Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
() : –

doi:10.3906/elk-1806-112

A distributed ADMM approach for energy-efficient resource allocation in mobile edge computing

Weiwei FANG[1]*, Wenchen ZHOU[1], Yangyang LI[2], Xuening YAO[1], Feng XUE[1], Naixue XIONG[3]
[1]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, P.R. China
[2]Innovation Center, China Academy of Electronics and Information Technology, Beijing, P.R. China
[3]Department of Mathematics and Computer Science, Northeastern State University, Tahlequah, OK, USA

Received: 14.06.2018 • **Accepted/Published Online:** 17.09.2018 • **Final Version:** ..201

Abstract: Mobile edge computing (MEC) is a new promising technique to provide cloud-computing capabilities at the edge of cellular networks in close proximity to mobile users. In this paper, we consider joint optimization of the communication and computation resources in a multiuser, multiserver MEC system. The objective of this optimization problem is to minimize the total energy consumption of mobile devices under the time-sharing constraint. Given the fact that no coordination is involved between mobile devices, we propose a light-weight and decentralized algorithm based on the alternating direction method of multipliers (ADMM) framework. Experimental results demonstrate that the proposed algorithm performs well in terms of convergence and outperforms the conventional centralized approach.

Key words: Mobile edge computing, resource allocation, energy efficient, ADMM

1. Introduction

With the prevalence of mobile computing, more and more new mobile applications, e.g., augmented reality, interactive gaming, and video streaming, are emerging and posing stringent requirements for intensive computation and tight latency [1]. However, current mobile devices generally possess limited local resources, which are insufficient to support sophisticated applications. To address this issue, mobile edge computing (MEC) was recently proposed as a promising paradigm that extends cloud computing capabilities to the edge of radio access networks [2, 3]. This paradigm enables telecom operators to deploy resource-rich base station clouds so that nearby mobile devices can access pervasive and agile computation services whenever needed. Compared to mobile cloud computing, MEC has more advantages in terms of communication latency, energy consumption, and back-haul load [4].

Since both computation latency and energy consumption are critical for resource-constrained mobile devices, the design of computation offloading schemes has attracted considerable attention for MEC systems. On one hand, some earlier studies concentrate on task offloading from a single user to a dedicated MEC server. Wang et al. [5] jointly optimized the computational speed, transmit power, and offloading ratio for mobile devices with two system design objectives: energy consumption minimization and execution delay minimization. A Markov decision process approach was proposed in [6] to handle the power-constrained delay minimization problem in MEC. You et al. [7] proposed a framework in which a MD can not only process computation tasks at a local CPU or offload them to the MEC server but can also harvest energy from the base station by microwave

*Correspondence: wwfang@bjtu.edu.cn

power transfer (MPT) technology. In [8], a dynamic computation offloading algorithm based on Lyapunov optimization was introduced to minimize the execution cost in a green MEC system with energy-harvesting mobile devices. On the other hand, recent studies mainly focus on task offloading and resource allocation in multiuser MEC systems. You et al. [2] designed centralized resource allocation to achieve minimum weighted sum mobile energy consumption in the multiuser scenario. A task offloading scheduling algorithm was proposed in [9], which considers minimization of mobile energy consumption under the constraints of resource capacity and computation latency. To sum up, most existing studies only take scenarios with a single MEC server into consideration. However, with the proliferation of small-cell base stations [4], mobile devices will be enabled to choose the most eligible edge server from those nearby to optimally offload computation tasks [10].

The goal of this paper is to propose a joint optimization of communication and computation resources for task offloading in a multiuser, multiserver MEC scenario. The main contributions are summarized as follows: 1- An optimization problem is formulated to minimize the total mobile energy consumption under the time-sharing constraint. 2- For reasons of performance, scalability, and robustness, we develop a new algorithm to solve this problem in a distributed fashion based on the alternating direction method of multipliers (ADMM). The proposed algorithm naturally decomposes computations onto the mobile devices and the MEC servers, making them autonomously collaborate to iteratively reach the optimum. 3- We conduct simulations to show that our algorithm can converge fast in dozens of iterations to a near optimum and can achieve significant energy savings for mobile devices.

