ONLINE LEARNING FOR COMPUTATION PEER OFFLOADING WITH SEMI-BANDIT FEEDBACK

Hongbin Zhu[*†], *Kai Kang*[‡], *Xiliang Luo*[*], *and Hua Qian*[‡]

[*]School of Information Science and Technology, ShanghaiTech University, Shanghai, China
[†]Shanghai Institute of Microsystem and Information Technology, CAS, Shanghai, China
[‡]Shanghai Advanced Research Institute, CAS, Shanghai, China
E-mail: {zhuhb, luoxl}@shanghaitech.edu.cn, {kangk, qianh}@sari.ac.cn

ABSTRACT

Fog computing is emerging as a promising paradigm to perform distributed, low-latency computation. Efficient computation peer offloading is critical to fully utilize the computational resources in fog networks. In this paper, we consider computation peer offloading problem in a fog network with time-varying stochastic time of arrival tasks and channel conditions. Such time-varying conditions are not available to all fog nodes. In order to minimize the latency of accomplishing arrival tasks, we propose an online algorithm based on combinatorial upper confidence bounds algorithm with two uncertain variables under the non-stationary bandit model. The proposed computation offloading policy is optimized based on historical feedback. The performance of the proposed scheme is validated through numerical simulations.

Index Terms— Fog Computing, Computation Peer Offloading, Online Learning, Combinatorial Multi-Armed Bandit (CMAB).

1. INTRODUCTION

Mobile applications, such as augmented reality (AR), online High Definition (HD) live and 3 dimensions (3D) modeling, are becoming more and more popular in our daily life. In general, these applications are latency-sensitive or/and computation-intensive. On the other hand, mobile devices always suffer from limited battery power and computational resources.

Fog computing (a.k.a., an extension of mobile edge computing (MEC)), offers computational capacity along the cloud to edge continuum [1]. In this architecture, mobile end-user devices can offload intensive computational workload to fog nodes (FNs) (e.g., access points, micro base stations) in the vicinity [2, 3]. Computation peer offloading is crucial to balance the uneven distribution of workloads and computation capabilities of FNs [4].

On the other hand, latency of the computation offloading is the most critical performance metric in fog systems. Recently, a variety of computation offloading schemes considering the latency problem have been proposed. Some researchers considered the computation offloading as a deterministic optimization problem, e.g., the joint optimization of radio and computational resource [5]; the joint optimization of latency and energy [6]; and the optimization of energy consumption while satisfying delay constraints [7]. However, the above works do not consider existing condition of real-time information of the fog network, such as, the queue condition of FNs. In this regard, the Lyapunov optimization method was adopted in [8–10] to turn the tricky stochastic programming problem into a sequential decision problem. Besides, the authors in [11] proposed an online secretary scheme to minimize the latency of computation peer offloading under uncertainty on the arrival process of neighboring FNs.

The majority of existing research perform computation peer offloading under (some of) the following assumptions: (i) information availability and (ii) static network characteristics [12, 13]. These assumptions may not be representative in reality. For example, many end users in fog network are mobile, so the network characteristics change over time and some real-time information (e.g., the queue condition of other FNs) is unaware to each FN.

In this paper, we consider a fog network where: (i) the arrival task of every FN is a stochastic and non-stationary random process, where its statistics is not known as a priori and (ii) the FN offloads tasks to other FNs in the vicinity, given no prior information on channel condition and the waiting tasks' queue length of other FNs. Existing algorithms for computation offloading do not fully consider the uncertainties mentioned above and are not in favor of these assumptions. To deal with these uncertainties, we formulate the computation peer offloading problem as a sequential decision problem. We investigate efficient computation peer offloading scheme with the minimal latency requirement based on combinatorial multi-armed bandit (CMAB) scheme. The proposed method endeavors to learn the statistical information of the network through historical feedback. Besides, numerical analysis validate the proposed computation offloading scheme.

The rest of the paper is organized as follows. Section 2 introduces the system model and formulates the optimization problem. An online learning algorithm for computation offloading is proposed in section 3 and numerical results are given in section 4. Finally, section 5 concludes the paper.

2. SYSTEM MODEL AND PROBLEM FORMULATION

In this work, we consider a fog network, defined by a set $\mathcal{N}$, consisting of N FNs. Each FN can cooperate with other neighboring FNs. FNs are assumed to be the roles of collecting, storing, controlling, and processing the task data from end-user devices, as is typical in practical fog networking scenarios [14]. In practice, the information exchange among the FNs will be difficult. The condition of each FN is changing constantly, so the exchanged information may be out of date to other FNs. Besides, how to coordinate the information exchange can be a challenging issue.

