A Novel Cross Entropy Approach for Offloading Learning in Mobile Edge Computing

Shuhan Zhu, *Student Member, IEEE,* Wei Xu, *Senior Member, IEEE,* Lisheng Fan, *Member, IEEE,* Kezhi Wang, *Member, IEEE,* and George K. Karagiannidis, *Fellow, IEEE*

***Abstract*—In this paper, we propose a novel offloading learning approach to compromise energy consumption and latency in a multi-tier network with mobile edge computing. In order to solve this integer programming problem, instead of using conventional optimization tools, we apply a *cross entropy* approach with iterative learning of the probability of *elite solution* samples. Compared to existing methods, the proposed one in this network permits a parallel computing architecture and is verified to be computationally very efficient. Specifically, it achieves performance close to the optimal and performs well with different choices of the values of hyperparameters in the proposed learning approach.**

***Index Terms*—Mobile edge computing (MEC), cross entropy (CE), computation offloading, probability learning.**

I. Introduction

With the rapid development of electronics and wireless networks, various services are currently supported by modern mobile devices (MD). However, most real-time applications require huge computation efforts from the MDs. Mobile edge computing (MEC) is a very promising technology to solve this dilemma for the next-generation wireless networks. According to MEC, edge servers, which are used to connect mobile terminals with a cloud server, provide high storage capability as well as fast computation ability.

In a MEC architecture both latency and energy consumption contribute to the network performance and it is of common interest to investigate the problem of balancing these factors with optimized offloading policies. Scanning the open literature, the authors in [1] proposed to offload a task from a single MD to multiple computational access points (CAP). Furthermore, a weighted sum of energy and latency was optimized by using convex optimization. This problem was recently extended in [2] to a scenario where multiple MDs perform offloading. The multiple tasks were scheduled based on a queuing state in order to adapt channel variations. Alternatively, in [3], the latency was minimized with scheduling the MEC offloading, while the energy consumption was considered as an individual constraint in MDs.

Recently, machine learning (ML) attracts much attention from both academia and industry, as an efficient tool to solve traditional problems in wireless communication [4]-[7]. Specifically, the authors in [4] proposed a payoff game framework to maximize the network performance through reinforcement learning. Furthermore, a deep Q-network was utilized in [5] to optimize the computational offloading, without *a priori* knowledge of the network. Most of these methods, including those which use deep learning network (DNN), focused on the offloading design from a perspective of long-term optimization and at the cost of complexity and robustness [6][7]. Moreover, these methods can hardly track fast channel changes, due to the requirement of offline learning. Thus, in general they cannot be applied for real-time applications in time-varying channel and it remains a problem of common interest to optimize offloading policies with a time-efficient method, which simultaneously ensures high-quality performance.

In this work, we introduce the *cross entropy* (CE) approach to solve the offloading association problem, by generating multiple samples and learning the probability distribution of *elite samples*. In contrary to the conventional algorithms, the proposed CE learning approach can use parallel computer architecture to reduce computational complexity, and it works for short-term offloading using online learning architecture, which has a stringent requirement on real-time evaluation. Our work generalizes the CE learning approach to solve the offloading problem with low complexity. The proposed approach is promising since it can effectively replace the traditional convex optimization tools.

II. System Model

We consider the problem of multi-task offloading in a network with multiple CAPs, where the MD has access to the CAPs. Each of the tasks can be selected to be executed at the local MD or offloaded to the CAPs, while a CAP serves only one task at each time. Since the index '0' represents the local CPU, $\mathcal{N} = \{1, 2, \cdots N\}$ and $\mathcal{M} = \{0, 1, \cdots M\}$ are defined as the sets of tasks and CAPs, respectively. In order to indicate the offloading status, we define the policy

$$x_{nm} = \begin{cases} 1, & \text{if task } n \text{ offloads to CAP } m \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Manuscript received October 1, 2019; revised November 25, 2019; accepted November 28, 2019. This work was supported by the National Key Research and Development Program 2018YFA0701602, by the Natural Science Foundation of Jiangsu Province for Distinguished Young Scholars under Grant BK20190012, by the NSFC under grants 61871109 and 61871139, and by the Royal Academy of Engineering under the Distinguished Visiting Fellowship scheme(DVFS21819\9\7). *(Corresponding author: Wei Xu.)*

S. Zhu is with the National Mobile Communications Research Laboratory, Southeast University, Nanjing, China (shzhu@seu.edu.cn).

