li, W., Fu, X.: Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw. **24**(5), 2795–2808 (2016)
15. Haubenwaller, A.M., Vandikas, K.: Computations on the edge in the internet of Things. Procedia Comput. Sci. **52**, 29–34 (2015)
16. Sardellitti, S., Scutari, G., Barbarossa, S.: Joint optimization of radio and computational resources for multicell mobile-edge computing. IEEE Trans. Sig. Inf. Process. Netw. **1**(2), 89–103 (2015)
17. Zhao, T., Zhou, S., Guo, X., Zhao, Y., Niu, Z.: A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing. In: 2015 IEEE Globecom Workshops (GC Wkshps), pp. 1–6 (2015)
18. Ge, Y., Zhang, Y., Qiu, Q., Lu, Y.H.: A game theoretic resource allocation for overall energy minimization in mobile cloud computing system. In: Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design, pp. 279–284 (2012)
19. Wang, X., Giannakis, G.B.: Power-efficient resource allocation for time-division multiple access over fading channels. IEEE Trans. Inf. Theory **54**(3), 1225–1240 (2008)
20. You, C., Huang, K., Chae, H., Kim, B.H.: Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans. Wirel. Commun. **16**(3), 1397–1411 (2017)