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61671436 and the Science and Technology Commission Foundation of Shanghai under Grant No. 18511103502.

To simplify the discussion, we assume that the operational timeline is discretized into time slots, $t \in \{1, 2, ...\}$. Once FN i receives x_i^t task, it decides how to offload the task. The incoming task can be divided into several parts and FN i allocates the partial task to different FNs (including itself). Let α_{ij}^t ($j \in \mathcal{N} \setminus i$) denote the fraction indicator between FN i and j, which indicates that FN i offloads $\alpha_{ij}^t x_i^t$ task to FN j. Notation α_{ii}^t represents the fraction of local computing. The task distribution variables are represented as vector $\boldsymbol{\alpha}_i^t = \left[\alpha_{i1}^t, ..., \alpha_{ii}^t, ..., \alpha_{iN}^t\right]$.

Next we consider the delay of peer computing and local computing. Peer computing delay can be mainly divided into three parts: transmission delay of dispatching tasks, computation delay and transmission delay of downloading computation results. The transmission delay of downloading can be negligible due to relative small sizes of computation results [7]. Given transmission power $P_{tx,i}$ of FN i, the achievable (approximate) transmission rate is given by the Shannon channel capacity,

$$r_{ij} = W \log_2 \left(1 + \frac{P_{tx,i} G_{ij}}{\sigma^2}\right), \tag{1}$$

where W is the channel bandwidth, σ^2 is the noise power and G_{ij} is the channel gain between FN i and j. Different from previous work, the channel condition in this work is assumed to be time-varying and FN i can not obtain the accurate channel state information (CSI). This assumption is practical under the consideration of minimal latency. The transmission power $P_{tx,i}$ and W are static in our work. For the transmission rate r_{ij}, it changes over time. We denote $\widetilde{a}_{ij}^t$ as the average transmission rate during the transmission period T_j. So the transmission delay is

$$T_j = \frac{\alpha_{ij}^t x_i^t}{\widetilde{a}_{ij}^t}. \tag{2}$$

When the task $\alpha_{ij}^t x_i^t$ arrives at FN j, it will experience a waiting period before previous tasks in the queue are processed. Without loss of generality, FNs obey the rule of first in first out (FIFO). The computing service rate μ_j is fixed and available to all FNs as a prior. On the other hand, due to the stochastic characteristics of arrival tasks, the queue lengths of other FNs are unaware to FN i when making the offloading decision. Let Q_j^t denote the current computation queue length of FN j when task $\alpha_{ij}^t x_i^t$ arrives. FN i can download the computation results from its neighbor node j when FN j finishes the execution of Q_j^t and $\alpha_{ij}^t x_i^t$. Meanwhile, the queue length and transmission rate $\widetilde{a}_{ij}^t$ at current time slot are observed. Based on the above discussions, the computation latency at FN j is

$$C_j = \frac{Q_j^t + \alpha_{ij}^t x_i^t}{\mu_j}, \tag{3}$$

and the total peer computing delay is

$$D_j = T_j + C_j. \tag{4}$$

The peer computing delay can not be infinite in practice, so we let it be bounded in $[0, T]$, where T is the maximum delay that FN i can tolerate.

Let μ_i denote the local fixed computing service rate and Q_i^t as the current queue length at FN i. Similarly, the local computing latency can be defined as

$$D_i = \frac{Q_i^t + \alpha_{ii}^t x_i^t}{\mu_i}. \tag{5}$$

The online task distribution problem can be formulated to minimize the maximum latency when computing an incoming task of FN i,

$$\min_{\boldsymbol{\alpha}} \quad \max(D_i, D_{j \in \mathcal{N} \setminus i}), \tag{6}$$

$$\text{s.t.} \quad \alpha_{ii}^t + \sum\nolimits_{j \in \mathcal{N} \setminus i} \alpha_{ij}^t = 1, \tag{6a}$$

$$\alpha_{ii}^t \in [0, 1], \alpha_{ij}^t \in [0, 1], \forall j \in \mathcal{N} \setminus i. \tag{6b}$$

3. LEARNING THE OPTIMAL COMPUTATION PEER OFFLOADING

In this section, we first discuss the relationship between the CMAB model and computation peer offloading problem. Then we propose our online learning algorithm for computation peer offloading.