The rest of this paper is organized as follows. Section 2 presents the system model and formulation of the optimization problem. Then the proposed ADMM-based algorithm is introduced in Section 3. Section 4 demonstrates the simulation results. Finally, we conclude the paper in Section 5.

2. System model and problem formulation

2.1. System model

We consider a MEC system consisting of M mobile devices (MDs) and N MEC servers in the same geographic area. Time is divided into equal-length slots. Each slot lasts for a duration of T seconds, where T is chosen to satisfy the latency requirement of mobile users. Considering an arbitrary time slot, the mth MD is required to compute R_m bit input data within this slot. Similar to [5–7], in this paper we assume that the considered computation tasks are dividable and offloadable. The MD m can choose to execute its computation task either locally by its own CPU or remotely at a MEC server.

For the local computation, we assume that the mobile CPU is operating at fixed frequency for data computation at each MD and may vary over different MDs. Following the model in [2], we denote C_m as the number of required CPU cycles for processing one bit of input data at MD m, and P_m^{cpu} as the energy cost per CPU cycle for local computation at MD m. Then, if b_m bits of input data are computed locally, the computational energy consumption of MD m can be given by $E_m^{cpu} = b_m C_m P_m^{cpu}$.

For the MEC offloading, we assume that a single MEC server can serve for multiple nearby MDs, each of which will be allocated a fraction of the time slot for data transmission. In any time slot, a MD can determine whether or not to offload some of its computation workloads to any one of the MEC servers. The fraction of time slot allocated by MEC server n to MD m for computation offloading is denoted as $t_{m,n}$, where $t_{m,n} \in [0, T]$. The time duration and energy consumption for MEC computing and result downloading are assumed to be negligible in resource allocation [2, 8]. For MD m and MEC server n, we denote by $r_{m,n}$ the achievable rate

(in bits/s), which can be given as:

$$r_{m,n} = B \log_2 \left(1 + \frac{P_{m,n}^{tx} h_{m,n}^2}{N_0} \right), \tag{1}$$

where B is the wireless bandwidth, $P_{m,n}^{tx}$ is the transmission power of m offloading computation to n, $h_{m,n}$ is the channel gain from m to n, and N_0 is the variance of white Gaussian noise. Accordingly, the total amount of offloaded data for MD m is given as $\sum_{n=1}^{N} r_{m,n} t_{m,n}$, and the energy consumption for computation offloading at MD m can be given by $E_m^{off} = \sum_{n=1}^{N} P_{m,n}^{tx} t_{m,n}$.

2.2. Problem formulation

We are now in a position to formally formulate the energy-efficient resource allocation problem as an optimization that minimizes the overall energy consumption of MDs, as follows:

$$\min_{\{t\}} \quad \sum_{m=1}^{M} \left[\sum_{n=1}^{N} P_{m,n}^{tx} t_{m,n} + (R_m - \sum_{n=1}^{N} r_{m,n} t_{m,n}) C_m P_m^{cpu} \right] \tag{2}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} t_{m,n} \leq T, \quad \forall n, \tag{3}$$

$$\sum_{n=1}^{N} r_{m,n} t_{m,n} \leq R_m, \quad \forall m, \tag{4}$$

$$\frac{t_{m,n}}{T} \leq x_{m,n}, \quad \forall m, n, \tag{5}$$

$$\sum_{n=1}^{N} x_{m,n} = 1, \quad \forall m, \tag{6}$$

$$\text{var.} \quad t_{m,n} \in [0, T], \quad \forall m, n, \tag{7}$$

$$x_{m,n} \in \{0, 1\}, \quad \forall m, n. \tag{8}$$

Here, the objective is to minimize the total energy consumption of all MDs in the system for an arbitrary time slot, and $x_{m,n}$ is defined to indicate whether assigning the tasks of MD m to the MEC n. The constraint of Eq. (3) ensures that the total allocated time duration to MDs by each MEC server is no more than T. The constraint of Eq. (4) ensures that each MD can be offloaded at most all amounts of input data to any MEC server. Constraints of Eqs. (5) and (6) jointly ensure that each MD can only be associated with and served by one MEC server for computation offloading, because offloading data to different MEC servers during the same time slot will incur extra overheads for the MD [2]. At the beginning of each time slot, this problem should be solved so as to optimally allocate resources to all MDs for processing their tasks in this slot.

