Energy Efficient Mobile Edge Computing using Joint Benders Decomposition and Distributed Dinkelbach Algorithm

Ye Yu* †, Xiangyuan Bu*, Kai Yang*, and Zhu Han†
* School of Information and Electronics, Beijing Institute of Technology, Beijing, China
† Department of Electrical and Computer Engineering, University of Houston, Houston, USA

***Abstract*—Currently, executing computation intensive and time sensitive tasks among the network becomes a significant challenge. Traditional cloud computing executes the task with high latency and energy cost. Mobile edge computing (MEC) is proposed as a supplement to cloud computing. In this paper, we formulate a problem to minimize the energy cost in MEC, considering transmit power and latency constraints. To solve the proposed mixed integer nonlinear programming problem, we propose a joint Benders decomposition and distributed Dinkelbach algorithm. The Benders decomposition performs as an outer loop algorithm, which separates the original problem into the subproblem and master problem. The distributed Dinkelbach algorithm solves subproblem in the inner loop in a distributed manner. The simulation results show that our proposed algorithm is energy efficient with high reliability.**

***Index Terms*—Mobile edge computing, energy efficient, Benders decomposition, distributed Dinkelbach algorithm, resource allocation.**

I. Introduction

Currently, mobile edge computing (MEC) becomes a popular topic in developing the new generation of the wireless communication network. Different from the traditional cloud computing, MEC is much closer to the user devices with low latency and better mobility. The computing nodes and small servers provided by MEC at the edge of the network give mobile users larger computing capability. Some heavy tasks may be impossible for local users to execute, especially for some Internet of Things (IoT) applications. Also, MEC can reduce the latency comparing to cloud computing. Although the MEC has a limited capacity for certain tasks, it performs as a good tradeoff between computing and delay.

Energy efficient design is significant in MEC. This is due to the maximum available energy of the users' devices are constrained by battery capacity [1]. There are many energy-consuming tasks for the mobile devices, such as playing 3-D games and editing videos. By offloading computation-intensive tasks to the edge cloud, mobile user devices can reduce energy consumption remarkably. The energy cost in the local device is more critical since the edge cloud usually keeps charging. Although the users can choose to offload the tasks to the edge cloud, both communication and local computing cost a large amount of energy. Thus, energy efficient algorithms are often proposed to save the battery life of the devices. Moreover, beamforming and MIMO can be introduced to reduce the costs of communicating [2].

Although the energy efficient and latency constrained MEC network has been studying deeply and widely, there is little work gives the general algorithm framework for the mixed integer programming in this scenario. In this paper, we consider the energy efficient MEC network. Our goal is to minimize the energy cost with the constraints of transmit power and latency. The main contributions of this paper are summarized as follows.

- We formulate the energy efficient MEC optimization problem as a mixed integer nonlinear programming problem (MINLP). The transmit power and latency constraints are both considered.
- We propose an algorithm framework by joint Benders decomposition and distributed Dinkelbach algorithm. Benders decomposition is adopted for separating the original problem into a subproblem and master problem. The distributed Dinkelbach algorithm is used for dealing with the fractional programming form of the subproblem distributedly. The algorithm framework is easy for solving general mixed integer programming problems in mobile edge computing.
- We implement and verify our proposed algorithm in the section of simulation results. The convergence of our algorithm is fast and the solution is the global optimal solution. The rest of the simulation results show the good performance of the algorithm.

The rest of the paper is organized as follows. In Section II, the related work on mobile edge computing in different aspects is presented. In Section III, we introduce the system model and formulate the optimization problem for the MEC network. In Section IV, we give the whole procedure and discussions of the proposed algorithm. The simulation results are shown in Section V, and Section VI concludes the paper.