Bandit problems are motivated from a variety of real world problems, such as, stock investment, medical trials and dynamic pricing. Though the bandit models are quite simple to solve the real world problems, these models still provide an insight to give efficient solutions to the problems [15].

To solve the optimization problem (6), the difficulty lies in how to learn the statistics of channel and the queue status of other FNs in real time. CMAB model can be applied to solve this problem. With CMAB model, the probed FNs can give feedback about the channel and queue status. Thus partial network information are updated in each iteration. Then a sequential decision problem can be formulated and optimized.

In the CMAB model, an agent gambles on a bandit machine with a finite set of arms, where each arm has unknown distribution. At each round, several arms defined as a super arm $\mathcal{S}$ ($\mathcal{S} \subseteq \mathcal{N}$) can be pulled simultaneously. In this work, we consider the semi-bandit feedback is available, meaning that the agent observes only the outcomes of pulled arms in one round of play [16]. The reward of the super arm depends on the outcomes of pulled arms. The CMAB problem is to decide which super arm to pull at each round in order to maximize the accumulated expected reward over time, and at the same time acquire knowledge about the the uncertainty of the bandit system. The expected values of the arms are estimated based on the instantaneous reward observations.

In our computation peer offloading model:

(a) The agent represents the FN i. At each round, the selected FNs that receive partial tasks are defined as a super arm $\mathcal{S}$ ($\mathcal{S} \subseteq \mathcal{N}$), or the super FN $\mathcal{S}$. In other words, the super FN $\mathcal{S}$ is defined to be the FN(s) whose fraction indicator $\alpha > 0$.

(b) The individual reward r_n^t ($n \in \mathcal{N}$) of each FN n in time slot t is related to the computation peer offloading delay as defined in (4).

Classical CMAB problem usually deals with one $i.i.d.$ variable. In our case, on the other hand, there are two unknown variables, i.e., the uncertainty of the channel status and the uncertainty of the load condition of other FNs. Another modification of our problem lies in that the agent is a special single arm with deterministic reward. The direct application of classical CMAB solutions ignores the utilization of FN i itself. To avoid ambiguity, we still call FN i the agent instead of a special arm. The individual reward r_n^t is defined as

$$r_n^t = \begin{cases} T - D_n, & \text{if } n \in \mathcal{S}, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

and the individual reward is also bounded with $[0, T]$. Let $R(\mathcal{S})$ denote the instantaneous reward when super FN $\mathcal{S}$ is played. We

have

$$R(\mathcal{S}) = \min\ (\boldsymbol{R}^t), \tag{8}$$

where $\boldsymbol{R}^t$ represents the set of $r_n^t, \forall n \in \mathcal{S}$. Equation (8) indicates that the reward of super FN $\mathcal{S}$ is determined by the minimum individual reward among the selected FN(s). It is a nonlinear reward function, and it is more complex than a simple summation of individual rewards of pulled arms [16].

Next we are going to study the optimal task distribution policy. Let us first introduce the following proposition.

Proposition 1. *If there exists a task distribution $\boldsymbol{\alpha}^*$ satisfying $D_k = D_l, \forall k, l \in \mathcal{N}$, then $\boldsymbol{\alpha}^*$ is the unique and optimal solution of problem (6).*

Proof of Proposition 1. Suppose there exists a different vector $\boldsymbol{\alpha}'$ that can achieve the same minimal latency as the task distribution $\boldsymbol{\alpha}^*$ does. For $\boldsymbol{\alpha}'$, we can find a certain FN A satisfying ${\alpha'}_A < \alpha_A^*$ (${\alpha'}_A \in \boldsymbol{\alpha}'$, $\alpha_A^* \in \boldsymbol{\alpha}$) yielding $D_A({\alpha'}_A) < D_A(\alpha_A^*)$. Due to the constraint (6a), there exists another FN B that satisfies ${\alpha'}_B > \alpha_B^*$ (${\alpha'}_B \in \boldsymbol{\alpha}'$, $\alpha_B^* \in \boldsymbol{\alpha}^*$), and $D_B({\alpha'}_B) > D_B(\alpha_B^*)$. Since $D_B({\alpha'}_B) > D_B(\alpha_B^*) = D_A(\alpha_A^*) > D_A({\alpha'}_A)$, the latency defined in (6) incurred by policy $\boldsymbol{\alpha}'$ is larger than $\boldsymbol{\alpha}^*$. Policy $\boldsymbol{\alpha}'$ can not achieve the same minimal latency as $\boldsymbol{\alpha}^*$, thus $\boldsymbol{\alpha}^*$ is optimal.