W. Xu is with the National Mobile Communications Research Laboratory, Southeast University, Nanjing, China, and also with Purple Mountain Laboratories, Nanjing, China (wxu@seu.edu.cn).

L. Fan is with the School of Computer Science, Guangzhou University, China (lsfan@gzhu.edu.cn).

K. Wang is with the Department of Computer and Information Sciences, Northumbria University, Newcastle, UK (kezhi.wang@northumbria.ac.uk).

George K. Karagiannidis is with the Aristotle University of Thessaloniki, Thessaloniki 54 124, Greece (geokarag@auth.gr).

and matrix $\mathbf{X} = [x_{nm}|x_{nm} \in \{0,1\}]_{N\times(M+1)}$ ensembles all the indices. Assume that each task can be offloaded to a single CPU. In this case holds

$$\sum_{m=0}^{M} x_{nm} = 1. \tag{2}$$

A. Latency

The execution latency consists of two components: transmission latency and computation time. The transmission time includes task data preparation at the MD, data transmission duration over the air, and received data processing at CAP before conducting computation. Also, the transmission time depends on the achievable rate of physical links. The uplink and downlink data rates can be defined as,

$$R_m^{y_1} = \log_2\left(1 + P_{y_2}\eta\right), m = 1, \cdots, M, \tag{3}$$

where $y_1 \in \{\mathrm{UL}, \mathrm{DL}\}$, $y_2 \in \{\mathrm{T}, \mathrm{R}\}$, $\eta = h_m^{y_1}/N_0$. P_{T} (P_{R}) is the transmitting (receiving) power, and $h_m^{y_1}$ is the channel gain between CAP and MD. When it turns to the specific R_0^{UL} and R_0^{DL}, they are set infinitely large because computing at local CPU leaves out the process of offloading. Let α_n denote the input data size in bits, γ_n is the computation data size (number of cycles required for CPU) and β_n is the output data size after computation. Then, for the offloaded task n, the computation time, the uplink and the downlink transmission time can be

$$t_{nm}^{\mathrm{Comp}} = \frac{\gamma_n}{r_m},\ t_{nm}^{\mathrm{UL}} = \frac{\alpha_n}{R_m^{\mathrm{UL}}},\ t_{nm}^{\mathrm{DL}} = \frac{\beta_n}{R_m^{\mathrm{DL}}}, \tag{4}$$

where the CAP m serves the tasks with a fixed rate of r_m cycles/sec.

In fact, the CAP can start computing after either one or all scheduled tasks are offloaded. Here, we consider computation after one task offloading completes. For this case, there is no intra-CAP overlap when evaluating the overall latency. This latency is simple in expression, but the following proposed algorithm is still effective for other general expression. The three steps, offloading, computing and transmitting, take place sequentially, which results in the overall latency at CAP m as follows

$$T_m\left(\mathbf{X}\right) = \sum_{n\in\mathcal{N}} x_{nm}\left(\frac{\alpha_n}{R_m^{\mathrm{UL}}} + \frac{\beta_n}{R_m^{\mathrm{DL}}} + \frac{\gamma_n}{r_m}\right). \tag{5}$$

Note that since all CAPs evaluate their tasks in parallel, the delay is the maximum one, given as

$$T\left(\mathbf{X}\right) = \max_{m\in\mathcal{M}} T_m\left(\mathbf{X}\right). \tag{6}$$