II. Related Work

Nowadays, MEC network is considered as a crucial technique to alleviate the burden of the network and improve the Quality of Service (QoS). Many researchers propose new designs and architectures to enhance the performance of edge computing. In [4], the authors propose a data-driven model to optimize the unmanned aerial vehicles (UAVs) aided edge computing network where the users are smart vehicles detecting cyber-threat using a probabilistic data structure (PDS)- based approach. In [5], the authors propose coded computation in the mobile edge computing with the benefits of optimality, universality, and mobility. In [6], the authors propose an IoT architecture for the quick service restaurants

and describe various edge computing applications. In [7], the authors propose the Stackelberg game-based ADMM to solve the proactive caching in large-scale mobile edge networks. In [8], the authors give a framework of edge computing for healthcare application, which using deep learning with high accuracy. In [9], the authors introduce the former standards for edge computing and propose a novel framework with moving resources closer distance to the users to reduce the signaling and latency for IoT services.

Energy efficiency is an important research area for MEC. Not only energy efficient algorithms are proposed, but also architecture and battery techniques are improved. In [10], the authors propose a user-centric energy-aware mobility management (EMM) scheme in Ultra-Dense Networks (UDN). In [11], the authors propose a novel framework for small-cell base stations (SBSs) edge computing by maximizing the system performance and constrained by energy. In [12], the authors consider energy consumption of offloading in small cells from both task computation and communication aspects, including fronthaul and backhaul. In [13], the authors give a framework in edge computing with renewable energy resources, which edge computing system and power supply system should cooperate together. In [14], the authors propose an energy-efficient vehicular edge computing (VEC) framework with battery constraint, which is solved by an alternating direction method of multipliers (ADMM)-based energy-efficient resource allocation algorithm.

Latency is another key factor in computing, which is often the major requirement for many applications. Thus, many works have been down considering both energy and latency in MEC networks. In [15], the authors joint optimize communication and computing in MEC with energy and latency constraints. In [16], the authors propose the mobility-aware hierarchical MEC framework with low energy cost and latency. In [17], the authors analyze both the computation latency and communication latency in the MEC network composed by a radio access network cascaded with a CS network. In [18], the authors give an online dynamic tasks assignment for the MEC system with energy harvesting (EH) ability to make the tradeoff between energy and latency.

III. System Model and Problem Formulation

In this section, we first introduce the frame of our system. Then, the problem of the network is formulated.

A. System Model

We consider there are N mobile device users within the edge cloud network. For each user, a computation task needs to be executed locally or offload to the edge cloud. The task model is assumed as the same as [19] which denoted as $U_n = (\omega_n, s_n)$, where ω_n is the overall CPU cycles per bit for the task and s_n is the data length of the task. In this paper, we assume that the task is not divisible which means the users can only choose to compute task locally or offload to the edge cloud. For the n^{th} user, we use h_n to denote the channel gain between the edge cloud and the user. In this paper, we assume that the channel state information (CSI) is known at the edge cloud. In addition, the users do not move during offloading, which means h_n is a constant. We assume that there is no interference among edge computing users. Thus, the uplink data rate for the n^{th} user can be expressed as

$$R_n = Blog_2\left(1 + \frac{p_n h_n}{\sigma^2}\right), \tag{1}$$

where B is the available bandwidth. p_n is the transmit power for the n^{th} user. σ^2 is the noise power of the devices. The MEC has two different kind of delay, which are transmission delay and computing delay. The transmission delay for the n^{th} user is calculated as

$$T_n^t = \frac{s_n}{R_n}. \tag{2}$$

For the computing delay, we assume that the edge cloud allocates Ω_n^e computing resources to the n^{th} user. Thus the computing duration on the edge cloud for the n^{th} user is formulated as

$$T_n^c = \frac{\omega_n}{\Omega_n^e}. \tag{3}$$

Thus, the time duration of the n^{th} user's task on the edge cloud is given as follows

$$T_n^e = T_n^t + T_n^c = \frac{s_n}{R_n} + \frac{\omega_n}{\Omega_n^e}. \tag{4}$$

For the local computing delay, we assume the capability of computing for the n^{th} user is Ω_n^l. Thus, the time consumed at the local user device is as

$$T_n^l = \frac{\omega_n}{\Omega_n^l}. \tag{5}$$

As the common consumption in most papers, we assume that the delay of sending computing results back to the users is not included in our formulation. The reason to do so is that the data length of the result is much smaller than the input data and the data rate downlink is higher than the uplink.