The optimization problem of Eq. (2) is a mixed-integer convex programming problem [11], which is NP-hard in general. Such a problem is usually transformed and solved by convex relaxation [12], i.e. removing the integer constraint of Eq. (8) or replacing it with $x_{m,n} \in [0, 1]$, $\forall m, n$. Then the transformed problem can be solved in a centralized way by some conventional solvers, and the exact solution can further be obtained

by various global or local methods [12]. However, the number of iterations the solver requires in conventional methods is directly related to the size of problem. In typical use scenarios, the number of MEC servers can be $O(10) - O(10^2)$, and the number of MDs can be $O(10^2) - O(10^3)$ or even more [4]. Therefore, it is very inefficient to solve such a large-scale problem in a centralized way. Furthermore, a central controller has to be deployed to collect indispensable private information from all MDs and MEC servers, solve the optimization problem accordingly, and transfer the allocation results back to each server and each MD. Such operations will bring about excessive computation latency and significant communication burdens for the whole system. These observations motivate us to develop a scalable and robust decentralized algorithm that is easily amenable to practical implementations.

3. Algorithm design

Our algorithm is based on ADMM [13], a simple yet very effective method that solves large-scale convex optimization problems without suffering from the aforementioned drawbacks. However, our problem in Eq. (2) cannot be directly solved by ADMM. The reasons for this are twofold. First, the Boolean variables $\{x_{m,n}\}$ comprise a nonconvex set, rather than the nonempty polyhedral set required by the ADMM framework. Secondly, the constraints of Eqs. (3) and (4) couple all variables together, while in ADMM problems the constraints should be separable for each set of variables.

According to [12], the nonconvex constraints can be relaxed by removing the constraints of Eqs. (5), (6), and (8). Note that we will add them in the update procedure of $t_{m,n}$ so as to ensure that the final solutions satisfy these constraints [12]. For the second challenge, the coupling restrains the problem of Eq. (2) from being solved in a distributed manner. To this end, we introduce a new set of auxiliary variables $s_{m,n} = t_{m,n}$, $\forall m, n$, and reformulate the problem (2) as follows:

$$\min_{\{t,s\}} \quad \sum_{n=1}^{N}\sum_{m=1}^{M} P_{m,n}^{tx} s_{m,n} + \sum_{m=1}^{M}(R_m - \sum_{n=1}^{N} r_{m,n} t_{m,n}) C_m P_m^{cpu} \tag{9}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} s_{m,n} \le T, \quad \forall n, \tag{10}$$

$$(4),$$

$$\text{var.} \quad (7),$$

$$s_{m,n} = t_{m,n}, \quad \forall m, n. \tag{11}$$

According to the ADMM framework, we can formulate the augmented Lagrangian of Eq. (9) as:

$$\begin{aligned} L_\rho(t,s,y) = &\sum_{n=1}^{N}\sum_{m=1}^{M} P_{m,n}^{tx} s_{m,n} + \sum_{m=1}^{M}(R_m - \sum_{n=1}^{N} r_{m,n} t_{m,n}) C_m P_m^{cpu} \\ &+ \sum_{m=1}^{M}\sum_{n=1}^{N} y_{m,n}(t_{m,n} - s_{m,n}) + \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{\rho}{2}(t_{m,n} - s_{m,n})^2, \end{aligned} \tag{12}$$

where $\rho > 0$ is the penalty parameter [13], and $y_{m,n}$ is the dual variable for the equality constraint in Eq. (11).