B. Energy Consumption

An MD consumes battery to compute the tasks locally or to transmit and receive the task data. The energy consumption in the two cases can be written as

$$E_1\left(\mathbf{X}\right) = P_0 \sum_{n\in\mathcal{N}} x_{n0} t_{n0}^{\mathrm{Comp}}, \tag{7}$$

$$\begin{aligned} E_2\left(\mathbf{X}\right) = {} & P_{\mathrm{T}} \sum_{m\in\mathcal{M}\setminus\{0\}} \sum_{n\in\mathcal{N}} x_{nm} t_{nm}^{\mathrm{UL}} \\ & + P_{\mathrm{R}} \sum_{m\in\mathcal{M}\setminus\{0\}} \sum_{n\in\mathcal{N}} x_{nm} t_{nm}^{\mathrm{DL}}, \end{aligned} \tag{8}$$

where P_0 denotes the energy for local computation. Then, the total energy consumption is

$$E\left(\mathbf{X}\right) = E_1(\mathbf{X}) + E_2(\mathbf{X}). \tag{9}$$

C. Optimization Problem

Low computational latency and energy consumption are two main objectives of MEC. Unfortunately, these objectives cannot be minimized simultaneously and the problem turns out to be a multi-objective optimization. We define the weights, λ_t and λ_e, to compromise the two objectives. Then, the weighted objective can be defined as [1]

$$\Psi\left(\mathbf{X}\right) = \lambda_t T(\mathbf{X}) + \lambda_e E(\mathbf{X}), \tag{10}$$

where $T(\mathbf{X})$ is defined in (6) as the maximum delay consumed by all the CAPs instead of the sum or average one.

We aim to solve computation resource allocation scheme under specific situation where λ_e and λ_t are fixed. The joint minimization problem of both power and latency can be formulated as

$$\begin{aligned} & \min_{\mathbf{X}}\ \Psi\left(\mathbf{X}\right) \\ & \text{s.t.} \sum_{m\in\mathcal{M}} x_{nm} = 1, \forall n \in \mathcal{N}, \\ & \quad x_{nm} \in \{0,1\}. \end{aligned} \tag{11}$$

III. Offloading Learning through Cross Entropy

The problem in (11) is a binary integer programming one, which can be optimally solved via the branch-and-bound (BnB) algorithm with exponentially large computational complexity, especially when $\mathbf{X}$ is large [7]. In future wireless networks, the number of tasks will increase and more CAPs will be involved. Then, the BnB algorithm can hardly satisfy the requirements of real-time applications. Besides, there are studies on trying to solve the problem by using conventional optimization methods. The most popular solution is to use convex relaxation, e.g. to relax $x_{nm} \in \{0,1\}$ as $x_{nm} \in [0,1]$ through linear programming relaxation (LPr) or to relax

$$T\left(\mathbf{X}\right) = \max_{m\in\mathcal{M}} T_m\left(\mathbf{X}\right) \text{ as } T\left(\mathbf{X}\right) \geq \max_{m\in\mathcal{M}} T_m\left(\mathbf{X}\right)$$

by semidefine relaxation (SDR) [1]. The relaxation, however, causes performance degradation compared to BnB algorithm.

Besides the above methods, the problem in (11) with discrete optimization variables can be solved by using a probabilistic model based method, in the way of learning the probability of each policy x_{nm}. The CE approach is a probability learning technique in the ML area [9], [10]. To solve (11), we propose a CE approach with adaptive sampling, namely adaptive sampling cross entropy (ASCE) algorithm.