The energy consumptions happen both for local computing users and edge computing users. For the n^{th} local computing user, the energy cost is given as

$$E_n^l = \phi\left(\Omega_n^l\right)^2 \omega_n, \tag{6}$$

which is the same as proposed in [20]. ϕ is the parameter corresponding to the different chip architecture. For the n^{th} edge computing user, the energy is mainly consumed during the transmission, which is shown as

$$E_n^e = p_n T_n^t = \frac{p_n s_n}{R_n}. \tag{7}$$

B. Problem Formulation

We aim to minimize the energy cost for the MEC network. We use δ_n to indicate the computing strategy of the n^{th} user. If the user choose to offload the task to the edge cloud, $\delta_n = 1$. Otherwise, $\delta_n = 0$. Furthermore, we use p_n^{max} to indicate the maximum transmit power for the n^{th} user. T_n^{total} is the computing deadline for the n^{th} task.

Thus, the problem is formulated as

$$\min_{\boldsymbol{\delta},\mathbf{p}} \quad \sum_{n=1}^{N} \delta_n E_n^e + \sum_{n=1}^{N} (1-\delta_n) E_n^l \tag{8}$$

$$\text{s.t} \quad 0 \le p_n \le p_n^{max} \quad \forall n, \tag{9}$$

$$\delta_n T_n^e + (1-\delta_n) T_n^l \le T_n^{total} \quad \forall n, \tag{10}$$

$$\delta_n \in \{0,1\} \quad \forall n, \tag{11}$$

where constraint (9) is the maximum transmit power constraint for all users. Constraint (10) indicates that the delay for the tasks cannot exceed the tolerance. Constraint (11) is the binary constraint for the binary variable.

The problem formulated is an MINLP problem. In the next section, we will propose the joint Benders decomposition and Dinkelbach algorithm to solve problem (8).

IV. Proposed Algorithm

In this section, we propose our algorithm which is composed by the Benders decomposition and Dinkelbach algorithm. The Benders decomposition is the outer loop algorithm, and the Dinkelbach algorithm performs the inner loop algorithm.

A. Benders Decompostiion

Benders decomposition is widely used for solving mixed integer programming. The principle is to separate the problem into two kinds of problems and solve them iteratively. The whole procedure for solving our proposed problem is described as follows.

First, we need to initialize the parameters for the iteration. The loop index i is set as 1. The scalar variable α, which will be explained later, is set to α^{down} with enough low value.

Then, we need to solve the subproblem. The subproblem is formulated by fixing the binary variable of the original problem. Thus, the continuous subproblem is given as

$$\min_{\boldsymbol{\delta},\mathbf{p}} \quad \sum_{n=1}^{N} \delta_n E_n^c \tag{12}$$

$$\text{s.t} \quad (9), (10),$$

$$\delta_n = \delta_n^i, \quad \forall n. \tag{13}$$

δ_n^i is the fixed constant in the i^{th} iteration. We denote that the dual variable for the constraint (13) is γ_n^i. This subproblem is solved by the Dinkelbach algorithm which is presented in the Section IV-B.

Next, we need to generate the upper bound and lower bound for the iteration control. The upper bound is given by obtaining the result of subproblem, which is calculated as

$$U^i = \sum_{n=1}^{N} \delta_n^i \frac{p_n^i s_n}{Blog_2\left(1+\frac{p_n^i h_n}{\sigma^2}\right)} + \sum_{n=1}^{N} (1-\delta_n^i)\phi\left(\Omega_n^l\right)^2 \omega_n, \tag{14}$$

where p_n^i is the optimal solution from the subproblem in the i^{th} iteration. The lower bound is

$$L^i = \alpha^i, \tag{15}$$

Algorithm 1 Benders Decomposition

```
Initialize: loop index i, α^i = α^down.
while U^i − L^i ≥ ε do
    Subproblem
        solve subproblem using Dinkelbach algorithm
    Bounds calculation
        calculate upper and lower bound U^i and L^i
        by (14) and (15)
    Master problem
        step 1: update loop index i = i + 1
        step 2: add new Benders cut to the problem (16)
        step 3: solve the problem in (16) to obtain the
                optimal value of δ and α
end while
```

which is given from the master problem. The iteration stops when the difference between the upper bound and lower bound is less than the pre-defined threshold.Otherwise, the iteration will go on.