Then, based on ADMM, Eq. (12) can be solved by updating t, s, and y sequentially [13, Chapter 3]. Specifically, for the $(k+1)$th iteration:

1. t-update Each MD m solves the following subproblem for $t_{m,n}^{k+1}$:

$$\begin{aligned} \min_{\{t_m\}} \quad & \sum_{n=1}^{N}[\frac{\rho}{2}t_{m,n}^2 + (y_{m,n}^k - \rho s_{m,n}^k - r_{m,n}C_m P_m^{cpu})t_{m,n}] \\ \text{s.t.} \quad & (4), (5), (6), \\ \text{var.} \quad & (8). \end{aligned} \tag{13}$$

The subproblem in Eq. (13) belongs to mixed-integer nonlinear programming and is expected to be NP-hard. Fortunately, since MD m can only choose one of the MEC servers (e.g., server $i \in \{1, ..., N\}$) to offload its task, we know that in the optimal solution $t_{m,i}^{OPT} > 0$ while $t_{m,j}^{OPT} = 0$, $\forall j \in \{1, ..., N\}$ and $j \neq i$. Then Eq. (13) can be transformed into the following problem:

$$\begin{aligned} \min_{\{n\}} \min_{t_{m,n}} \quad & h(t_{m,n}) = \frac{\rho}{2}t_{m,n}^2 + (y_{m,n}^k - \rho s_{m,n}^k - r_{m,n}C_m P_m^{cpu})t_{m,n} \\ \text{s.t.} \quad & r_{m,n}t_{m,n} \leq R_m, \\ \text{var.} \quad & (8). \end{aligned} \tag{14}$$

To solve Eq. (14), MD m first finds the optimal solution $t_{m,n}^*$ that minimizes $h(t_{m,n})$, for each $n \in \{1, ..., N\}$. Then, from the resulting set of $t_{m,n}^*$, $t_{m,i}^{OPT}$ is chosen as the one that makes $h(t_{m,i}^{OPT})$ the minimal element in the set of $h(t_{m,n}^*)$. In this procedure, $t_{m,n}^*$ can be calculated as follows:

$$t_{m,n}^* = \begin{cases} a_{m,n}, & \frac{r_{m,n}C_m P_m^{cpu} + \rho s_{m,n}^k - y_{m,n}^k}{\rho} > a_{m,n}, \\ 0, & \frac{r_{m,n}C_m P_m^{cpu} + \rho s_{m,n}^k - y_{m,n}^k}{\rho} < 0, \\ \frac{r_{m,n}C_m P_m^{cpu} + \rho s_{m,n}^k - y_{m,n}^k}{\rho}, & Otherwise, \end{cases} \tag{15}$$

where $a_{m,n} = \min\{\frac{R_m}{r_{m,n}}, T\}$.

2. s-update Each MEC server n solves the following subproblem for $s_{m,n}^{k+1}$:

$$\begin{aligned} \min_{\{s_n\}} \quad & \sum_{m=1}^{M}[\frac{\rho}{2}s_{m,n}^2 + (P_{m,n}^{tx} - \rho t_{m,n}^{k+1} - y_{m,n}^k)s_{m,n}] \\ \text{s.t.} \quad & (10), \end{aligned} \tag{16}$$

$$\text{var.} \quad s_{m,n} \in [0, T], \quad \forall m, n. \tag{17}$$

This subproblem is a small-scale quadratic problem that has only M variables and can be easily solved with the standard convex optimization technique [14].

3. y-update With the optimal $t_{m,n}^{k+1}$ and $s_{m,n}^{k+1}$, the final step is to update the dual variables:

$$y_{m,n}^{k+1} = y_{m,n}^k + \rho(t_{m,n}^{k+1} - s_{m,n}^{k+1}). \tag{18}$$

The above three update procedures are conducted alternatively until convergence. As depicted in Algorithm 1, the distributed nature of this algorithm allows for a very efficient parallel implementation in the multiuser multiserver MEC system. Step 2 involves solving the subproblem of Eq. (14) with N variables, which can be implemented in parallel at M MDs. Steps 3 and 4 involve solving the subproblems of Eq. (16) and (18) with M variables, which can also be implemented in parallel at N MEC servers.