Furthermore, D_j is a monotonically increasing function with respect to α_{ij} since $\frac{\partial D_j}{\partial \alpha_{ij}} > 0$ and so does D_i. Hence, $\boldsymbol{\alpha}^*$ is unique. ■

Proposition 1 reveals the fact that if the elements of $\boldsymbol{R}^t$ are all equal, $R(\mathcal{S})$ can achieve the maximum reward; if Q_j^t and $\widetilde{a}_{ij}^t$ are known, the optimal policy will pull, at each round, the super FN with the highest reward.

The regret of policy $\boldsymbol{\alpha}$ is the difference between its expected accumulated reward and that of the policy which always pulls the best arm(s). Hence, the regret is a metric of the loss due to not knowing the reward profile of the arms. The desired algorithm is to find a policy with sub-linear regret [15]. An algorithm that has good theoretical results in terms of regret with nonlinear reward is the combinatorial upper confidence bounds (CUCB) algorithm [16]. Theoretical results in [16] show that the regret of the CUCB algorithm is bounded by $\mathcal{O}(\log(t))$.

Let A_j denote the number of times the FN j has been selected. Once FN j has been selected in one time slot, $A_j \leftarrow A_j + 1$, otherwise, $A_j \leftarrow A_j$. Our proposed algorithm based on the framework of CUCB is listed in **Algorithm** 1.

Algorithm 1 Water Filling CUCB (WFCUCB) for Computation Peer Offloading

1: For each FN j, choose an arbitrary super FN $\mathcal{S} \in \mathcal{N}$ such that $j \in \mathcal{S}$ and update variables A_j and $\overline{r}_j$.
2: $t \leftarrow N - 1$.
3: **while** true **do**
4: $\quad t \leftarrow t + 1$.
5: $\quad$ For each FN j, set $\hat{r}_j = \overline{r}_j + \gamma \cdot T \cdot \sqrt{\frac{3 \ln t}{2 T_j}}$.
6: $\quad$ **if** $\hat{r}_j > T$ **then**
7: $\quad\quad \hat{r}_j = T$.
8: $\quad$ **end if**
9: $\quad$ Obtain $\boldsymbol{\alpha}_i^t$ using **Algorithm** 2 with $\hat{r}_j$.
10: $\quad$ Choose $\mathcal{S}$ according to $\boldsymbol{\alpha}_i^t$ and update A_j and $\overline{r}_j$.
11: **end while**

Here $\gamma \in (0, 1]$, which is an attenuation factor. Step 1 and 2 of **Algorithm** 1 guarantee that every FN has been selected once in first $N - 1$ rounds. Notation $\overline{r}_j$ denotes the individual reward sample mean, and $\hat{r}_j$ denotes the perturbed version of $\overline{r}_j$. If $\hat{r}_j$ exceeds T, we simply replace it with T. The proposed WFCUCB algorithm does not utilize the estimates $\overline{r}_j$ to solve the computation peer offloading problem. Instead we utilize the perturbed versions $\hat{r}_j$. The perturbation in step 5 promotes the selection of FNs that are not selected frequently, by artificially increasing their expected reward estimates. Step 5 also indicates the tradeoff between exploration and exploitation: whether one should try some FNs that have not been selected much (exploration) or one should stick to the FNs that provide good reward so far (exploitation).

In our computation peer offloading problem, there are two uncertain variables, Q_j^t and $\widetilde{a}_{ij}^t$. The variable $\widetilde{a}_{ij}^t$ can be regarded as $i.i.d.$, in a dynamic network, the statistical characteristics of task arrival change over time, so that Q_j^t are not necessarily identically distributed. Here we assume Q_j^t are independent through time, but their distribution may change mildly from one time slot to another.