Then, we need to solve the master problem. After we update the loop index $i = i+1$, the continuous variable $\mathbf{p}$ is fixed to the previous solution. The master problem is formulated as

$$\min_{\alpha} \quad \alpha \tag{16}$$

$$\text{s.t} \quad \sum_{n=1}^{N} \delta_n^{\mu} \frac{p_n^{\mu} s_n}{Blog_2\left(1+\frac{p_n^{\mu} h_n}{\sigma^2}\right)} + \sum_{n=1}^{N} (1-\delta_n^{\mu})\phi\left(\Omega_n^l\right)^2 \omega_n + \sum_{n=1}^{N} \gamma_n^{\mu}\left(\delta_n - \delta_n^{\mu}\right) \le \alpha \quad \mu = 1, ..., i-1, \tag{17}$$

$$\alpha^{down} \le \alpha, \tag{18}$$

$$(11).$$

Constraint (17) is called the Benders cut, and it is generated from the previous iterations. In each iteration, a new Benders cut will added to the master problem to make a better approximation using hyperplanes. By solving the master problem, we can obtain the optimal solution of the binary variable for solving the subproblem in the next iteration. The whole procedure is described in Algorithm 1.

B. Distributed Dinkelbach Algorithm

The Dinkelbach algorithm is used for transforming the fractional programming into a parametric subtractive form. The continuous subproblem (12) is a fractional programming problem with the transmit power variable $\mathbf{p}$. Thus, we reformulate the objective function in (12) with a continuous auxiliary variable $\boldsymbol{\lambda}$. The objective function, which is a monotonically increasing function, is given as

$$f(\boldsymbol{\lambda}) = \lambda^l \sum_{n=1}^{N} \delta_n^i p_n^i s_n - \sum_{n=1}^{N} Blog_2\left(1+\frac{p_n^i h_n}{\sigma^2}\right). \tag{19}$$

The non-negative λ^l updates in each iteration of the Dinkelbach algorithm.

Algorithm 2 Distributed Dinkelbach Algorithm

Initialize: loop index $l = 0$, maximum tolerance σ and
$\lambda^l = 0$
while $|\lambda^l \sum_{n=1}^{N} \delta_n^i p_n^i s_n - \sum_{n=1}^{N} Blog_2\left(1 + \frac{p_n^i h_n}{\sigma^2}\right)| \geq \sigma$ **do**
  User n: Solve problem (21) to obtain the optimal solution p_n^{l*} and the dual variable γ_n^{l*} with λ^l
  Edge cloud: Updates λ^l as (25) and broadcast to all users
  $l = l + 1$
end while

Therefore, the problem for each iteration is reformulated as

$$\min_{\boldsymbol{\delta},\mathbf{p}} \quad \lambda^l \sum_{n=1}^{N} \delta_n^i p_n^i s_n - \sum_{n=1}^{N} Blog_2\left(1 + \frac{p_n^i h_n}{\sigma^2}\right) \tag{20}$$
$$\text{s.t} \quad (9), (10), \text{ and } (13),$$

which is a convex problem. To promote the utilization of the resources and increase the efficiency, we solve this problem in a distributed manner. For user n, solve the problem in iteration l as follow

$$\min_{\delta_n, p_n} \quad \lambda^l \delta_n^l p_n^l s_n - Blog_2\left(1 + \frac{p_n^l h_n}{\sigma^2}\right) \tag{21}$$
$$\text{s.t} \quad 0 \leq p_n \leq p_n^{max}, \tag{22}$$
$$\delta_n T_n^e + (1 - \delta_n) T_n^l \leq T_n^{total}, \tag{23}$$
$$\delta_n = \delta_n^l, \tag{24}$$

which is easy to solve for each user. After solve the problem, the users send the optimal solution p_n^{l*} and the dual variable γ_n^{l*} back to the edge cloud. Then, the edge cloud update the λ^l as

$$\lambda^l = \frac{\sum_{n=1}^{N} Blog_2\left(1 + \frac{p_n^{l*} h_n}{\sigma^2}\right)}{\sum_{n=1}^{N} \delta_n^l p_n^{l*} s_n}. \tag{25}$$