Algorithm 1 Distributed solution to Eq. (2) using ADMM

1: Each MEC server n initializes $s_{m,n}^0 = 0,\ y_{m,n}^0 = 0,\ \forall m$.
2: Given $s_m^k = [s_{m,1}^k, s_{m,2}^k, ...]$ and $y_m^k = [y_{m,1}^k, y_{m,2}^k, ...]$, each MD m solves subproblem (14) for the optimal solution $t_{m,n}^{k+1}$ and sends $t_{m,n}^{k+1}$ to the corresponding MEC server n.
3: Given $t_n^{k+1} = [t_{1,n}^{k+1}, t_{2,n}^{k+1}, ...]$, each MEC sever n solves subproblem (16) for the optimal solution $s_{m,n}^{k+1}$ with local information $y_n^k = [y_{1,n}^k, y_{2,n}^k, ...]$.
4: Each MEC server n updates the dual variables $y_n^{k+1} = [y_{1,n}^{k+1}, y_{2,n}^{k+1}, ...]$ as in Eq. (18). Then n sends $s_{m,n}^{k+1}$ and the dual variable $y_{m,n}^{k+1}$ to the corresponding MD m.
5: Return to Step 2 until convergence.

This new algorithm will inevitably bring about additional energy consumption in computation and communication in the solving process. However, such costs are often very small and negligible [15–17], as compared to those for data processing.

4. Numerical results

In this section, we evaluate the proposed distributed MEC offloading algorithm by conducting MATLAB-based simulations. The parameters are set as follows unless specified otherwise. There are $M = 100$ mobile devices and $N = 20$ MEC servers in the system. The time slot is $T = 2$ s. For the wireless channel, we choose $B = 1$ MHz, $N_0 = 10^{-9}$, and $h_{m,n}$ is modeled as independent Rayleigh fading with average value as 1 [2]. The transmission power is $P_{m,n}^{tx} = 0.01$ W [18]. For the computation task, both the data size and the required energy consumption per bit follow the uniform distribution with $R_m \in [0, 100]$ Mb and $C_m P_m^{cpu} \in [516.5, 1116.5]$ W/b [5]. The penalty parameter ρ is set to 0.5. Furthermore, we define the primal residual as $r^{k+1} = t^{k+1} - s^{k+1}$ and then choose $\|r^k\|_2 \leq 2 \times 10^{-4}$ as the convergence criterion [13]. For comparison, we use a well known solver, "lp_solve", to solve the problem of Eq. (2) in the conventional centralized way [19].

To understand the algorithm performance on a microscopic level, we plot a representative execution result in Figure 1 and Figure 2. The total energy consumption achieved by our algorithm is much higher than the optimal value at the beginning of iterations, since the constraints in Eq. (11) cannot always be satisfied. By enforcing the regularization terms for coupling constraints, the proposed algorithm would gradually satisfy these constraints in a few iterations. Eventually, it converges to modest accuracy after 60 iterations, with a gap of only 1.46‰. After that, we can observe that the algorithm still attempts to converge to high accuracy, although very slowly. These observations indicate that our ADMM-based algorithm can produce acceptable results of practical use in a few dozen iterations.

Figure 3 demonstrates the effects of varying parameter ρ on algorithm convergence. We can observe that the results with different ρ finally converge to almost the same optimal value with only a slight gap. However, the choice of ρ affects the convergence rate of our algorithm, i.e. setting a smaller ρ results in a faster convergence, especially before the 100th iteration.

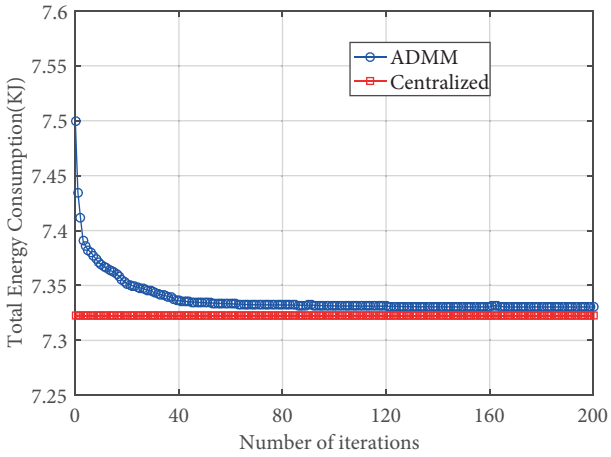

Figure 1. Total energy consumption convergence versus the number of iterations.