For FN i, the updated rules of estimated $\overline{Q}_j$ and $\overline{a}_{ij}$ in time slot K ($K \geq 1$) are as follows,

$$\overline{Q}_j = \frac{\sum_{t=\max\{1,K-\tau+1\}}^{K} \rho^{K-t} \cdot Q_j^t \cdot \mathbb{I}_{\{\alpha_{ij}^t > 0\}}}{\sum_{t=\max\{1,K-\tau+1\}}^{K} \rho^{K-t} \cdot \mathbb{I}_{\{\alpha_{ij}^t > 0\}}},$$

$$1 \geq \rho > 0, \tau > 0, \tag{9}$$

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{K} \widetilde{a}_{ij}^t \cdot \mathbb{I}_{\{\alpha_{ij}^t > 0\}}}{T_j}, \tag{10}$$

here ρ is a forgetting factor which represents and compensates the non-stationary characteristics of queue length, τ represents the sampling window length. Notation $\mathbb{I}_{\{\alpha_{ij}^t > 0\}}$ denotes the indicator function that returns 1 if $\alpha_{ij}^t > 0$ and returns 0, otherwise. In order to probe all FNs a sufficient number of times, we define the perturbed versions $\hat{Q}_j$ and $\hat{a}_{ij}$ as follows,

$$\hat{Q}_j = \overline{Q}_j - \frac{\hat{r}_j - \overline{r}_j}{\mu_j}, \tag{11}$$

$$\hat{a}_{ij} = \overline{a}_{ij}. \tag{12}$$

The introduction of $\hat{Q}_j$ and $\hat{a}_{ij}$ can balance the tradeoff of exploration and exploitation. Thus we can reliably estimate the associated queue length and channel condition.

The queue processing time of each FN is defined as, $T_j^{em} = \hat{Q}_j / \mu_j$ and the queue processing time of the agent is $T_i^{em} = Q_i / \mu_i$. Let $\boldsymbol{T}^{em} = [T_1^{em}, ..., T_i^{em}, ..., T_N^{em}]$, then we reorder $\boldsymbol{T}^{em}$ such that $T_k^{em} < T_l^{em}$ if $k < l, \forall k,\ l \in \mathcal{N}$.

Algorithm 2 Task Distribution Adaptation Algorithm

1: **for** $k = 1 : N$ **do**
2: $\quad$ Compute $\boldsymbol{\alpha}^*$ according to **Proposition** 1 among ranked top k FN(s) in $\boldsymbol{T}^{em}$ and obtain optimal delay time D.
3: $\quad$ **if** $D \leq T_{k+1}^{em}$ or $k = N$ **then**
4: $\quad\quad$ **return** $\boldsymbol{\alpha}^*$.
5: $\quad\quad$ Break.
6: $\quad$ **else**
7: $\quad\quad$ Continue.
8: $\quad$ **end if**
9: **end for**

Algorithm 2 determines the participants of computation peer offloading and their related workload. Its computation complexity is linear ($\mathcal{O}(n)$) and has negligible latency compared to the latency of local computing or peer computing.

4. NUMERICAL RESULTS

In this section, we consider a fog network with tasks arrival in FNs according to a non-stationary Poisson process with increasing arrival rate. The non-stationary Poisson process can be naturally regarded as the formation of the computation peak period, e.g., the beginning of office hour. Variable $\widetilde{a}_{ij}$ follows a uniform distribution in $(20, 30)$ MB/s in each time slot. Variable μ_{ij} and μ_i are generated from a uniform distribution in $(10, 30)$ MB/s. Besides, $\gamma = 0.1$, $\rho = 0.99$, $\tau = 10$ and T is 100 ms during the simulations.

To demonstrate the effectiveness of WFCUCB, we compare the performance of the proposed algorithm to following strategies: (i) **Local**: only local computing is considered for all arrival tasks; (ii) **Random**: the selected FNs are random and the amount of offloading tasks are determined in an average allocation strategy at each round; (iii) **Fixed**: all FNs are selected and the computation allocation of each FN is equal; (iv) **Optimal**: the optimal policy with the help of task distribution adaptation algorithm in hindsight.

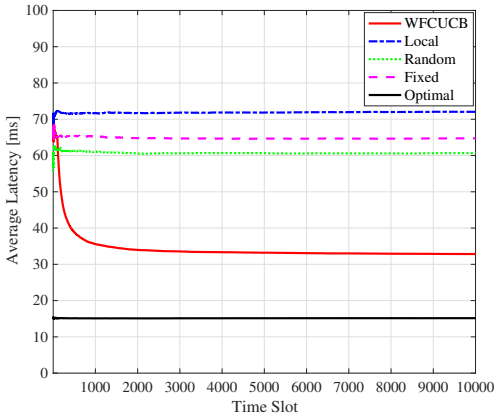

Fig. 1. Average latency of WFCUCB, local, random, fixed and optimal computation peer offloading, $x_i^t = 1$MB, $N = 100$.