A. The CE Concept

Cross entropy, also known in probability theory as Kullback-Leibler (K-L) divergence, serves as a metric of the distance between two probability distributions. For two distributions, $q(x)$ and $p(x)$, the CE is defined as

$$D(q||p) = \underbrace{\sum q(x)\ln q(x)}_{H(q)} - \underbrace{\sum q(x)\ln p(x)}_{H(q,p)}. \tag{12}$$

Note that in our proposed CE-based learning method $p(x)$ represents a theoretically-tractable distribution model that we try to learn for obtaining the optimal solutions, while $q(x)$ is the empirical distribution which characterizes the true distribution of the optimal solutions. Particularly, in machine learning, distribution $q(x)$ is known from observations and $H(q)$ is the entropy of $q(x)$, which leads to the equivalence of learning the CE in (12) and $H(q,p)$.

We are inspired by the definition of CE, a popular cost function in machine learning, to solve problem (11) via probability learning. We learn $p(x)$ by iteratively training samples, and then generate the optimal policy of $\mathbf{X}$ according to $p(x)$, which is close to the empirical one, $q(x)$.

B. The ASCE-based Offload Learning

For probability learning, the probability distribution function $p(\boldsymbol{x})$ is usually introduced with an indicator $\boldsymbol{u}$, e.g., $p(\boldsymbol{x},\boldsymbol{u})$ can be a Gaussian distribution and $\boldsymbol{u}$ contains its mean and variance [11]. Denoting that L equals to $N\times(M+1)$, the indicator $\boldsymbol{u}$ is a vector of L dimensions, defined as $\boldsymbol{u}=[u_1,u_2,\cdots,u_L]\triangleq[\boldsymbol{u}_0^T,\boldsymbol{u}_1^T,\cdots,\boldsymbol{u}_M^T]$ where $\boldsymbol{u}_m^T=[u_{1m},u_{2m},\cdots,u_{Nm}]$ and $u_{nm}\in[0,1]$ represents the probability of $\Pr(x_{nm}=1)$. With this method, we learn $p(\boldsymbol{x};\boldsymbol{u})$ by learning its parameter $\boldsymbol{u}$. Accordingly, $\mathbf{X}$ is vectorized as $\boldsymbol{x}=\left[\boldsymbol{x}_0^T,\boldsymbol{x}_1^T,\cdots,\boldsymbol{x}_M^T\right]$ where $\boldsymbol{x}_m^T=[x_{1m},x_{2m},\cdots,x_{Nm}]$. Following the Bernoulli distribution, we have the distribution function $p(\boldsymbol{x},\boldsymbol{u})$ as [13]

$$p\left(\boldsymbol{x},\boldsymbol{u}\right)=\prod_{l=1}^{L}{u_l}^{x_l}(1-u_l)^{(1-x_l)}. \tag{13}$$

According to (2), one task associates to at most one CAP. Thus if a task is assigned to one CAP, its probability of being associated to other CAPs becomes zero. Aiming to reduce the redundancy of generated samples, we divide one sample, i.e., a vector $\boldsymbol{x}$ of L dimensions, into $M+1$ independent blocks, $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, $\cdots$, $\boldsymbol{x}_M$, and each of them associates to one CPU, e.g., the feasible block $[x_{1m},\cdots,x_{Nm}]$ indicates the task assignment of tasks 1-N to CAP m. Let $\mathcal{G}$ denote the set of indices of the selected blocks in sampling and $\mathcal{T}$ is another set to store the samples satisfying the constraints in each iteration. We first uniformly choose g, an index in $\mathcal{M}\backslash\mathcal{G}$. To generate an $\boldsymbol{x}_g$ given g, we draw the entries of $\boldsymbol{x}_g$ according to the probability density function $p(\boldsymbol{x}_g,\boldsymbol{u})$. For each u_l in $\boldsymbol{x}_g$, it is drawn according to the Bernoulli distribution of parameter u_l. The indicator $\boldsymbol{u}_m$ of the remaining blocks in $\mathcal{M}\backslash\mathcal{G}$ is then adjusted based on $\boldsymbol{x}_g$, that is, if $x_{ig}=1$ we have $u_{im}=0$ for $m\in\mathcal{M}\backslash\mathcal{G}$. When the cardinality of $\mathcal{G}$, denoted as $|\mathcal{G}|$, is equal to M, one valid sample is generated. Note that we draw the sample, while the non-feasible samples are excluded on the way. All the valid samples gather in $\mathcal{T}$ and the sampling repeats until the cardinality of $\mathcal{T}$, denoted as $|\mathcal{T}|$, reaches S.