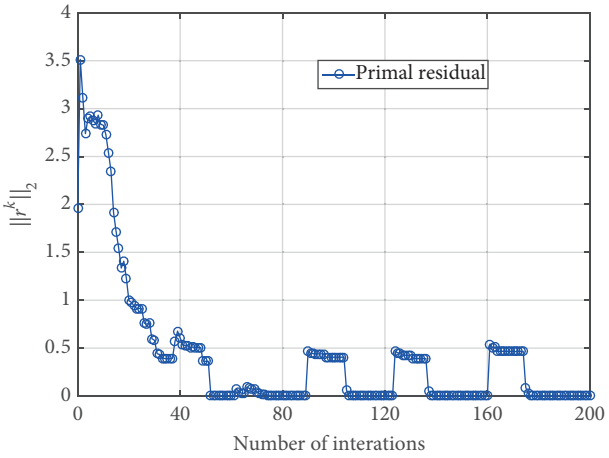

Figure 2. Primal residual versus the number of iterations.

Figure 4 plots the CDF of the number of iterations the two algorithms take to achieve convergence for 15 runs. It is clear that the proposed algorithm converges much faster than the centralized method. Our algorithm takes at most 93 iterations to converge, while the centralized method takes at least 519 iterations. For 80% of the time our algorithm converges within 83 iterations, while the centralized method takes 653 iterations. These results show the fast convergence of our algorithm compared to conventional methods.

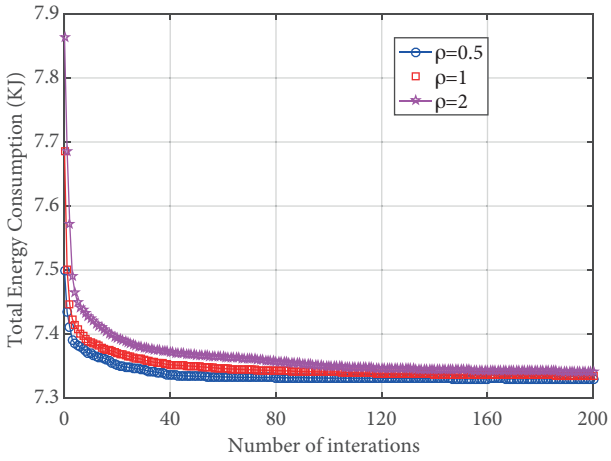

Figure 3. The effects of ρ on convergence of ADMM algorithm.

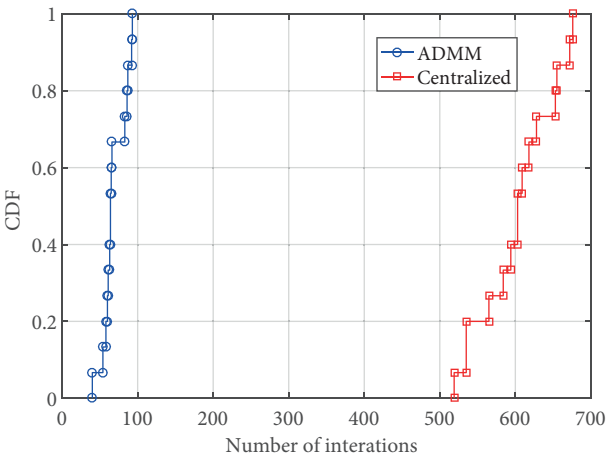

Figure 4. CDF of the number of iterations to achieve convergence.

Then we investigate the performance of our algorithm in different system parameter conditions. Figure 5 shows that the higher the wireless bandwidth is, the smaller the total energy consumption of all MDs is. This corroborates Eqs. (1) and (2) as MDs are more capable to offload their tasks to MEC servers with higher link rates. Figure 6 illustrates the impacts of varying the transmission power of a MD. As the transmission power increases, the energy consumption on wireless transmission will be increased while the energy consumption in local computation will be reduced. According to Figure 6, using a higher transmission power is more favorable for reducing the total energy consumption of MDs. Both Figures 5 and 6 show that even when some key system parameters are changed, our algorithm can still converge in a few dozen iterations.