We note that although the edge cloud only uses the last dual variable for updating the Benders cut constraint in the master problem, the users have no information about the iteration procedure. Thus, users need to send the results back to the edge with extra energy cost. However, the energy cost for sending results and update the small-scale problem (21) is quite negligible. Therefore, we ignore the energy cost during the distributed computing and signaling. The whole procedure of the distributed Dinkelbach algorithm is presented in Algorithm 2.

V. Simulation Results

In this section, we give simulation results to show the effectiveness of our proposed algorithm. We assume that all users are uniformly distributed in a cell with a radius of 100 m. The mobile users are static, and we assume the distance-based path loss and the Rayleigh multi-path fading are included in our channel model. The available bandwidth for transmission

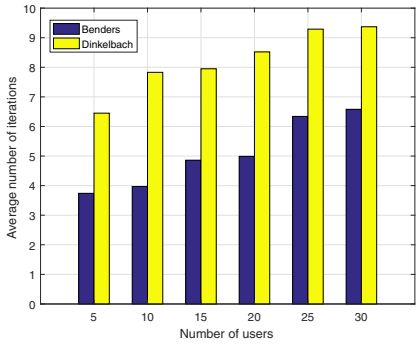

Fig. 1: Average number of iterations versus number of users.

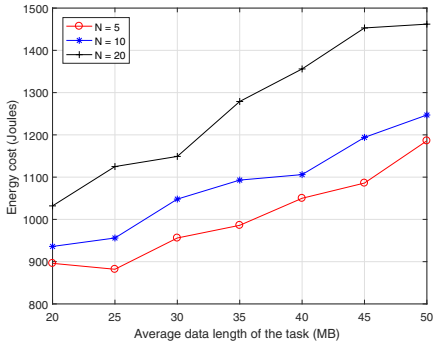

Fig. 2: Energy cost versus average data length of the task.

is set to 20 MHz. The power of Gaussian noise is $2*10^{-13}$ W. We obtain task parameters, such as overall CPU cycle per bit ω_n and the data length s_n, from the probability distribution. The allocated computing resources Ω_n^e for edge computing relate to the task size. The computing capability for each user Ω_n^l is randomly generated lower than the one-tenths of the edge cloud capacity. The initial lower bound of α is set to $\alpha^{down} = -25$. We adopt Matlab with the MOSEK solver for implementing the proposed algorithm. All shown results are generated after over 100 times simulation.

A. Covergence Performance

In Fig. 1, we show the average number of iterations versus the number of users. The maximum transmit power p^{max} is set to 23 dBm, and the user number is set to 10. We can see from the figure that both the outer loop algorithm and inner loop algorithm can converge very fast. Although the iterations increasing with the number of users, the increasing rate is tolerable. The average number of iterations keep below 10 with less than 30 users.

B. Impact of the Task Data Length

In Fig. 2, we give the energy cost of the MEC versus the average data length of the task with different numbers of users. We can see that the energy cost is increasing with the average

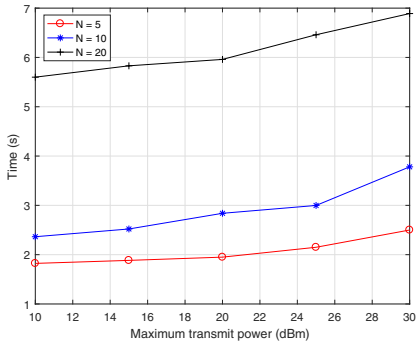

Fig. 3: Running time versus the maximum transmit power.

data length. More average data length will cause more energy from the network to execute it. While the number of users increasing, the energy cost increasing also. This is due to more users will need to execute more tasks, which is quite straightforward.