In the proposed CE approach, computations in each iteration can be conducted in parallel, while the iterations are implemented in sequential. As will be shown later in the simulation results, we can adjust the hyperparameters of the proposed algorithm, including $S, S_{\text{elite}}, \alpha$, to compromise between the amount of parallel computations per iteration and the number of iterations for convergence. This makes a flexible tradeoff between performance and latency.

Algorithm 1 : ASCE-based Offload Learning Algorithm

Initialize: $\mathcal{G}=\mathcal{T}=[\,]$, $\boldsymbol{u}^{(0)}=0.5\times\mathbf{1}_{L\times1}$.
for $t=0:T$
While $\mathcal{G}\neq\mathcal{M}$ and $|\mathcal{T}|<S$
Select an index g from $\mathcal{M}\backslash\mathcal{G}$;
Generate entries of $\boldsymbol{x}_g$ based on $p(\boldsymbol{x},\boldsymbol{u})$ and update $\mathcal{G}$, $\mathcal{T}$;
Adjust $\boldsymbol{u}_m$ where $m\in(\mathcal{M}\backslash\mathcal{G})$ based on $\boldsymbol{x}_g$;
end while
Calculate the objective $\{\Psi\left(\boldsymbol{x}^s\right)\}_{s=1}^{S}$;
Sort $\{\Psi\left(\boldsymbol{x}^s\right)\}_{s=1}^{S}$;
Select the minimum S_{elite} $\boldsymbol{x}^s$ as elites;
update $\boldsymbol{u}^{(t+1)}$ according to (17);
end for
Output: $\boldsymbol{x}$.

Now we take the CE in (12) as the lost function. It shows that the smaller $H(q,p)$ is, the smaller the distance between $q(x)$ and $p(x)$ is. This implies

$$\begin{aligned}\min H\left(q,p\right)&=\max\sum q\left(x\right)\ln p\left(x\right)\\&=\max\frac{1}{S}\sum\ln p\left(\boldsymbol{x},\boldsymbol{u}\right),\end{aligned} \tag{14}$$

where $q(x)$ is $\frac{1}{S}$, since the probability of each independent solution in the set of samples is $1/S$ where S is the cardinality of the set [9]. Regarding the problem in (14), the objective is equivalently to finding the optimal indicator $\boldsymbol{u}$ minimizing $H(q,p)$. During the tth iteration, S series of random samples $\boldsymbol{x}$, serving as candidates, are drawn according to probability $p(\boldsymbol{x},\boldsymbol{u})$. The feasible samples generated by the adaptive sampling are under evaluation. We evaluate the objective $\{\Psi\left(\boldsymbol{x}^s\right)\}_{s=1}^{S}$ of (11) and sort them as

$$\Psi\left(\boldsymbol{x}^{[1]}\right)\leq\Psi\left(\boldsymbol{x}^{[2]}\right)\leq\cdots\leq\Psi\left(\boldsymbol{x}^{[S]}\right).$$

Then, S_{elite} samples, i.e., $\boldsymbol{x}^{[1]},\boldsymbol{x}^{[2]},\cdots,\boldsymbol{x}^{[\text{elite}]}$, yielding the minimum objective, are selected as elites. Now, the best indicator $\boldsymbol{u}$ for policy $\boldsymbol{x}$ can be determined as

$$\boldsymbol{u}^*=\arg\max_{\boldsymbol{u}}\frac{1}{S}\sum_{s=1}^{S_{\text{elite}}}\ln p\left(\boldsymbol{x}^{[s]},\boldsymbol{u}\right). \tag{15}$$