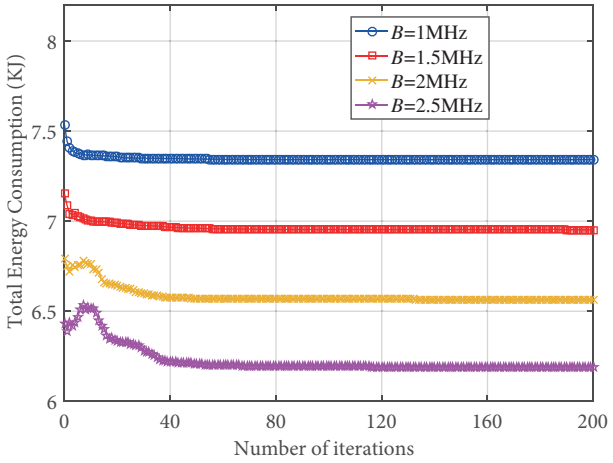

Figure 5. Impacts of wireless bandwidth on energy consumption and algorithm convergence.

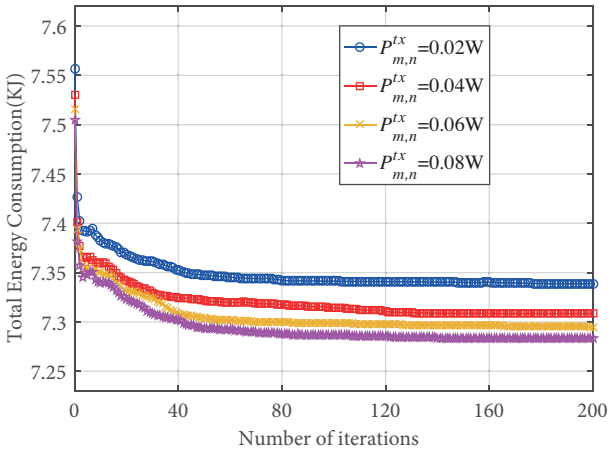

Figure 6. Impacts of transmission power on energy consumption and algorithm convergence.

Finally, we vary the number of MEC servers N to study its impact on algorithm performance. The results are shown in Figure 7. As N increases, the MDs will have more options and better choices to offload their tasks, resulting in significant reductions in energy consumption. For example, when $N = 40$, the total energy consumption is reduced by 21.5% as compared to the case without task offloading (i.e. when $N = 0$). However, the growth in the number of variables slows down the convergence speed of the proposed algorithm according to our definition of the convergence criterion in Section 4. For instance, the algorithm converges after about 16 iterations when $N = 1$, about 39 iterations when $N = 10$, and about 115 iterations when $N = 40$.

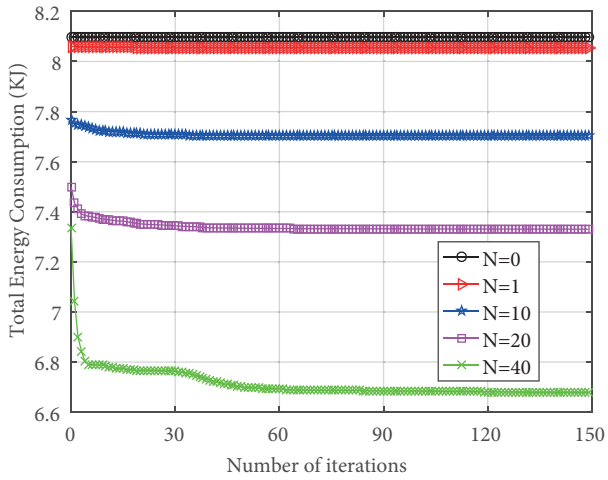

Figure 7. Impacts of the number of MEC servers on energy consumption and algorithm convergence.