Fig. 1 depicts the average latency of the proposed WFCUCB algorithm and other algorithms. We observe that WFCUCB achieves lower average latency compared to local, random and fixed algorithms, which does not require prior information of channel condition and the stochastic characteristics of arrival tasks. The convergence performance of WFCUCB represents the online learning process of other FNs' characteristics. Optimal strategy assumes the availability of global knowledge of the CSI and queue length of all other FNs and performs best among these algorithms. However, the overhead of acquiring such information prohibits its practical adoption in computation peer offloading.

Fig. 2 presents the impact of arrival task size in terms of average latency. The average latency performance of these algorithms is (approaching) stable after $10,000$ rounds. As the arrival task size increases, the average latency of local computing increases fastest among these algorithms which indicates the necessity of computation offloading. Another interesting observation is that local strategy performs better than random and fixed strategies when the task size is small, in which case the arrival task size is so small that it can be executed in time at local nodes.

The robustness of the proposed algorithm with respect to different network size N are depicted in Fig. 3. The simulation results reveal that the network latency of WFCUCB is much lower than random or fixed strategy. When the network size increases, the WFCUCB algorithm achieves better performance. It can be interpreted as the agent has more choices when the network size becomes larger.

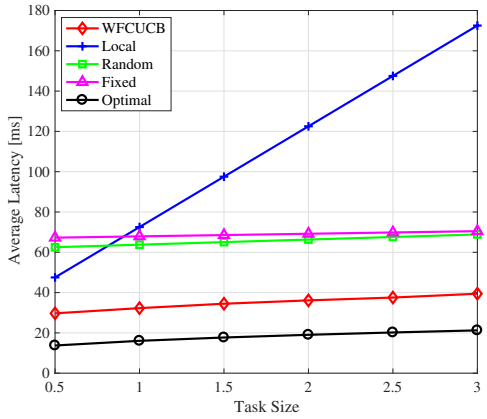

Fig. 2. Average latency of different arrival task size, $t = 10,000$, $N = 100$.

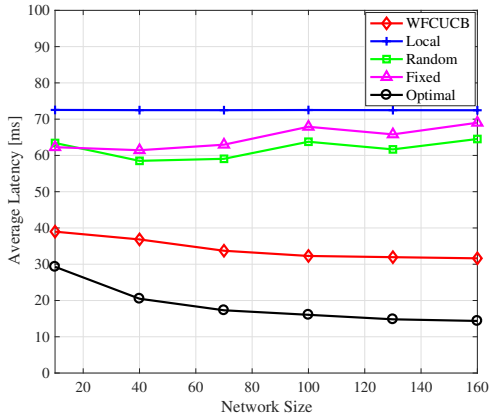

Fig. 3. Average latency of different network size, $x_i^t = 1$MB, $t = 10,000$.

5. CONCLUSIONS

In this paper, we have considered computation peer offloading in fog network with minimal latency requirement. The global knowledge of all FNs is unavailable in practice. We have modeled the problem as a CMAB problem without prior information about channel condition and the stochastic characteristics of arrival tasks. To solve this problem, we have proposed the WFCUCB algorithm, which extends classical CMAB problem to the case with one $i.i.d.$ variable and one non-stationary random variable. Numerical results have confirmed that the WFCUCB algorithm can learn fast and achieve better performance compared to other feasible strategies.

6. REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*. ACM, 2012.

[2] L. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[3] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, pp. 169–186. Springer, 2014.

[4] Lixing Chen, Sheng Zhou, and Jie Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.

[5] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and cpu time allocation for mobile edge computing," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Dec. 2016.

[6] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[7] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[8] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.

[9] L. Pu, X. Chen, J. Xu, and X. Fu, "D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted d2d collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

[10] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[11] G. Lee, W. Saad, and M. Bennis, "An online secretary framework for fog network formation with minimal latency," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, May 2017.

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

[13] H. Zhu, H. Wang, X. Luo, and H. Qian, "An online learning approach to wireless computation offloading," in *IEEE Global Conf. Signal and Inf. Process. (GlobalSIP)*, Nov. 2018.

[14] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[15] E. Hazan et al., "Introduction to online convex optimization," *Found. Trends Optim.*, vol. 2, no. 3-4, pp. 157–325, 2016.

[16] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2013, pp. 151–159.