Using (13) and (15) and by forcing $\frac{\partial H(q,p)}{\partial u_l}=0$, the saddle point can be evaluated as

$$u_l^*=\frac{1}{S_{\text{elite}}}\sum_{s=1}^{S_{\text{elite}}}x_l^{[s]}. \tag{16}$$

In the proposed learning algorithm, we choose the CE-based metric for updating the probability. Considering the randomness of sampling, especially when the number of samples is small, we update $\boldsymbol{u}^{(t+1)}$ in the $(t+1)$th iteration not only on the basis of $\boldsymbol{u}^*$ which is handled with (15) and (16), but also $\boldsymbol{u}^{(t)}$ learned in the last iteration. It follows

$$\boldsymbol{u}^{(t+1)}=\alpha\boldsymbol{u}^*+(1-\alpha)\,\boldsymbol{u}^{(t)}, \tag{17}$$

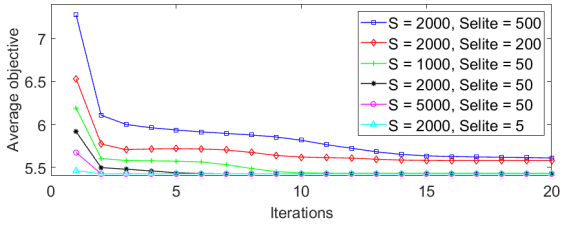

Fig. 1. Weighted objective against iterations with varying S and S_{elite}

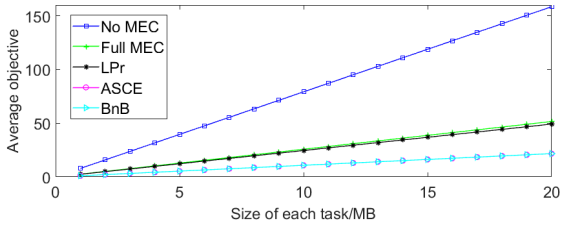

Fig. 2. Weighted objective under varying sizes of each task

where $\alpha \in [0, 1]$ is the learning rate [10]. In general, for the CE-based method, the iterations converge to an optimized solution of the problem [14].

The proposed algorithm is summarized in **Algorithm 1**. The CE approach combining with the indicator updating mechanism can replace conventional convex optimization methods, to compromise complexity and performance.

IV. Simulations and Discussion

This section validates the efficiency of the proposed approach through simulations, by using the same parameters as in [1]. The MD is equipped with a CPU and $r_0 = 200$ Mcycles/sec, $P_0 = 0.8$ W, $P_{\text{T}} = 1.258$ W and $P_{\text{R}} = 1.181$ W. The CPU frequencies of the three CAPs are $r_1 = 2 \times 10^9$, $r_2 = 2.2 \times 10^9$ and $r_3 = 2.4 \times 10^9$ cycles/sec. The data rates, R_k^{UL} and R_k^{DL}, are set to be 10 Mbps. The average objective in the figure results is the average value of the objective in (10) over a number of trials.

Fig. 1 shows the convergence of the proposed ASCE algorithm under various choices of hyperparameters S and S_{elite}. From Fig. 1, it is evident that the algorithm converges fast and the average objective reduces with S_{elite}, which can be considered as closer to the optimum one. Moreover, the average objective converges to almost the same optimal value for all the different choices of the values of hyperparameters. We therefore conclude that, the proposed ASCE algorithm performs robustly to different values of parameters.