C. Impact of the Maximum transmit power

Fig. 3 depicts the running time versus the maximum transmit power with different numbers of users. From the figure, we can see that our proposed algorithm increase only a little with the increasing of transmit power. Although more user will increase the running time, it's still tolerable as an iteration algorithm. The main part of the running time consumed at the Dinkelbach algorithm, which is distributedly computed. If we care more about the time, a centralized algorithm will be an alternative.

VI. Conclusions

In this paper, we formulated an energy cost minimization problem including transmit power and latency constraints in mobile edge computing. The proposed problem is MINLP problem. Thus, we proposed a joint Benders decomposition and distributed Dinkelbach algorithm to solve the problem. The Benders decomposition is an outer framework algorithm by dividing the original problem into subproblem and master problem, which are solved iteratively. In subproblem, we use the distributed Dinkelbach algorithm to deal with the fractional programming and solve the problem in a distributed manner. The simulation results have shown that our proposed algorithm has a good performance for the mobile edge computing.

Acknowledgement

This work is partially supported by National Natural Science Foundation of China under Grant 61501028, 61771054, and US National Science Foundation CNS-1702850, CNS-1646607, ECCS-1547201, CCF-1456921, CMMI-1434789, CNS-1443917, and ECCS-1405121.

References

[1] J. An, K. Yang, J. Wu, N. Ye, S. Guo, and Z. Liao, "Achieving sustainable ultra-dense heterogeneous networks for 5g," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 84–90, Dec. 2017.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.

[3] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint radio and computational resource allocation in iot fog computing," *IEEE Transactions on Vehicular Technology*, pp. 1–11, Mar. 2018.

[4] S. Garg, A. Singh, S. Batra, N. Kumar, and L. T. Yang, "Uav-empowered edge computing environment for cyber-threat detection in smart vehicles," *IEEE Network*, vol. 32, no. 3, pp. 42–51, May 2018.

[5] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Architectures for coded mobile edge computing," in *IEEE Fog World Congress (FWC)*, Santa Clara, CA, Oct. 2017.

[6] K. Ayta and . Korak, "Iot edge computing in quick service restaurants," in *16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, Shanghai, China, China, May 2018.

[7] Z. Zheng, L. Song, Z. Han, G. Y. Li, and H. V. Poor, "A stackelberg game approach to proactive caching in large-scale mobile edge networks," *IEEE Transactions on Wireless Communications*, pp. 1–14, May 2018.

[8] G. Muhammad, M. F. Alhamid, M. Alsulaiman, and B. Gupta, "Edge computing with cloud for voice disorder assessment and treatment," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 60–65, Apr. 2018.

[9] S. Husain, A. Kunz, A. Prasad, K. Samdanis, and J. Song, "Mobile edge computing with network resource slicing for internet-of-things," in *IEEE 4th World Forum on Internet of Things (WF-IoT)*, Singapore, Feb. 2018, pp. 1–6.

[10] Y. Sun, S. Zhou, and J. Xu, "Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[11] L. Chen, J. Xu, and S. Zhou, "Computation peer offloading in mobile edge computing with energy budgets," in *IEEE Global Communications Conference*, Singapore, Dec. 2017.

[12] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5g," *IEEE Transactions on Vehicular Technology*, pp. 1–12, Jan. 2018.

[13] W. Li, T. Yang, F. C. Delicato, P. F. Pires, Z. Tari, S. U. Khan, and A. Y. Zomaya, "On enabling sustainable edge computing with renewable energy resources," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 94–101, May 2018.

[14] Z. Zhou, P. Liu, Z. Chang, C. Xu, and Y. Zhang, "Energy-efficient workload offloading and power control in vehicular edge computing," in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Barcelona, Spain, Spain, Apr. 2018, pp. 191–196.

[15] J. Zhang, X. Hu, Z. Ning, E. C. H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency trade-off for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, pp. 1–13, Dec. 2017.

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

[17] S. W. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modelling and latency analysis," *IEEE Transactions on Wireless Communications*, pp. 1–16, Jun. 2018.

[18] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Transactions on Industrial Informatics*, pp. 1–12, Jun. 2018.

[19] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[20] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11 365–11 373, Mar. 2018.