5. Conclusion

In this paper, we investigate energy-efficient resource allocation with the consideration on the latency requirement for mobile edge computing. We formulate the problem as a mixed-integer convex optimization, which minimizes the total energy consumption of all mobile devices in the system. We designed an efficient distributed algorithm based on the ADMM framework, which can decompose the original problem into a number of small-

scale subproblems that can be effectively and efficiently solved. Simulation experiments are conducted to verify the advantages of the proposed algorithm in many aspects.

Acknowledgment

This work was supported by the Fundamental Research Funds for the Central Universities of China under Grants 2017JBM021 and 2016JBZ006, and CETC.

References

[1] Satyanarayanan M. The emergence of edge computing. Computer 2017; 50: 30-39.

[2] You C, Huang K, Chae H, Kim BH. Energy-efficient resource allocation for mobile-edge computation offloading. IEEE T Wirel Commun 2017; 16: 1397-1411.

[3] Wang S, Zhang X, Zhang Y, Wang L, Yang J, Wang W. A survey on mobile edge networks: convergence of computing, caching and communications. IEEE Access 2017; 5: 6757-6779.

[4] Mao Y, You C, Zhang J, Huang K, Letaief KB. A survey on mobile edge computing: the communication perspective. IEEE Commun Surveys Tuts 2017; 19: 2322-2358.

[5] Wang Y, Sheng M, Wang X, Wang L, Li J. Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. IEEE T Commun 2016; 64: 4268-4282.

[6] Liu J, Mao Y, Zhang J, Letaief KB. Delay-optimal computation task scheduling for mobile-edge computing systems. In: IEEE 2016 International Symposium on Information Theory; 10–15 July 2016; Barcelona, Spain. New York, NY, USA: IEEE. pp. 1451-1455.

[7] You C, Huang K, Chae H. Energy efficient mobile cloud computing powered by wireless energy transfer. IEEE J Sel Area Comm 2016; 34: 1757-1771.

[8] Mao Y, Zhang J, Letaief KB. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J Sel Area Comm; 34: 3590-3605.

[9] Tao X, Ota K, Dong M, Qi H, Li K. Performance guaranteed computation offloading for mobile-edge cloud computing. IEEE Wirel Commun Le 2017; 6: 774-777.

[10] Dinh TQ, Tang J, La QD, Quek TQS. Adaptive computation scaling and task offloading in mobile edge computing. In: IEEE 2017 Wireless Communications and Networking Conference; 19–22 March 2017; San Francisco, CA, USA. New York, NY, USA: IEEE. pp. 1-6.

[11] Lubin M, Yamangil E, Bent R, Vielma JP. Extended formulations in mixed-integer convex programming. In: 2016 Integer Programming and Combinatorial Optimization Conference; 1–3 June 2016; Liège, Belgium. Cham, Switzerland: Springer. pp. 102-113.

[12] Diamond S, Takapoui R, Boyd S. A general system for heuristic minimization of convex functions over non-convex sets. Optim Method Softw 2018; 33: 165-193.

[13] Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn 2011; 3: 1-122.

[14] Boyd S, Vandenberghe L. Convex Optimization. New York, NY, USA: Cambridge University Press, 2004.

[15] Liu B, Cao Y, Wang W, Jiang T. Energy budget aware device-to-device cooperation for mobile videos. In: IEEE 2015 Global Communications Conference; 6–10 December 2015; San Diego, CA, USA. New York, NY, USA: IEEE. pp. 1-7.

[16] Liu M, Mao Y, Leng S, Mao S. Full-duplex aided user virtualization for mobile edge computing in 5g networks. IEEE Access 2018; 6: 2996-3007.

[17] Chang Z, Gong J, Li Y, Zhou Z, Ristaniemi T, Shi G, Han Z, Niu Z. Energy efficient resource allocation for wireless power transfer enabled collaborative mobile clouds. IEEE J Sel Area Comm 2016; 34: 3438-3450.

[18] Fang W, Li Y, Zhang H, Xiong N, Lai J, Vasilakos AV. On the throughput-energy tradeoff for data transmission between cloud and mobile devices. Inform Sciences 2014; 283: 79-93.

[19] Kim HG. A connection method of lpsolve and excel for network optimization problem. J Korea Ind Inf Syst Res 2010; 15: 187-196.