In Fig. 2, we compare the proposed ASCE algorithm with the LPr-based offloading algorithm in [1], BnB [7], No MEC and Full MEC. Among them, No MEC and Full MEC represent that all the tasks are arranged to local CPU and CAP 1, respectively. The proposed ASCE algorithm greatly outperforms the LPr method and it approaches the theoretically globally optimal solution obtained by BnB. By contrast of "Full MEC" and "No MEC", "No MEC" is far inferior to "Full MEC", which implies that the MDs of multiple tasks can work efficiently with the assist of MEC. From [12], the complexity of the CE approach and BnB algorithm is $\mathcal{O}(L)$ and $\mathcal{O}\left(2^L\right)$,

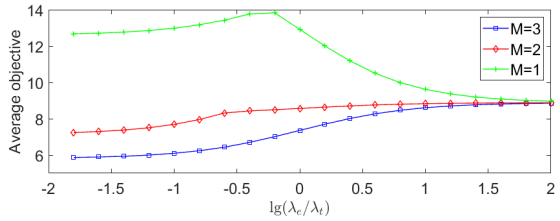

Fig. 3. Impacts of ratio of λ_e to λ_t on the weighted objective

respectively. The latter is far larger because the CE-method of parallel architecture optimizes L parameters in one iteration while BnB solves parameters sequentially. Besides, the BnB algorithm requires much more memory for storage.

The offloading policy is a tradeoff between latency and energy consumption to a certain extent. The value of $\frac{\lambda_e}{\lambda_t}$ is chosen to be 10^q, where q grows from -1.8 to 2 with step size 0.2. While $T(\mathbf{X})$ plays an increasing role in the objective function, the curve presents an increasing trend for $M = 2, 3$. As for $M = 1$, there is only one CAP serving the MD, which makes the minimized latency $T(\mathbf{X})$ much higher than the cases with multiple CAPs. Because the minimized $E(\mathbf{X})$ reduces to the energy consumption of all the tasks computed locally, which is the same for all different values of M, the curve of $M = 1$ finally decreases to the minimized $E(\mathbf{X})$.

V. Conclusion

In this paper, we present an efficient computational offloading approach for a multi-tier Het-MEC network. We propose the ASCE algorithm, which occupies less memory and has lower computational complexity than traditional algorithms. The proposed algorithm performs robustly, while it approaches closely to the optimal performance.

References

[1] T. Q. Dinh *et al.*, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[2] X. Lyn *et al.*, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2603–2616, Jun. 2018.

[3] J. Liu *et al.*, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE ISIT*, Barcelona, Spain, Jul. 2016.

[4] T. Q. Dinh *et al.*, "Learning for computation offloading in mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6353–6367, Dec. 2018.

[5] X. Chen *et al.*, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE IoT J.*, vol. 6, no. 3, Jun. 2019.

[6] C. Lu *et al.*, "MIMO channel information feedback using deep recurrent network," *IEEE Commun. Lett.*, vol. 23, no. 1, pp. 188–191, Jan. 2019.

[7] C. Lu *et al.*, "Bit-level Optimized Neural Network for Multi-antenna Channel Quantization," *IEEE Wireless Commun. Lett.*, accepted to appear, early access, 2019.

[8] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Trans. Comput.*, vol. C-26, no. 9, pp. 917–922, Sept. 1977.

[9] X. Huang *et al.*, "Learning oriented cross-entropy approach in load-balanced HetNet," *IEEE Wireless Commun. Lett.*, vol. 7, no. 6, pp. 1014–1017, Dec. 2018.

[10] P. D. Boer *et al.*, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, 2005.

[11] M. Kovaleva *et al.*, "Cross-entropy method for electromagnetic optimization with constraints and mixed variables," *IEEE Trans. Antennas Propag.*, vol. 65, no. 10, pp. 5532–5540, Oct. 2017.

[12] D. MacKay, *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.
[13] A. Vinciarelli, "Role recognition in broadcast news using bernoulli distributions," in *Proc. IEEE ICME*, Beijing, China, Jul. 2007.
[14] S. Mannor *et al.*, "The cross entropy method for fast policy search," in *Proc. ICICML*, Washington, Aug. 2003.